**RSA LANs**

**BY**

ESLAM ELSHARKAWY

ID: 201800473

Phone: 01066512891

Email: [s-eslam\_elsharkawy99@zewailcity.edu.eg](mailto:s-eslam_elsharkawy99@zewailcity.edu.eg)

CIE Major

MATH 308

**Course Instructor**

Dr.Ashraf Elsayed

**ACKNOWLEDGMENT**

*I’d like to show my gratitude to Dr.Ashraf Elsayed for his amazing way in explaining course content and I wish to attend more courses with such an amazing instructor again.*

**ABSTRACT**

*RSA is the best crypto algorithm has ever designed till now, it was such a mathematical phenomenon introduced by MIT professors, Now RSA is the backbone of the entire internet, It’s also used in many systems to provide more secures communication solutions, in this report i will introduce RSA based protocol that i will implement using* ***python*** *programming language, This protocol can be used in IOT servers, Some LANs and many other Computer Networks.*

**TABLE OF CONTENT**

**Chapter 1 | Project Outlines 5**

Introduction5

Problem Definition 6

The Recommended Solution 14

**Chapter 2 | Literature Review 16**

Review of Related Work 16

Overview of Implementation Tools 18

**Chapter 3 | Project Design 19**

System Design 19

System’s MATH 20

System Implementation 22

Testing 25

Target Model 27

**Chapter 4 | Conclusion 28**

**References 29**

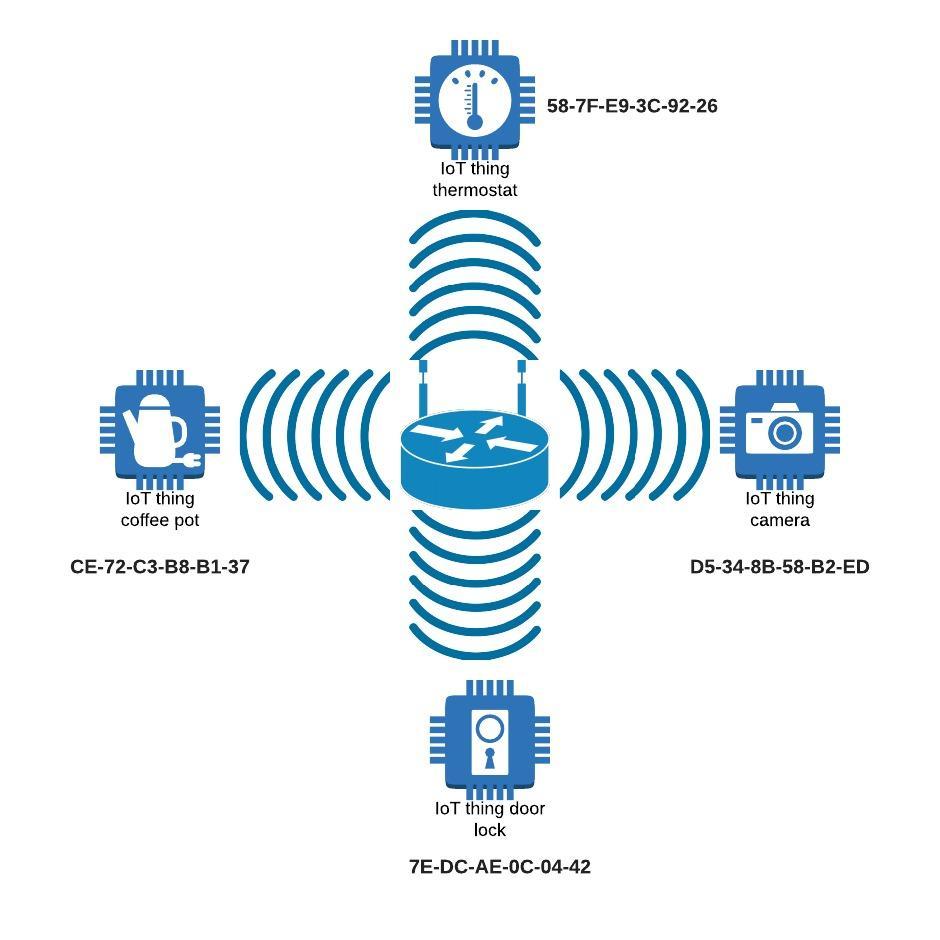
**Chapter 1 | Project Outlines**

**INTRODUCTION**

*This protocol will be used secure the client-server model based communications systems, the protocol will prevent anonymous login attack vector since the server will identify each client by their key, the system will also use the associated key sets with each connection to encrypt the channel and this will prevent the man in middle attack, the possible threats and how to exploit them will be introduced in the problem definition section, then i will introduce how the protocol protect against these threats alongside proof of concept using python code.*

**PROBLEM DEFINITION**

*Let’s imagine in of the client-server based models used in computer networks, for example let’s see the network that consists of modern routers (that have built in switch, modems and firewalls) and other devices on the network i will choose some IOT devices since they have low software based security level, the following diagram shows the network (let’s imagine the phones are the iot devices).*

**

**Figure 1: IOT Network Model**

**Possible Threat:**

*The problem with these IOT routers is that once the device is gained access to the wireless lan each packet generated is sent to its destination device via its MAC address, hence the router identifies the devices by their MAC address after connecting to the wireless LAN.*

