

Cyber final project

Project’s subject: Anonymous games & communication platform based on onion routing

Student’s name: Yuval Didi

Student’s ID: 214422560

School and city: De-Shalit high school, Rehovot

Teacher’s name: Eran Bineth

Hand-in date: 17/4/2021

Table of Contents

[1 Introduction 3](#_Toc71649586)

[2 Theory 4](#_Toc71649587)

[3 Final Product 8](#_Toc71649588)

[4 The project-writing process 17](#_Toc71649589)

[5 Solution Components 19](#_Toc71649590)

[6 Testing Scenarios 26](#_Toc71649591)

[7 Reflection 27](#_Toc71649592)

[8 Instructions for installation and operation 28](#_Toc71649593)

[9 Bibliography 29](#_Toc71649594)

[10 Appendix 29](#_Toc71649595)

# 1 Introduction

## The subject

The subject of my project is an anonymous game and communication platform. The platform's name is **Odidion**. Odidion's users will be able to communicate with each other and play online games, using Odidion as an anonymous mediator. Odidion's communication is based on the Onion Routing protocol, which is known for its ability to verify a near-absolute anonymous communication.

## Main Goals

The main goals of this platform are:

* Constitute a generic global network, allowing programmers to easily promote their services, such as games, chat servers, and websites, conforming to the platform's APIs.
* Enable platform's users to transfer large data volumes such as files, movies, and pictures fast, besides facing the encryption overhead in time and msg length.
* Keep as little information as possible, about users and their services that use the platform, to ensure that they will enjoy maximum privacy.

## The Rational

I chose to develop an anonymous communication platform because I was and still am frustrated that nowadays our data becomes a product for the "big companies", and our privacy is compromised day-in and day-out. This frustration encouraged me to get familiarized with the field of anonymity and internet encrypted communication.

As I am interested in this subject, I also gave in my cyber class a presentation about Tor (one of the most popular browsers and platforms of Darknet) and onion routing. After this presentation, I decided that I will try my best to change "today's rules" by developing an anonymous communication platform as my final project.

In addition, I trust that this project will be a great opportunity for me to study new things such as the Scapy module and the encryption theory and its implementation, and also to improve my python programming skills.

## The reference to school material

To develop this project, I employed several tools and methods which, were taught in Cyber class, such as Scapy - to handle transferred packets, and Git to control, and correctly manage my code.

Besides, I made use of communication protocols, formats, and standards which I learned in my Cyber class, amongst which are UDP TCP, JSON, and others. All were used as a great knowledge-base for the development of the POC (Proof of Concept), and the exploration of additional subjects, tools, and theories to build the entire platform.

# 2 Theory

## Theory

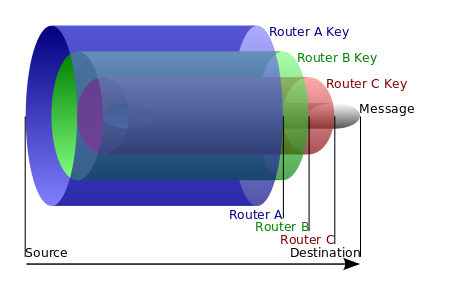
Onion routing is a routing method that was developed in the 1990's by DARPA and the U.S. Navy to protect the communication of the US online intelligence communications.

תמונה שמכילה טקסט, מכשיר חשמלי, טבע

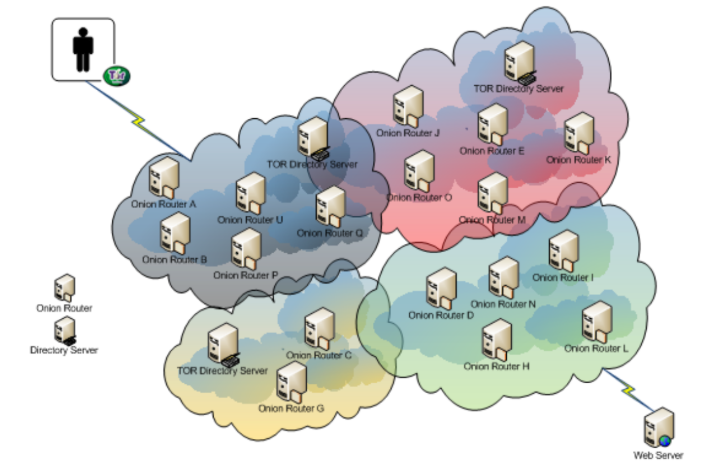
התיאור נוצר באופן אוטומטיThe common routing method which is used mainly on surface web (Google, YouTube, universities, and governments networks) requires transparent communication between source and destination, and therefore, even if encrypted, might be sniffed to disclose involved parties.

Unlike the common scheme, the "Onion Routing", is anonymous and each node in the routing path knows only what it must know, with no access to the sent data – its knowledge is limited to its direct sender and next router within the sending route.