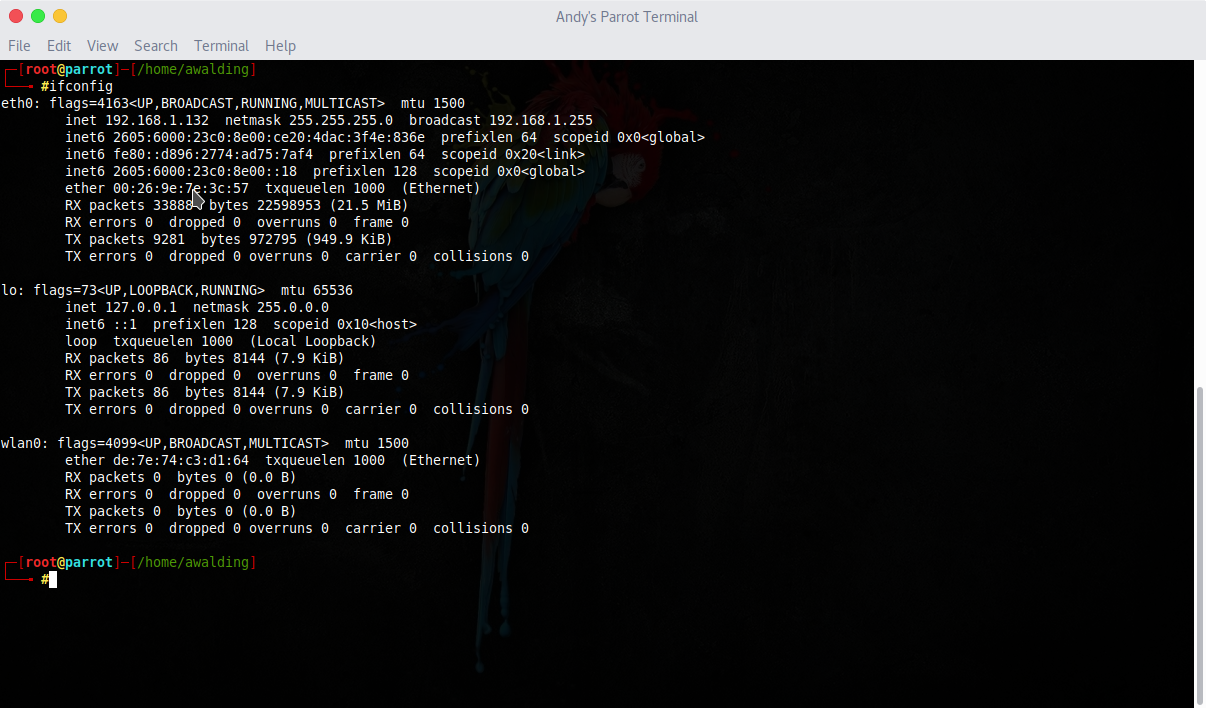
**Exploiting The Vulnerability:**

*Some security based linux distributions like* ***KALI*** *and* ***Parrot OS*** *have some tools to exploit such a vulnerability like tools to crack into WPA and WPA2 wireless networks, After the attacker gains access to the wireless LAN he can use another tool to clone one of the MAC addresses for example he can clone* ***D5-34-8B-58-B2-ED*** *which represents the camera in the network diagram in* ***figure 1****, Then the attacker will parse the stream of data as a camera data by splitting the data frames and constructing the images to make a live feed from that camera hence spy on the system, The attacker may also do more dangerous stuff like controlling or disabling vital parts in the system like the thermostat which can do it by cloning the MAC address of the central computer**then reverse engineer the communication protocol between the central computer and the MAC address* ***58-7F-E9-3C-92-26*** *via some dynamic analysis, After that he will be able to completely control such a vital device and many other possible attacks which can take many forms like different types of spying or harming.*

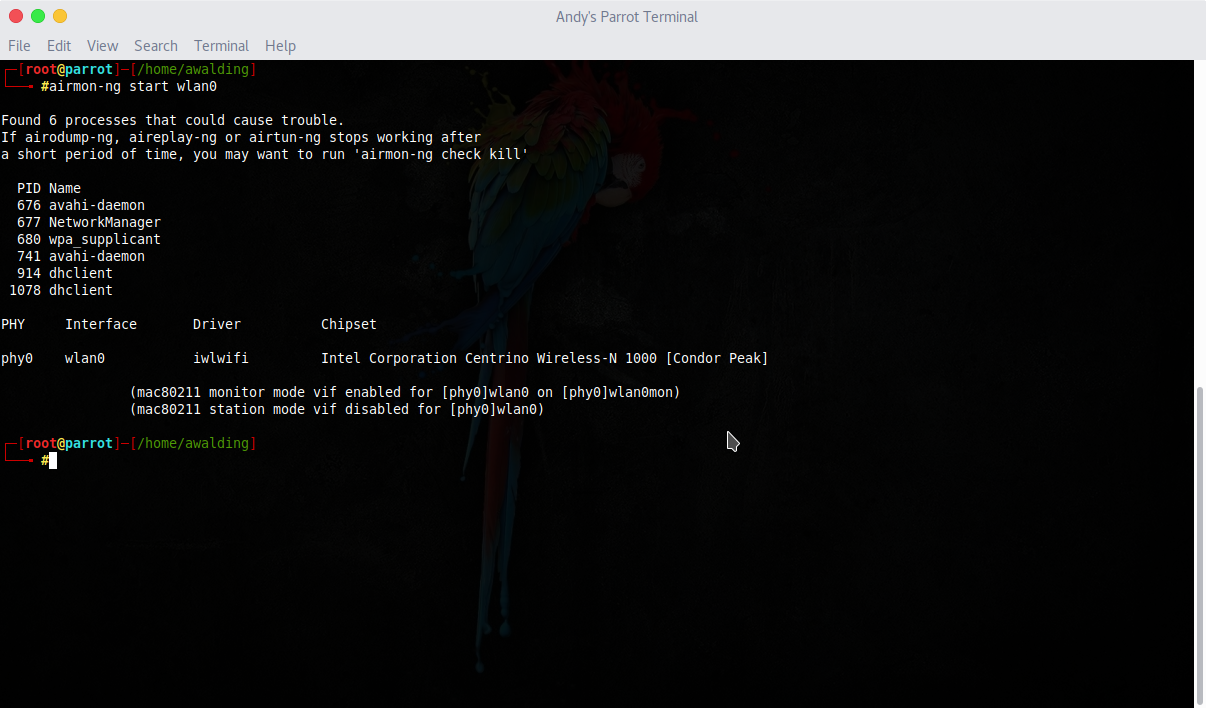
**Vulnerability POC (proof of concept):**

*To show this proof of concept**we need a security based linux distro like* ***KALI*** *or* ***Parrot OS*** *with a wireless card that supports monitor mode, My demonstration will be on* ***Parrot OS****.*

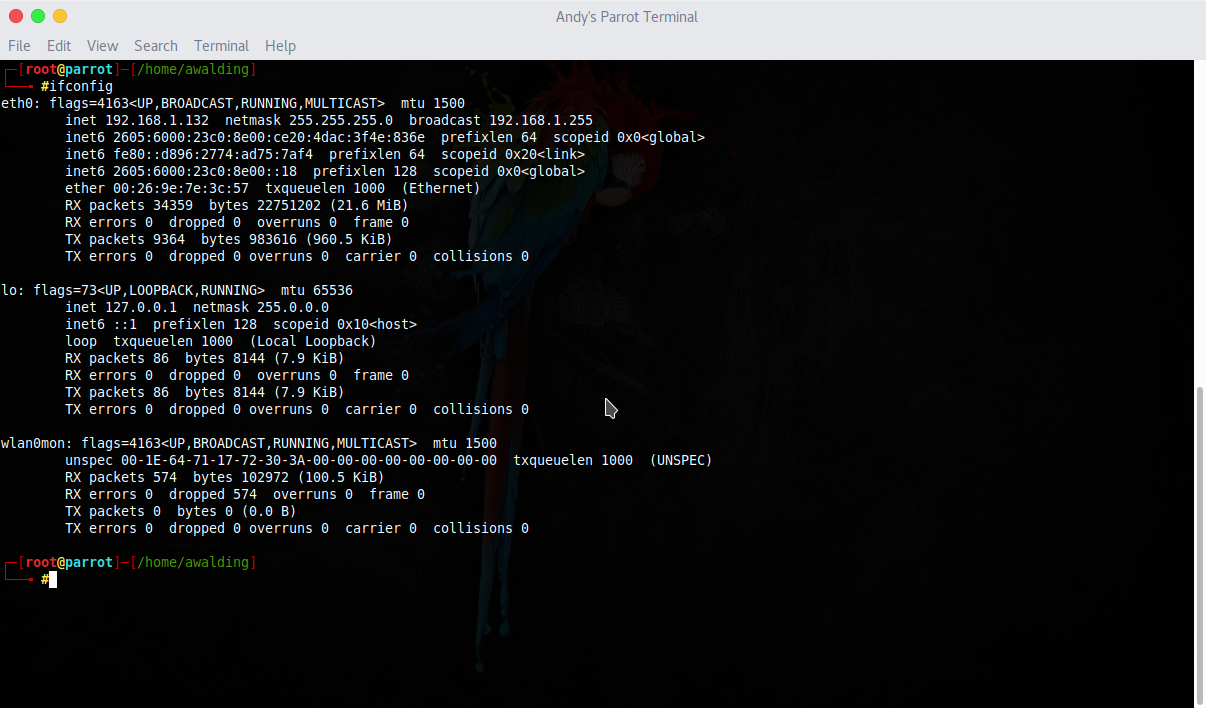
1. ***Setting up wireless monitor mode:-***