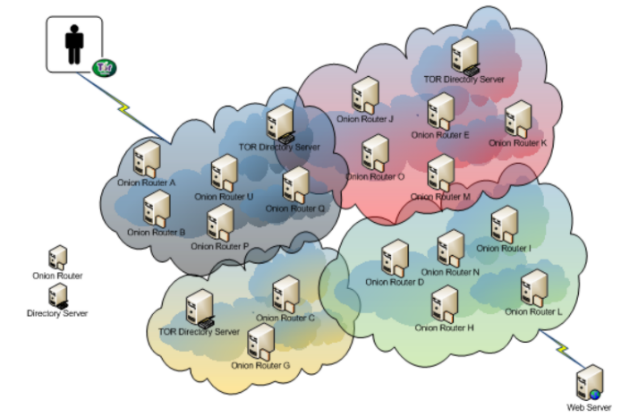
These features are implemented by encrypting all data each router should not "know", thereby creating a structure that looks like an onion, where the extraction of further layers requires encryption pealing on a need-to-know basis.

This platform has 4 main elements:

* Onion Directory Server – managing network and maintaining the functioning of the platform, by fairly dividing the traffic of the network between the onion routers. It also checks if all the onion routers are connected, and registers new onion routers and services to the platform.
* Onion Routers – implementing the onion routing scheme by transferring the encrypted packets across the network. These routers also act as "representatives" of the anonymous services across the platform, whereby data about those services (such as their IPs, PORTs) is known only by their representative routers (called Rendezvous Points).
* Services – such as websites, servers…
* Clients – initiators of communication. Building the encrypted packet according to the onion routing scheme and receiving data from the services as needed – the platform supports duplex and communication (the client receives only ACK's).

to build the packet correctly, the client (sender) should carry the following steps:

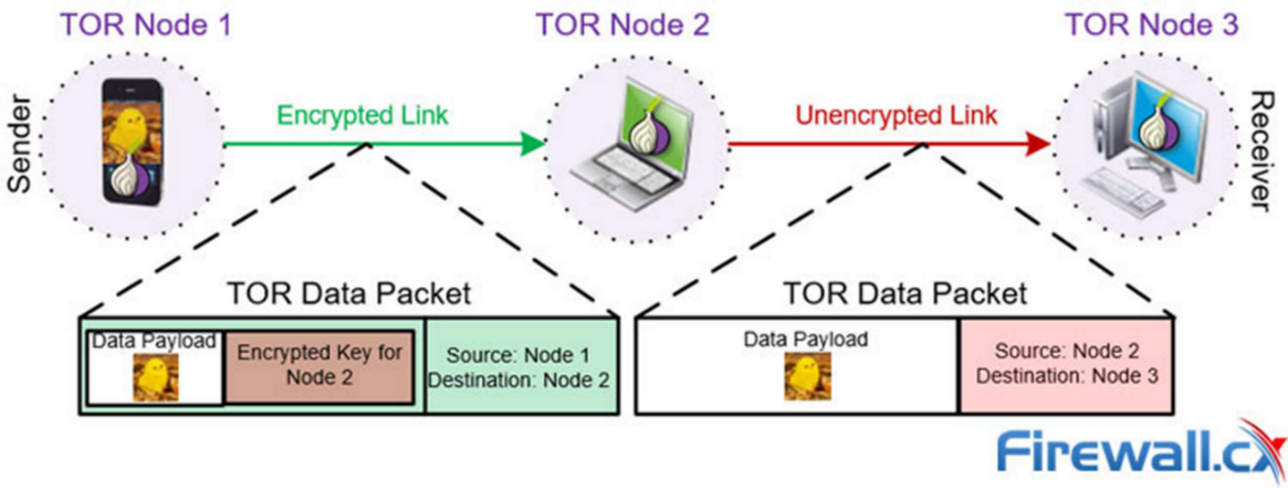
1. The client asks the Directory server for nodes in the network (onion routers) of his routing session.
2. The directory server responds with details about routers and the rendezvous point:

* Public asymmetric key (RSA)
* IP and port
* Rendezvous (the routers have serial numbers, the last one is the rendezvous point)

1. The client builds and encrypts the packet according to the Onion Routing scheme

תמונה שמכילה טקסט, מפה

התיאור נוצר באופן אוטומטי

The client creates the entire packet based on the envisioned route - each router will receive and pass it to its corresponding destination following decrypting its intended data (header of the next packet). Therefore, each router would be able to decrypt only the next packet header which is the data in the received payload.

1. תמונה שמכילה טקסט, מפה

   התיאור נוצר באופן אוטומטיThe client sends the packet or the group of the packets(according to the size of the sent data) to the first node in the routing path.
   1. Existing similar products

The most dominant platform in this field is TOR (The Onion Router) which provides a platform for anonymous communication based on the onion routing method. Two products that seem competitive with my project and act as a browser interface to the TOR network are ThankSoft and Eblocker.