******

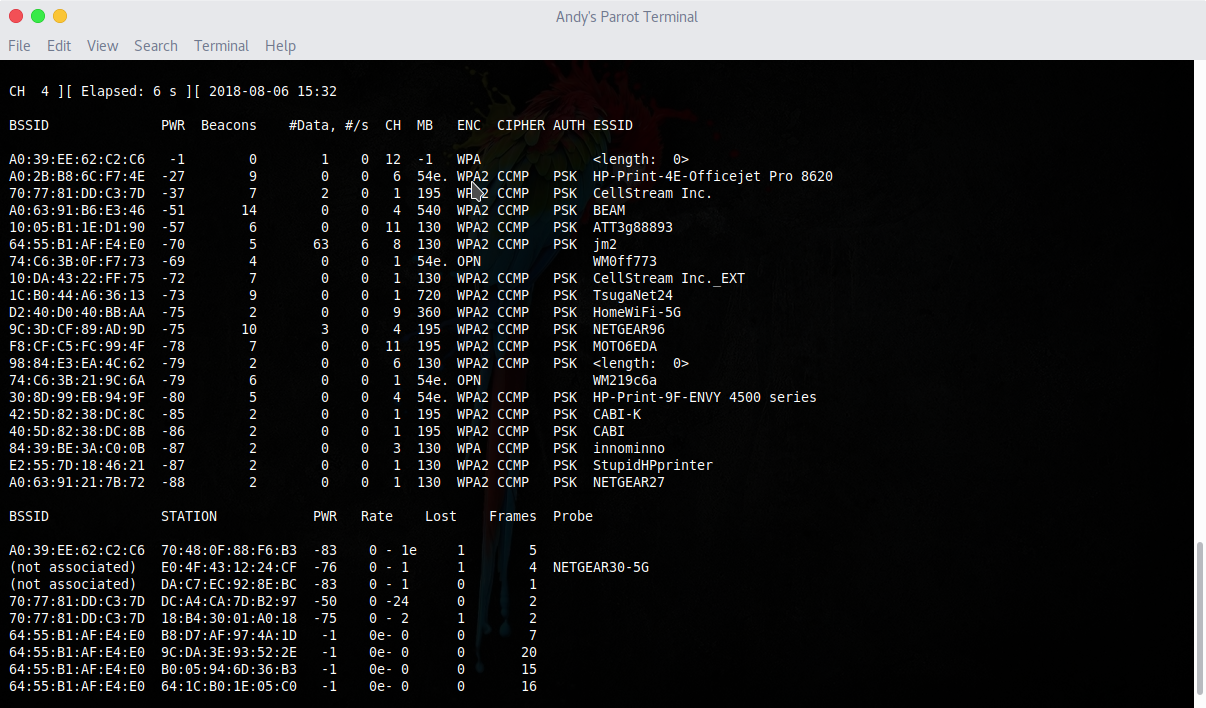
**Figure 2: Get Devices' Network Interfaces Info Command**

****

**Figure 3: Start Monitor Mode on The Wireless Interface Command**

****

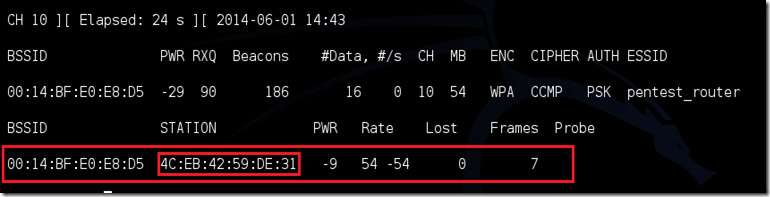
**Figure 4: Checking The New Configurations**

****

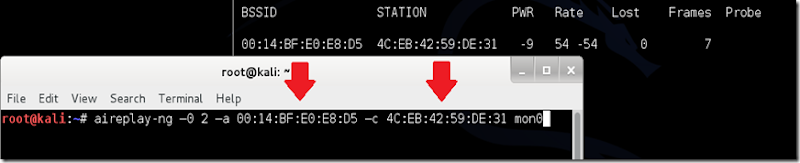
**Figure 5: Monitoring Available Wireless Networks Command**

1. ***Capturing the 4-way handshake***

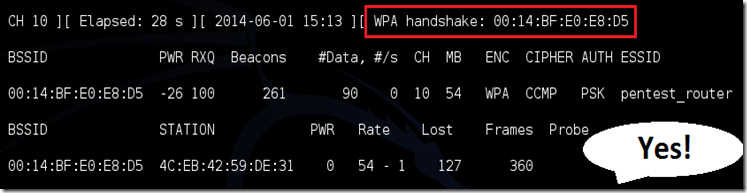
******

******

**Figure 6: Monitor on a Specific Network Command**

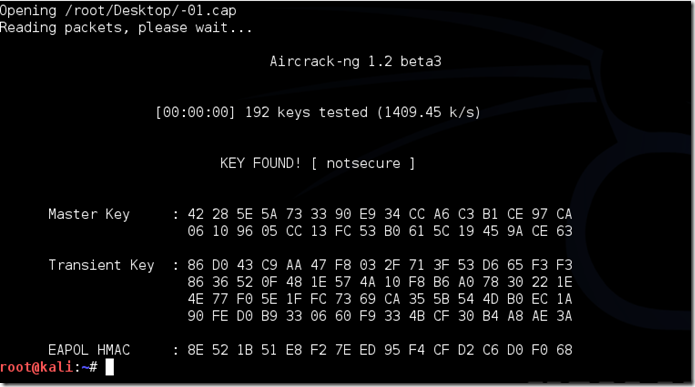
****

**Figure 7: Sending deauth Signals to Speed Things up**

****

**Figure 8: 4-way handshake Captured**

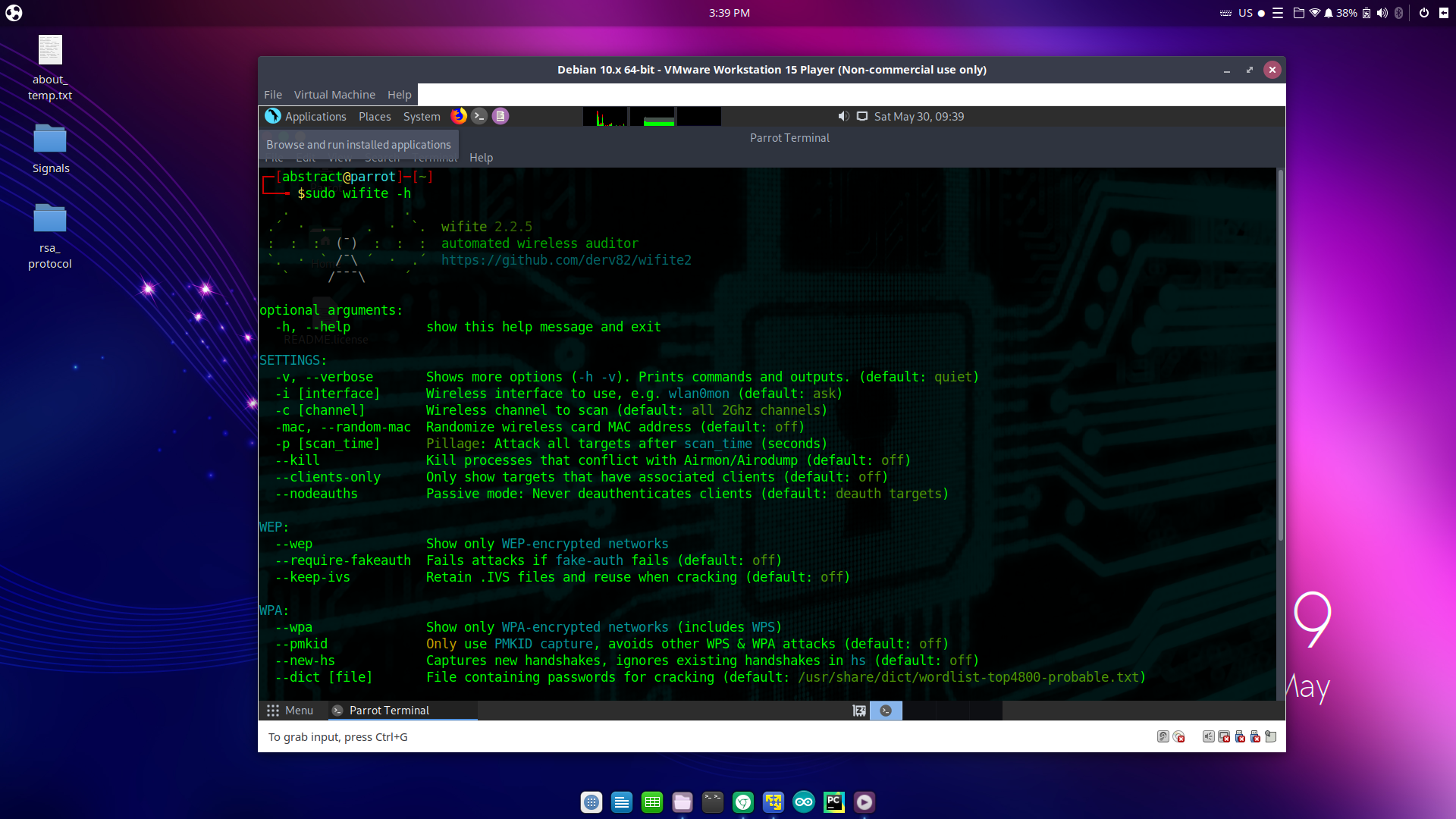
1. ***Cracking The Captured Packet***

******

**Figure 9: Cracking the handshake**

**Another Way To Crack a Wireless Network:-**

*Wifite tool is designed to automate the all wireless analysis process and choose the appropriate attack to execute the usage guide can be found in linux man pages or online, more info about the tool can be found in:* [*https://github.com/derv82/wifite2*](https://github.com/derv82/wifite2)