[ThankSoft](http://thanksoft.com/) - MaskSurf (link - [URL](http://thanksoft.com/products/mask-surf-everything/)) – an application that anonymizes local applications communication, by implementing the Onion Routing Scheme towards the TOR network. The application manages to change the user's IP so the anonymity of the user is maintained.

[Eblocker](https://eblocker.org/en/how-eblocker-works/)(link - [URL](https://eblocker.org/en/how-eblocker-works/)) - a plugin/software that anonymizes the users' IP by routing the users' packets through VPN or TOR network. The software recognizes data collectors, ads, and malware based on patterns by checking the data packets.

Enclosed is a brief comparison between the three:

|  |  |  |  |
| --- | --- | --- | --- |
|  | ODIDION | ThankSoft | Eblocker |
| Full encrypted (from client to the destination) | V | V/X | V/X |
| Checks The Sent data | X | X | V |
| Decentralization | V | V/X | V/X |
| Based on Onion Routing | V | V | V/X |
| Open Source | V | X | X |
| Free | V | V/X | X |
| open source and easy maintenance | V | V | V |
| Anonymizes programs that interface the platform | X | X | V |
| Selection of a country-specific IP address | X | V/X | V |
| Performance | High | Regular | Regular |
| Anonymizer for LAN – the program anonymizes the communication through the LAN elements | V | X | X |

Odidion offers free, easy, and wide optional communication platforms for developers who want to create amazing client-server architecture-based softwares that will be totally anonymous and possibly faster several other anonymous softwares such as the compared ones.

# 3 Final Product

## Project description

The project is a platform for anonymous communication. The platform's goal is to enable anonymous access to websites and other services that interface the platform, such as anonymous games, chat servers, and online stores.

It comprises of

* A directory server presiding over the network and a distributed network of nodes, implementing the Onion Routing scheme. Each node that joins the network, downloads the onion router software, signs in to the platform's network, and subsequently takes part in the network routing scheme.
* client software that enables connection to the network by browsers or desktop clients, wishing to make use of its benefits. In order to interface the platform, clients should adhere to the APIs protocol specifications

In addition, the project provides supplementary services, such a blog server, an FTP server, etc., in order to portray the strength of the platform.

To implement this scheme effectively, the platform has to:

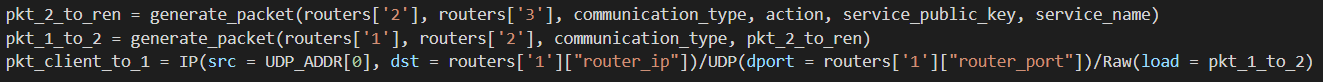
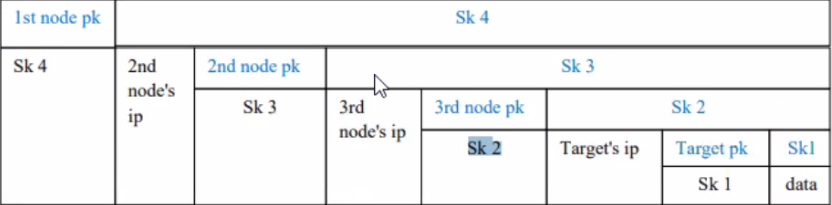
* Find the right balance between anonymity and efficient communication – the more efficient, the less anonymity because to make the communication more anonymous, the platform should use more variables and larger keys.
* Handle full-duplex communication despite the constraint that the service knows almost nothing about its client (client's IP / port).
* Be generic, and mimic open-source concept, to allow volunteers to contribute and upload new services.

The platform will present the following features:

* Develop Services – the platform offers an API that enables easy implementation of server-client programs based on the platform.
* Connect private anonymous services by the onion routing scheme, while verifying anonymous surf and hiding user identity.
  1. Main Algorithms

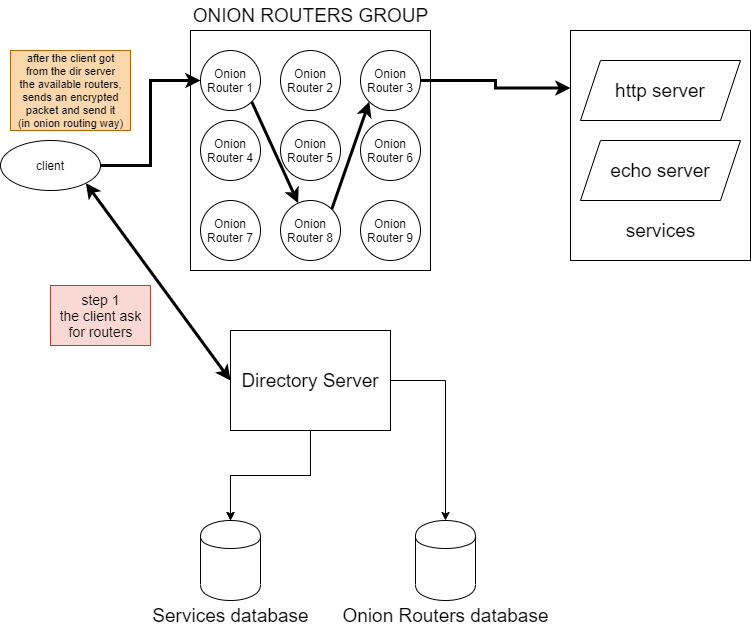
The main algorithms in my project handle the communication between the platform's elements and the implementation of the onion routing scheme.

1. Implementation of the Onion Routing (center algorithm):
   1. Step 1: The client software connects to the directory server which is known by all platform users, and asks for a minimum of three nodes to create an anonymous route, of which the service' rendezvous point is its end-point. The directory sends the client the following data about each node:

* Serial-num in routing path – (e.g. 3 if node is Rendezvous)
* IP and port
* Public asymmetric key
  1. Step 2: The client uses the received data to build the "onion packet" by encryption methods (as specified in *Theory).* In order to support lighter and longer packets, the client software creates the packet encryption layers accordingly :
     1. Generates a symmetric key and encrypts the next packet with service data. The keys are generated for each encryption layer (router), as well as the destination service, to ensure safe communication up to the Rendezvous point and the service provider as well.
     2. The client receives the asymmetric public key (RSA) of the platform element (onion router or service) from the directory server. It encrypts the symmetric key with the public key.
     3. Each encryption layer is added to the next encryption layer as the payload.
     4. The client follows steps 1,2,3 from the last layer backward (the client first creates the destination layer and last the first node layer).

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטיThe client program carries the following actions to build the onion packet:

* 1. Step 3: The client sends the encrypted packet or packets (depends on the length of the sent data) and waits for acknowledgments of the received data or replies from the destination (the service), meanwhile, the packets are transferred by the nodes.

1. Client-Service session:

The client connects to the directory server and asks for details about the chosen service. The directory server sends the client details about the service and the routing path including the rendezvous node, according to his request. The client software then, builds the onion packets and then the session starts.

תמונה שמכילה חץ

התיאור נוצר באופן אוטומטי

1. Register Router

תמונה שמכילה טקסט, אדום, שחור, כהה

התיאור נוצר באופן אוטומטיThis platform is based on its node volunteers in order to make sure that the anonymity is been kept. Each volunteer has to register the network by sending the following details to the directory server:

* 1. Router Name
  2. Ip
  3. Port
  4. Public key

The directory server saves the details above and starts a keep-alive session with the registered onion router.

Possible errors:

* The name of the new service is already taken
* Not all data was sent

the directory server will send an error alert and ask to change the router name/ send again the service data.

1. תמונה שמכילה טקסט

   התיאור נוצר באופן אוטומטיRegister service

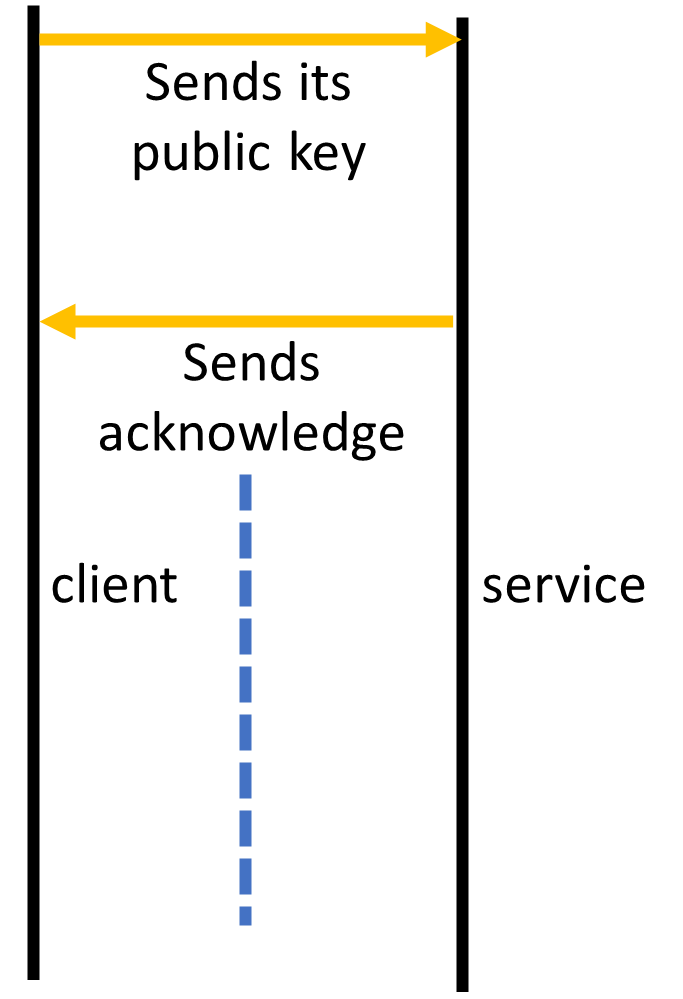
The service software sends its details to the directory server in order to get a rendezvous point (aka RenPoint) and be known by the directory server to be available in the network.