****

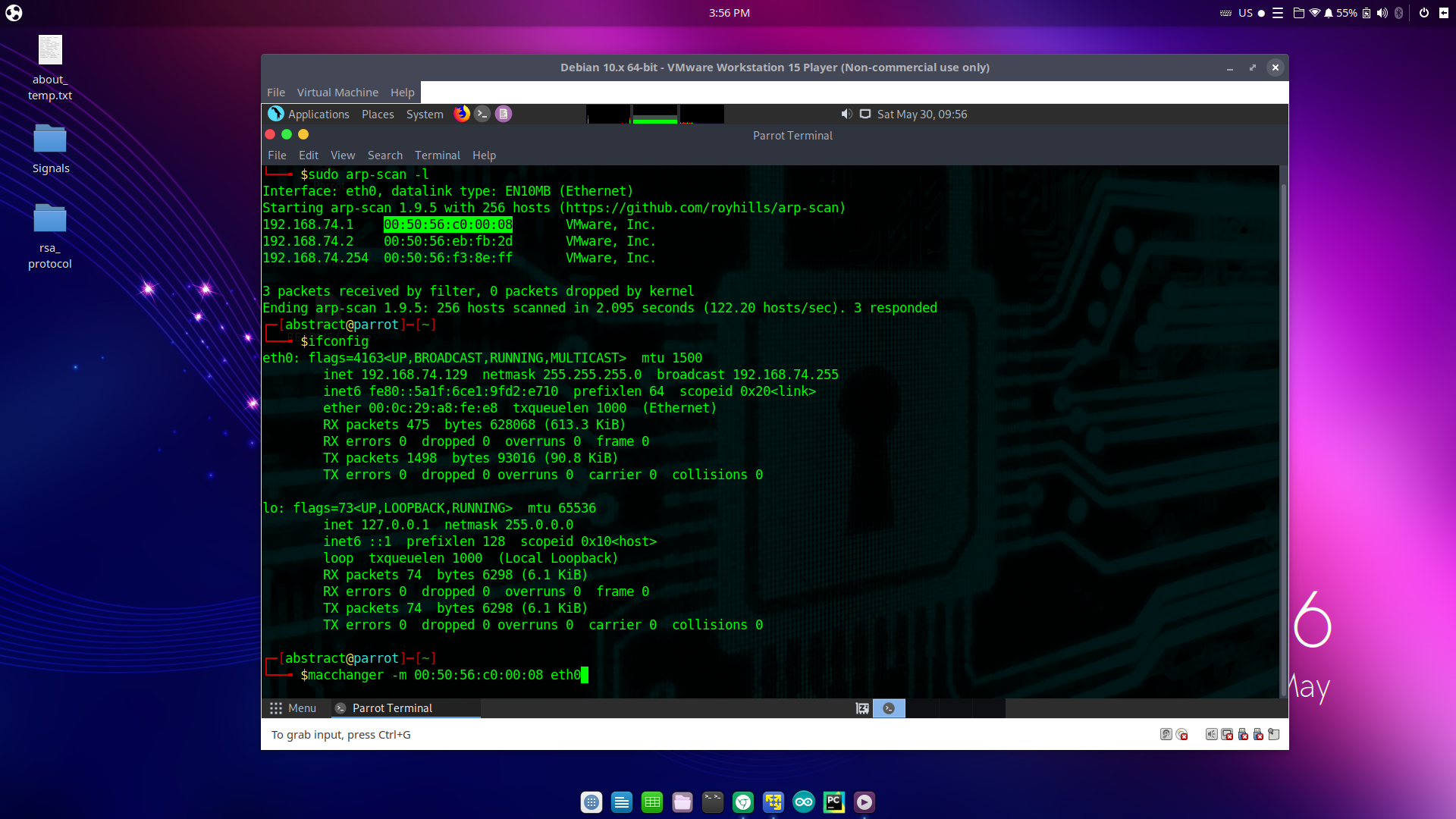
**Figure 9: Using wifite easy Tool**

1. ***MAC address Cloning***

******

**Figure 10: Scan MAC Addresses on Current LAN**

*After choosing the target MAC address make another deauth wave to disconnect the target, Then take its place as shown in* ***Figure 11****.*

******

**Figure 11: MAC Address Cloning**

*Now you can act as if you were that device or spy on other devices via simple networking scripts you can find them online or write them on your own if you know the fundamentals of programming and computer networking.*

**RECOMMENDED SOLUTION**

*First we will remove the MAC address and password authentication from the router’s firmware, Then put the* ***RSA Communication Protocol****.*

**The RSA Communication Protocol:**

1. **Authentication protocol:-**

*When a device request access to the network using an arbitrary id the router will respond with a random string, Now the client should encrypt that random string using its RSA key and send it back, now the router will try to decrypt the* ***CIPHERTEXT*** *if it matches the random string that was originally sent then the device is successfully authorized in the network****.***

1. **Transmission Protocol:-**

*Now the device has authorized access to the network there are two ways to secure the transmission* ***strong mode*** *which can be used in systems that don’t care about the network performance or* ***fast mode*** *which can be used in systems that the network performance is essential.*

* 1. **Strong Mode**

*Data between the two endpoints should be encrypted with it’s pair of RSA key, The other endpoint should decrypt it using its pair and so on.*

* 1. **Fast Mode**

*The two endpoints will use the constructed secure channel and exchange a symmetric encryption key via diffie hellman key exchange algorithm algorithm, Then use that key encrypt the data in this channel.*

*Now the attacker can’t gain access because he doesn’t have a registered RSA key and can’t intercept the communication because the channel is encrypted using asymmetric encryption like in* ***(strong mode)*** *or symmetric encryption* ***(fast mode)****, The new network diagram will be as shown in* ***figure 12****.*

**

**Figure 12: Secured Network Model**

**Chapter 2 | Literature Review**

**REVIEW OF RELATED WORK**

*RSA algorithm used in many IT applications to provide secure communications solutions, some of these solutions are presented from a company called RSA after the name of the algorithm, This company is one of leading companies in the math based cybersecurity industry, The following sections shows some of these solutions.*

1. ***Manage third-party risks:-***

*Using the RSA algorithm to manage third-party based systems achieves best practice in terms of security because each endpoint can be identified by its key which increases confidentiality on the network.*

1. ***Manage dynamic systems:-***

*Dynamic systems such as decentralized communication systems need good asymmetric algorithms to provide the system users secure compliance, authentication and resources access.*

1. ***Secure cloud transformations:-***

*RSA is used to secure your access to cloud resources and applications, specially when using third-party softwares to manage cloud based resources, For example for a computer to gain SSH access to a server one of the options is to have a registered RSA key to use it in authentications, These RSA pairs are generated by an engineer call sysadmin and distributed on different system users with different privileges.*

1. ***Manage task automation risks:-***

*Some RSA technologies are used to secure industry automations systems (IOT based industry networks) which provide reliability in complete digital transformations.*

1. ***Implement new identity access assurance:-***

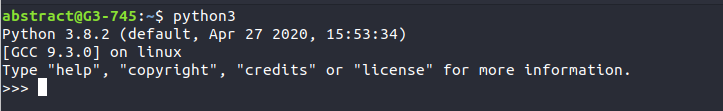
*Many systems now use RSA key pairs to authorize different users in a secure way, For example the SSH based RSA authentication, Some online websites use this technique too, this technique also provide protection from account takeovers (phishing attacks) since no regular credentials is required to login a system, network or a website, That helps making access to online platforms more convenient, easy and secure.*

**OVERVIEW OF IMPLEMENTATION TOOLS**

* **Python Programming Language:-**

*Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace, Python is dynamically typed and garbage-collected, which allow the code writer to focus more on the algorithm and how to solve the problem instead of the technical details of programming, so it’s wildly used in scientific research, testing and prototyping softwares before implementing them on a production level.*

* **Python Interpreter 3.8.2:-**

****

*Python interpreter is the software that will execute our python scripts.*

* **PyCharm IDE Community Edition:-**

*Pycharm is one of jetbrains company products that used to write and test python code, I will use pycharm IDE as a debugger to test and optimize my scripts.*