Possible errors:

* The name of the new service is already taken
* Not all data was sent

the directory server will send an error alert and ask to change the service name/ send again the service data.

1. Client – Service handshake:

Because the service cannot send replies without the client's public key, if some data has not arrived, the client will send again according to timeout.

Possible errors:

* The public key has not arrived.

the client will try again sending the public key 3 times according to the set timeout and if it will not succeed it will alert about a communication error.

* 1. Constrains and requirements

The platforms requirements for client, services and onion router are:

* Support multi-client and multi-service.
* Fast heavy data transferring.
* The Directory Server has to be online all the time
* At least 3 onion routers online

The platforms constraints are:

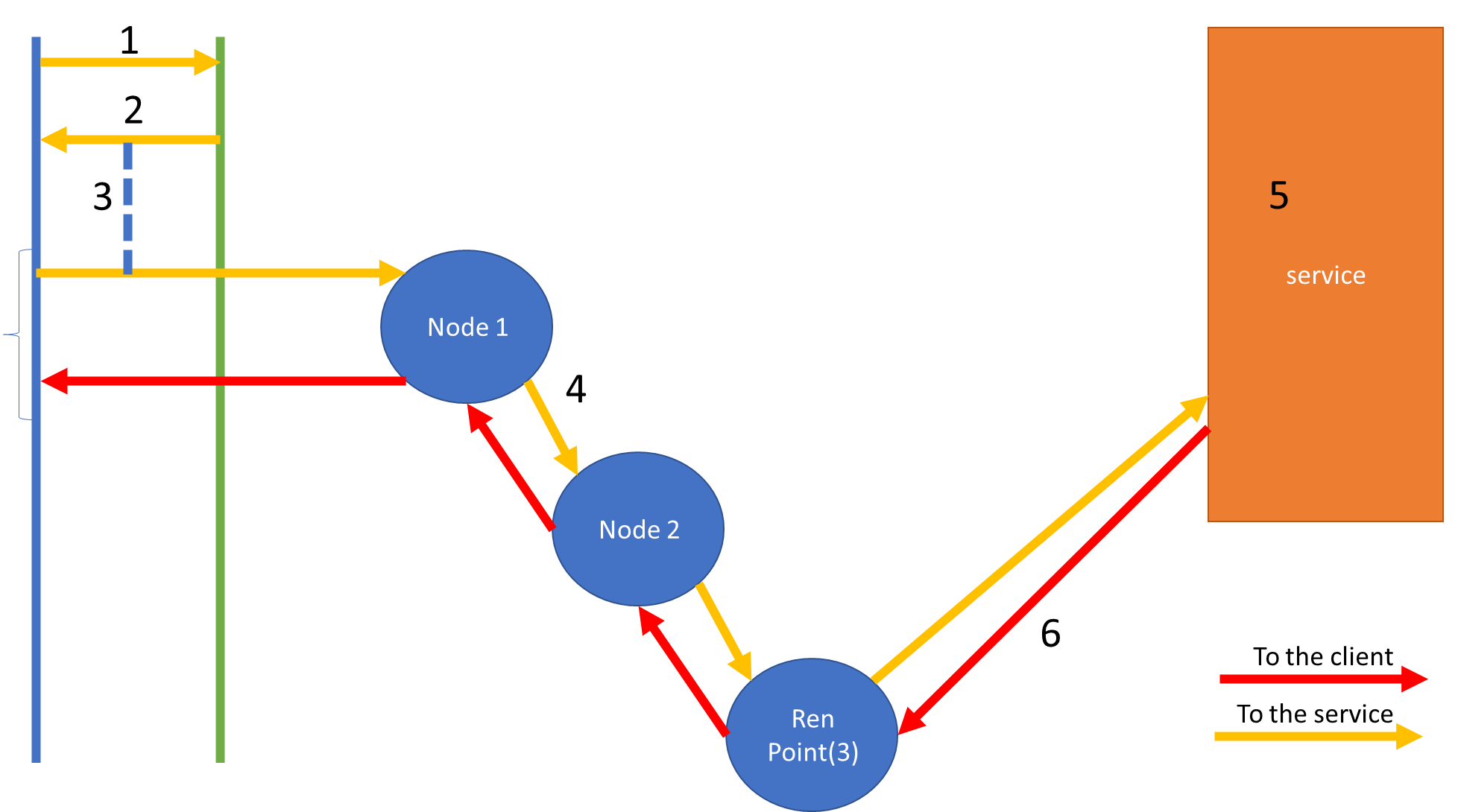
* Operation System at least windows 7
* At least MTU which is higher than 1400 – Maximum onion packet size!
  1. Outgoing and incoming interfaces

None

* 1. Reference to Security

The project's reference to Security is the implementation of the Onion Routing Scheme. This scheme enables to send data via the internet, whereas nodes in the routing path have no access to the sent data, in contrast to the regular routing scheme, as further explained in Theory Chapter.