* **Visual Studio Code:-**

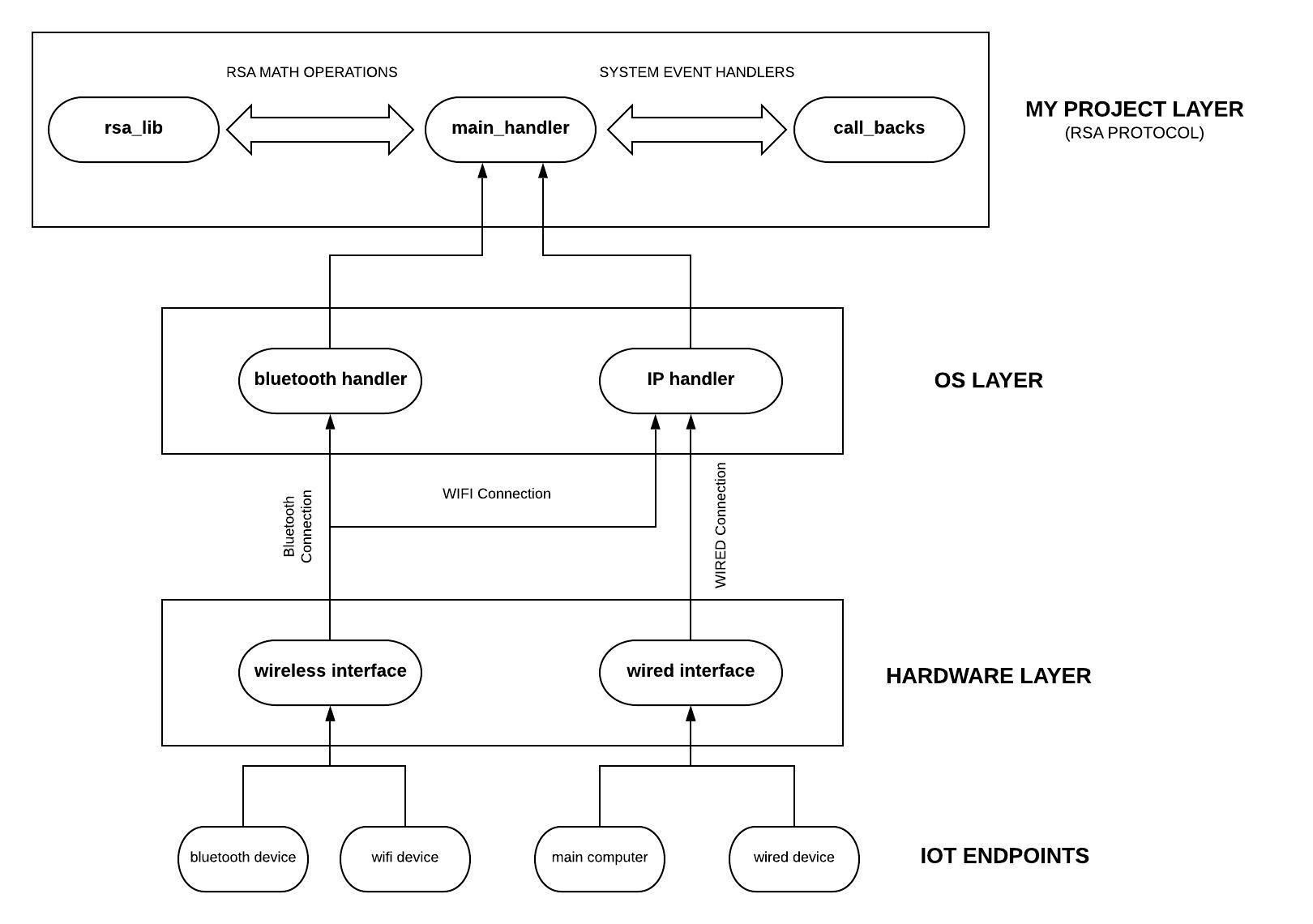
*Visual studio code is an open source code editor that has a lot of features, shortcuts and built in terminals and shells which helps in the software development process, it’s also light on system resources.*

* **Linux Bash Interbeter:-**
  + *Linux Bash interpreter is the modern shell in most of the modern linux distributions, i will use it to execute the python runtime binaries which will execute my python script in the process given to the python runtime from the operating system.*

**Chapter 3 | Project Design**

**SYSTEM DESIGN**

*Figure 13 shows the abstract system diagram and how different parts of the system will interact with each other, notice we splitted system call backs in a different module so we can provide the system with flexibility and integrability i.e. anyone can implement his system call backs easily and customize system’s behavior as he wants which make the system usable in many many fields not IOT only.*

**

**Figure 13: System Diagram**

**SYSTEM’S MATH**

**RSA Algorithm:-**

***Mills’ Primes generator:-***

P(n) = ⌊ ⌋

𝜇 = 1.3063778838630806904686144926026057129167845851567136443680537599664340537668

*Each RSA key pairs take the form* e = (E, N) *and* d = (D, N) *to find these values we follow the following steps:-*

*First we pick two very large primes (using mills’ generator for example)* P1, P2 then compute the following values:-

N = P1 \* P2

ɸ(N) = (P1 - 1) \* (P2 - 1)

(E) must be chosen as follows:-

1 < E < ɸ(N) & GCD(N, E) = 1 & GCD(ɸ(N), E) = 1

(D) must be chosen as follows:-

D \* E (mod ɸ(N)) = 1

**Diffie Hellman Key Exchange Algorithm:-**

* Let’s assume Alice tries to share same key with Bob
* Let’s assume there two public numbers every one knows **p, g**
* Alice picks random number **a**
* Bob picks random number **b**
* Then Alice computes **A =** and sends it to Bob
* Bob computes **X1 =**
* Then Bob computes **B =** and sends it to Alice
* Then Alice computes **X2 =**
* We notice **X1 = X2**
* Now Alice and Bob have the same key **K = X1 = X2**

*Notice that only the values*  **A =** *and* **B =** *transmitted, if someone intercepts the communication then he knows* ***A, B, p, g*** *but he can’t get* ***a, b*** *from this information because all the arithmetic operations were done under the modular so he can’t guess the secret key* ***K.***

**SYSTEM IMPLEMENTATION**

**Project File Structure:-**

**├── call\_backs.py**

**├── main\_handler.py**

**├── rsa\_db.txt**

**└── rsa\_lib.py**

**0 directories, 4 files**

**RSA Library:-**

**Euclid’s Algorithms:-**

* *Used in computing RSA key pairs and alos finding the multiplicative inverse of a number under some arbitrary modular.*

**def gcd(a, b):**

**while b != 0:**

**a, b = b, a % b**

**return a**

**def egcd(a, b):**

**if a == 0:**

**return b, 0, 1**

**else:**

**g, y, x = egcd(b % a, a)**

**return g, x - (b // a) \* y, y**

**Multiplicative inverse Algorithm:-**

* *This algorithm uses extended euclid's algorithm to compute the multiplicative inverse if a number* ***a*** *under modular* ***m.***