* 1. Use cases

1. User requests to send a message to the service

Steps explained:

1. The client asks the directory server for routing path to the requested service.
2. The directory server builds and sends the client's routing path in the following order: Node1, Node2, Ren (Node 3).
3. The client builds the onion packets it "wants" to send to the service – first he sends his public key and then the data itself.
4. Activate the onion routing to the service.
5. The service receives the data.
6. The service sends back its replies to the received data.

## User interface:

ADMIN VIEW:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטיAs my project is an anonymous platform, the main Graphical interface is the dashboard. The services that use this platform host they own GUI, and considered separately.

Not very nice…

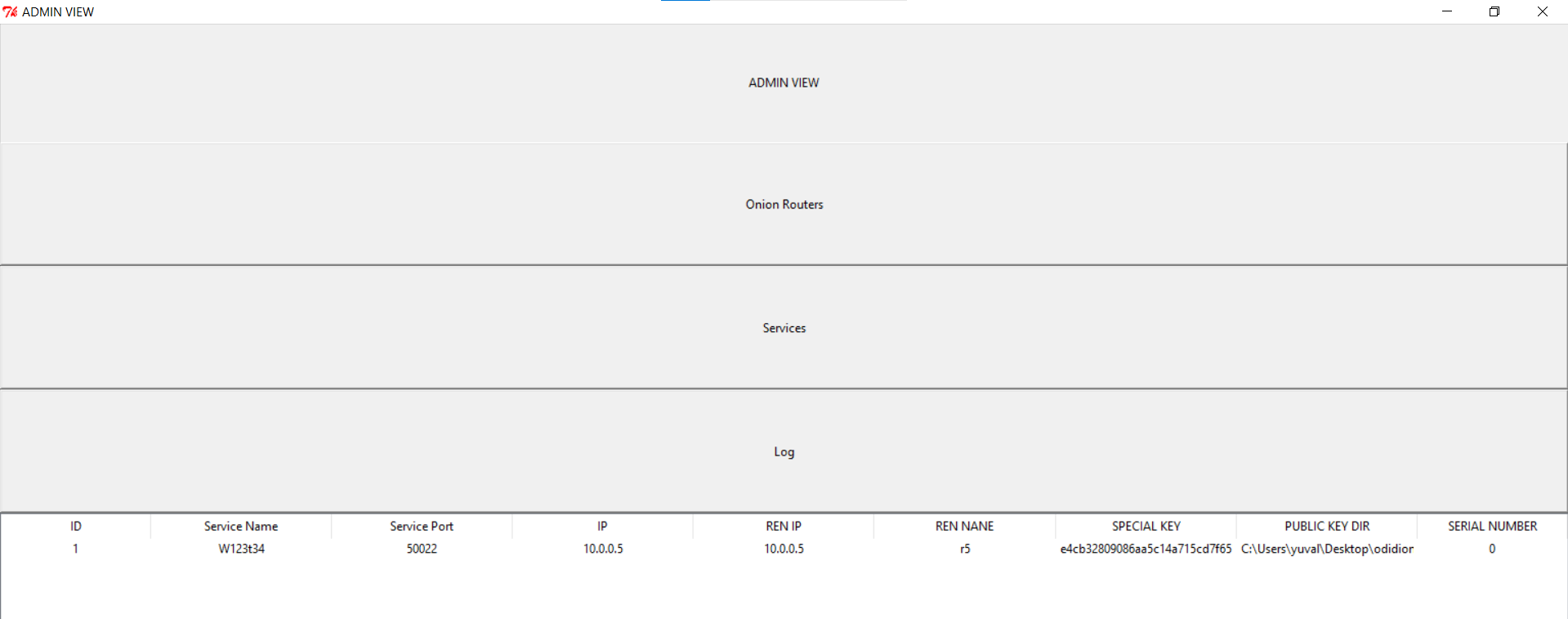
You will never understand my design attitude – Only one person have access to this interface and it's me so I know the best what good for me 😊

1. תמונה שמכילה שולחן

   התיאור נוצר באופן אוטומטיRouter View:
2. Log View:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

1. Services Condition/ Dashboard:

Missing an examples of platform services, and the output #the services are in console design

# 4 The project-writing process

## The process

In the beginning, the project was started with more planning than coding. Besides the roadmap, I had to plan and design the network and the databases structures of all network elements (directory server, onion routers, services, and clients). Besides the onion routing scheme, I wanted to stick to the following principles such as anonymity and privacy across my network.