**def get\_mod\_inv(a, m):**

**g, x, y = egcd(a, m)**

**if g != 1:**

**raise Exception('[ERROR]: MODULAR INVERSE IS NOT EXIST!')**

**else:**

**return x % m**

**RSA Key Pairs Generator:-**

**def gen\_keypair(p, q):**

**n = p \* q**

**phi = (p-1) \* (q-1)**

**e = random.randrange(1, phi)**

**g = gcd(e, phi)**

**while g != 1:**

**e = random.randrange(1, phi)**

**g = gcd(e, phi)**

**d = get\_mod\_inv(e, phi)**

**return (e, n), (d, n)**

**Mill’s Primes Generator:-**

**def get\_rand\_mills\_prime(seed):**

**mu = np.float128(1.3063778838630806904686144926026057129167845851567136443680537599664340537668)**

**p = np.floor(np.power(mu, np.power(3, seed)))**

**return int(p)**

**Encryption and Decryption Functions:-**

**def encrypt(pk, plaintext):**

**key, n = pk**

**cipher = [pow(ord(char), key) % n for char in plaintext]**

**return cipher**

**def decrypt(pk, ciphertext):**

**key, n = pk**

**plain = [chr(pow(char, key) % n) for char in ciphertext]**

**return ''.join(plain)**

**RSA Database:-**

* *I used* ***rsa\_lib.py*** *script to generate five key to use in testing, Theses keys are stored in* ***rsa\_db.txt*** *file, Each key have the format* **client\_id:pub\_key:priv\_key:N***, notice that* ***N*** *is the same for all keys because theses are generated from the same primes base* ***P1, P2****.*

*11700:6719:8479:14971*

*11701:4779:12419:14971*

*11702:7683:747:14971*

*11703:9441:9761:14971*

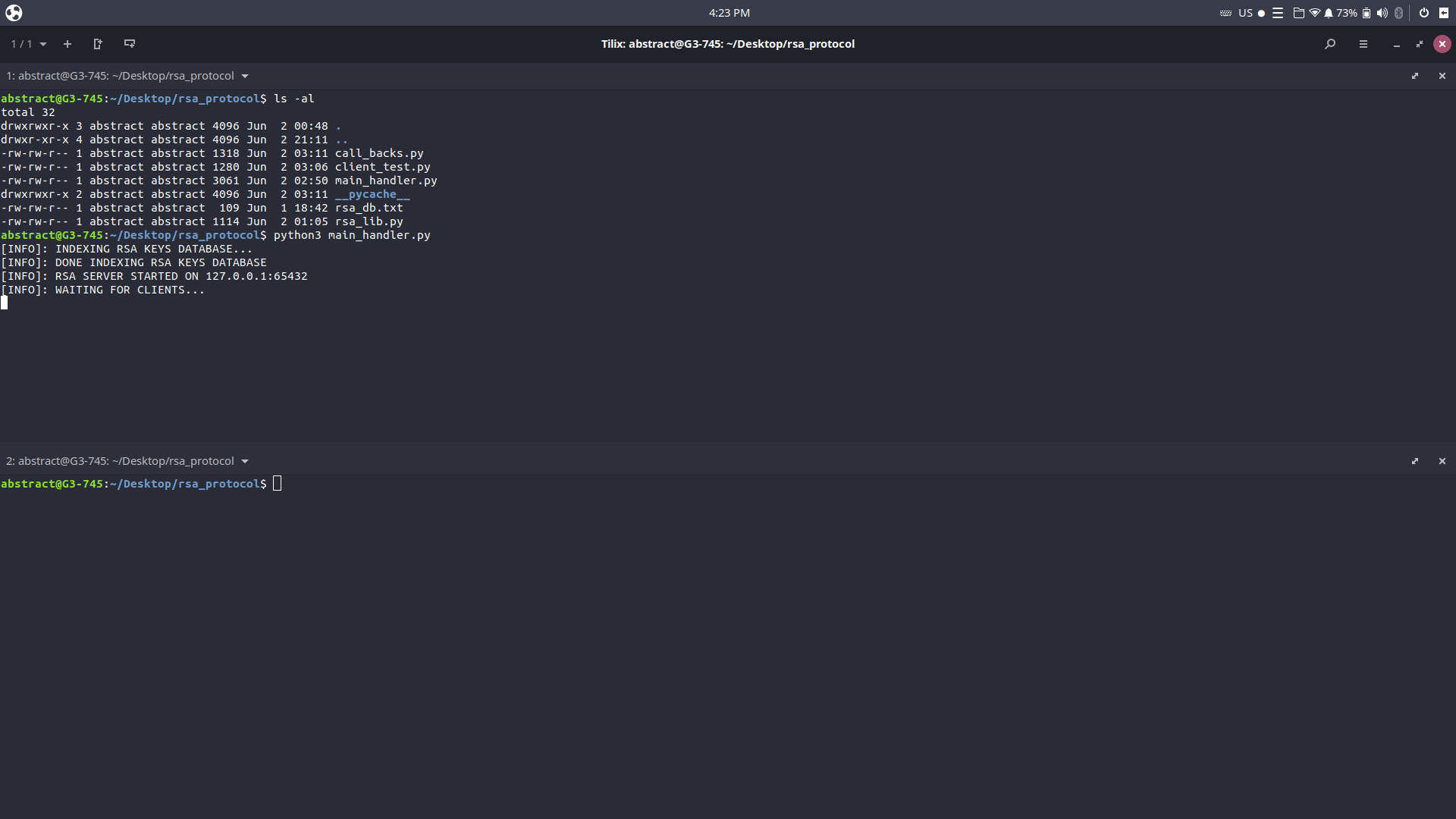
*11704:2679:2919:14971*

*Notes:-*

* ***client\_test.py*** *this file is not part of the system it’s just for testing purposes.*
* ***call\_backs.py*** *and* ***main\_handler.py*** *contains the system’s main behavior.*