After two or three weeks of planning, deliberation, and consultations with my Cyber teacher about the network structures, the cryptographic methods, I started to develop the network elements in the following order: Directory Server, Onion Routers, Services, Client.

Along the developing period, I studied a lot of new things such as JSON formatting, RSA, AES, Scapy and improved my coding as a result. When I finished the developing section, I felt that my platform design was too specific and not generic enough as expected from a platform, which means that I could not develop another service or client without undergoing big changes in the current code, and therefore I decided to create – "Odidion Tools". The goal of this mini-project was to create and make my platform extensible and easy to maintain by providing 2 modules - one for client development and the second for service development, allowing the formation of new services on top of my platform.

When the "Odidion Tools" project was done, I integrated all its portions to form a client-server platform, capable to host new user-defined services

## Challenges and different implementation options

Through this project I coped with two main challenges:

* Duplex communication – the main goal of the onion routing scheme is to anonymize the identity of the sender; therefore, it took me a lot of time to think about an implementation which keeps the anonymity and privacy of the sender and enables the network elements to send service replies to the client, with the least saved data about the sender. I came up with the following solution: id\_key!

The idea was to create a one-of-a-kind and non-reversible value for each client in my platform. The id\_key value would be the output of the Hash function on the client's private key, which makes this id\_key value almost 100% unique. This key enables duplex communication because the onion routers and the services can save details about the previous node without knowing the real identity of the sender.

* The onion packets size – because of the multiple encryptions and the additional data that is added to packets in order to optimize the routing scheme, I faced an unexpected challenge - the packets were not sent from the client, after a lot of research, reading and consultations with my classmates, I discovered the reason to the problem, Scapy sends UDP packets only if their size is less than 1,500. Accordingly, I decided to send data in partitions as needed and generate an onion packet for each one of them.

# 5 Solution Components

## Project disciplines

The project focuses on the following fields:

* Communication – the communication between the elements in the onion routing scheme is mainly based on UDP protocol. Besides, the communication between the Directory Server to its clients is based on TCP protocol.
* Encryption – the encryptions in the onion routing scheme are based on asymmetry (RSA) and symmetry (AES) encryption methods.
* Text Protocols – JSON format
* DATABASE – the data in the project is saved in an SQL database, dictionaries, and tuples.
* Coding Standards – Client-Server architecture and OPP (class –base) implementation
  1. Environmental requirements
* Python 2.7
* The following modules
  + Scapy
  + JSON
  + pycryptodome and fernet modules

## Topological view

Where is ADMIN #THERE IS NO ADMIN

Some explanation on the diagram #IN THE DIAGRAM

## 

## Data structure definition

**The main data structures of each Onion Router:**

Dictionary –SERVICES

Role: This dictionary contains the details about the represented service of the onion router

Structure:

Service\_serial\_num : {

"service\_name",

"service\_ip",

"service\_port",

"service\_communication\_type"

}

Dictionary –ROUTING\_PROCESSES

Role: This dictionary contains the IP, port of the senders of the packet in order to enable sending back sessions, the senders' details correspond to the id\_key which was documented in the onion packet.

Structure: Id\_key : (sender\_ip, sender\_port)

**The main data structures of the services**

Dictionary – Send\_back\_id

Role: This dictionary contains the IP, port of the senders of the packet in order to enable sending back sessions, the senders' details correspond to the id\_key which was documented in the onion packet.

Structure: Id\_key : (sender\_ip, sender\_port)

Dictionary – Messages

Role: This dictionary contains all of the received messages of the clients, because of the fact that the received data may be split because of the size of the packets, the service saves in the dictionary, and in the end, all the parts are connected.

Structure: Id\_key : { "serial\_number",

"data"

}

**The main data structures of the clients**

Dictionary - Communication\_details:

Role: This dictionary contains data about the routing path.

Structure:

{

"routers": {'1,2,3': 'ip', 'port' , 'public\_key'},

"communication\_type",

"serial\_number",

"service\_public\_key"

}

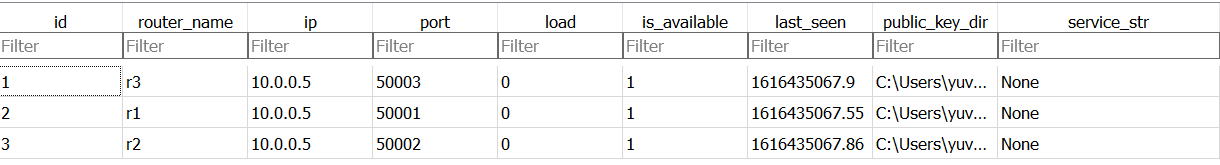
## Database definition

**The main database structures in the project:**

* Onion Routers (internal to the directory server)

The following database contains data about the nodes in the networks such as their IP, Port, amount of services it represents, if they are online or not.