**TESTING**

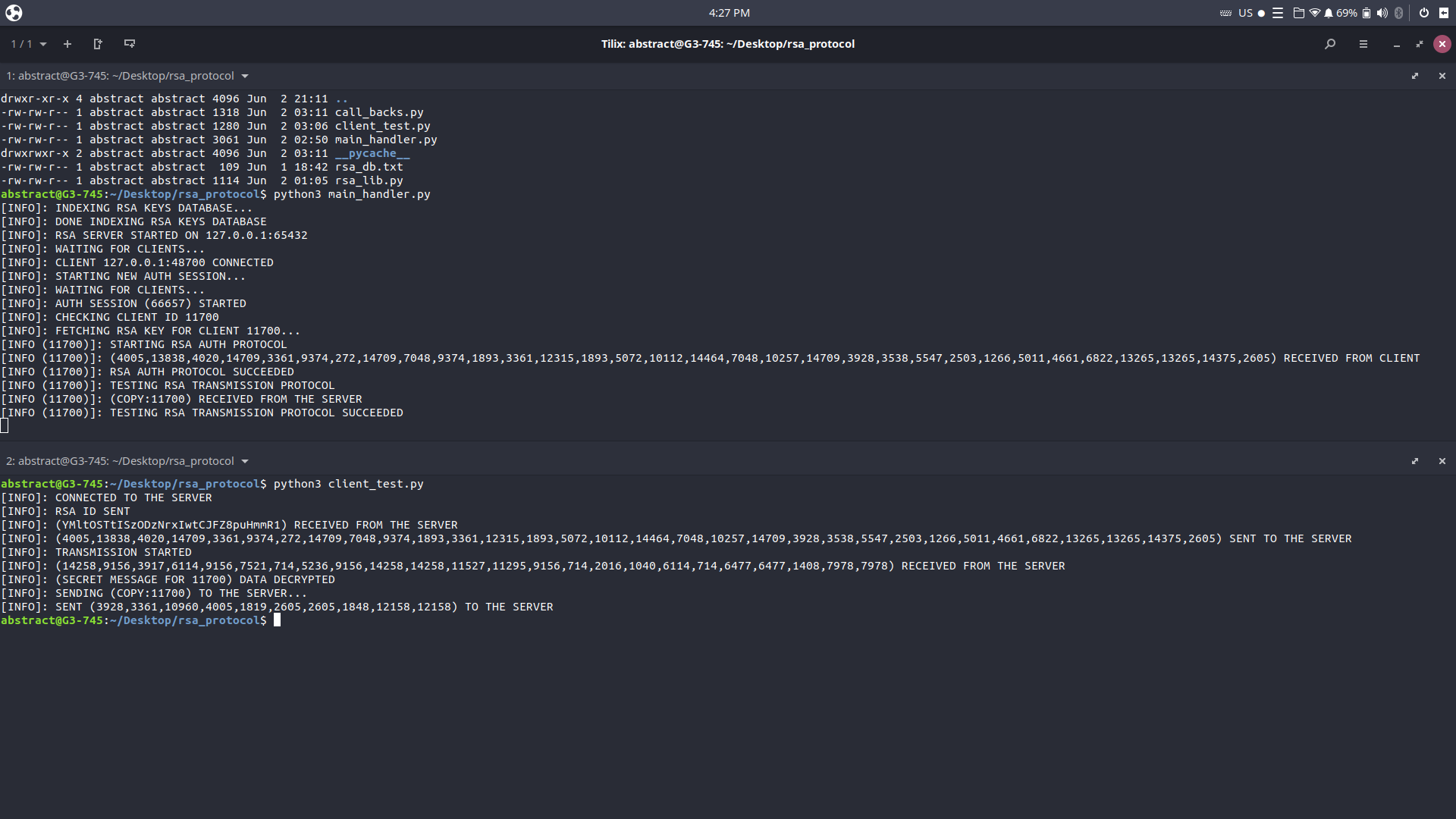
*Figure 14 shows the test screen, We notice two screens the top screen is the* ***server terminal,*** *the bottom screen is the client terminal.*

**

**Figure 14: Test Screen #1**

*We notice here that the server is listening on a local network level* ***(ip:127.0.0.1),*** *the server is binded on the port* ***65432.***

*Figure 15 shows the test screen, when the client starts connecting to the server.*

**

**Figure 15: Test Screen #2**

**Connection Workflow:-**

* *When the client connects to the server, The server sends a random 32 bytes string to the client this log shows that random string from the client terminal.*

**[INFO]: (YMltOSTtISzODzNrxIwtCJFZ8puHmmR1) RECEIVED FROM THE SERVER.**

* *Now the client should encrypt the that string using it’s RSA key and send the cipher back to the server, this log is from the client terminal and shows that the cipher text was sent to the server.*

**[INFO]: (4005,13838,4020,14709,3361,9374,272,14709,7048,9374,1893,3361,12315,1893,5072,10112,14464,7048,10257,14709,3928,3538,5547,2503,1266,5011,4661,6822,13265,13265,14375,2605) SENT TO THE SERVER.**

* *After the server receives the cipher text it decrypts it using the RSA key associated with the client id, if the text was the same then the* ***RSA AUTH PROTOCOL*** *is completed successfully, this log is from the server terminal and it shows that the server received the cipher from the client and decrypted it and found that it was the same text then reported that* ***RSA AUTH PROTOCOL SUCCEEDED****.*

**[INFO (11700)]: STARTING RSA AUTH PROTOCOL**

**[INFO (11700)]: (4005,13838,4020,14709,3361,9374,272,14709,7048,9374,1893,3361,12315,1893,5072,10112,14464,7048,10257,14709,3928,3538,5547,2503,1266,5011,4661,6822,13265,13265,14375,2605) RECEIVED FROM CLIENT**

**[INFO (11700)]: RSA AUTH PROTOCOL SUCCEEDED.**

* *Now the server tests the connection by sending a secret message to the client and then the client response back, This is called* ***RSA TRANSMISSION PROTOCOL,*** *the following logs shows that protocol on both client and server.*

**SERVER LOGS:**

**[INFO (11700)]: TESTING RSA TRANSMISSION PROTOCOL**

**[INFO (11700)]: (COPY:11700) RECEIVED FROM THE SERVER**

**[INFO (11700)]: TESTING RSA TRANSMISSION PROTOCOL SUCCEEDED**

**CLIENT LOGS:**

**[INFO]: TRANSMISSION STARTED**

**[INFO]: (14258,9156,3917,6114,9156,7521,714,5236,9156,14258,14258,11527,11295,9156,714,2016,1040,6114,714,6477,6477,1408,7978,7978) RECEIVED FROM THE SERVER**

**[INFO]: (SECRET MESSAGE FOR 11700) DATA DECRYPTED**

**[INFO]: SENDING (COPY:11700) TO THE SERVER...**

**[INFO]: SENT (3928,3361,10960,4005,1819,2605,2605,1848,12158,12158) TO THE SERVER**

*Note: if someone intercepts that connection he won’t be able to know that data because he don’t have a RSA key and if secured keys distribution then attackers won’t have any chance to hack the system.*

**TARGET MODEL**

**Optimization & Security Enhancements:**

* *System should be implemented in* ***C/C++*** *to increase the performance.*
* Use the RSA Optimized Library instead of implementing it manually.

**Adding Functionality:**

* **Adding Fast Mode:**
  + *Now only the strong secure mode is implemented in the system, but if we wanted the same level of security with higher connection speed, we can use the first secure transmission to achieve diffie hellman key exchange, That will be very useful when connection speed is very critical like some crisis handling IOT mechanism which is being controlled using continuous signal like a PID control for example.*

**Chapter 4 | Conclusion**

*To conclude, RSA is one of the greatest inventions in the math base information security world, the problem introduced in the problem definition section was the ability to hack IOT wireless network and taking over it completely, After applying the RSA solution hackers can’t gain access to the wireless network as before and if they did* ***(that won’t happen)*** *the mac address cloning attack will be useless because the system’s authentication and transmission is handled using the RSA key pairs and the the attacker won’t be able to do anything without these keys, if those keys fall in a hacker’s hand that will be a human mistake not a mistake from the system, Now we can say* ***PROBLEM SOLVED****.*

**REFERENCES:-**

**Kallam, S. (2015). Diffie-Hellman:Key Exchange and Public Key Cryptosystems. Retrieved from** [**http://cs.indstate.edu/~skallam/doc.pdf**](http://cs.indstate.edu/~skallam/doc.pdf)