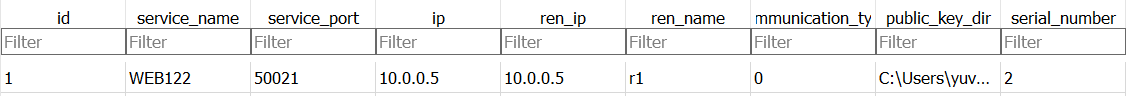
This database helps the directory server to maintain the network by 'building the routing path' to the clients, and choosing which node will represent service in the network.

The database:

* Services (internal to the directory server):

The following database contains data about each service that is connected to the service, such as the node which represents the service, IP, Port, service name.

This database acts as a DNS database to the directory server when the client asks for a routing path for a specific service.

The database:

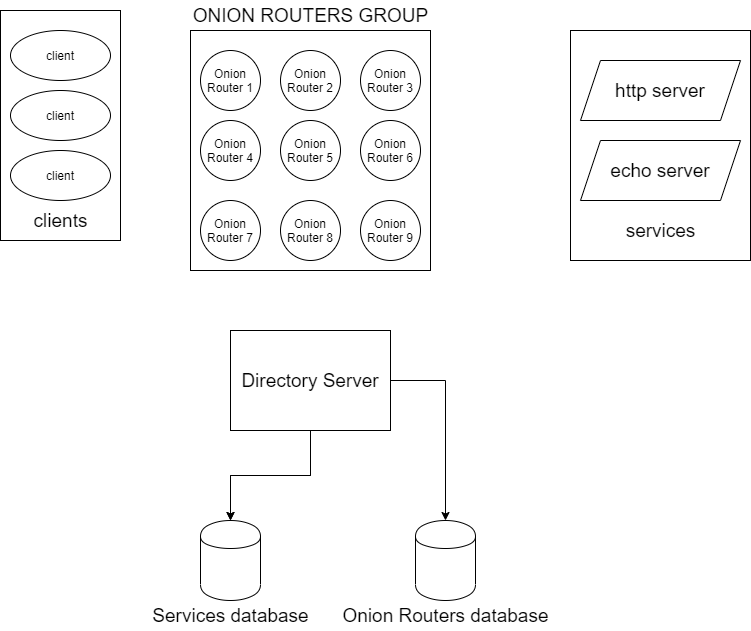
* Services (internal to the Onion Routers)

The following database contains data about the services that the onion router represents.

תמונה שמכילה שולחן

התיאור נוצר באופן אוטומטיThe database:

## Modular view

The project comprises of 3 entities:

**Directory server:**

* Client functions: Building and sending the routing paths for each client who wants to connect to service through the network.
* Onion Routers functions: Handling keep-alive sessions with each one of them, and defines which onion router will represent the services.
* Services function: Registering new services to the network.

**Onion Router:**

* Onion Routing functions: Decrypting the received onion packets and sends them to the next node, back to the client, and direct to the services – as documented in the packet.
* Keep-alive: Handling the keep-alive session with the directory server by updating it about its condition.

**Client:**

* Onion Routing functions: Creating the onion packet according to the routing path which was received by the directory server.

## Main modules

All Odidion's network elements' modules:

onion\_encryption\_decryption:

this module is a collection of all the cryptographic methods which are required to implement the routing through the Odidion network(Onion Routing).

|  |  |  |
| --- | --- | --- |
| Function | I/O | Description |
| generate\_keys | Input: the path of the folder that will contain the key file(.pem), name of the key(the name of the element by default).  Output: the function returns a tuple of the public and the private keys. | This function role is generating the network elements RSA keys. |
| RSA\_Encryption/ Decryption | Input: encrypted/ raw object, RSA key – public/ private (according to the chosen one)  Output: the function returns the encrypted/ raw object(according to the chosen one) | This function role is implementing the RSA encryption/ decryption schemes. |
| Sym\_Encryption/ Decryption | Input: encrypted/ raw object, symmetric key – public/ private (according to the chosen one)  Output: the function returns the encrypted/ raw object(according to the chosen one) | This function role is implementing the RSA encryption/ decryption schemes. |
| Encrypt\_pkt | Input: symmetric key, private key, communication type, and the raw packet.  Output: the function returns an encrypted packet according to the onion routing format | This function role is implementing the Onion Routing encryption scheme. |
| Generate\_sym\_key | Input: None  Output: the function returns a symmetric encryption key. | This function role is generating the network elements AES keys. |

Client main modules:

|  |  |  |
| --- | --- | --- |
| Function | I/O | Description |
| Generate\_packet | Input: nodes details, data, id\_key, public keys…  Output: the function returns a part of the onion packet – 'one layer of encryption' | This function role is generating packets according to the routers |
| Create\_onion\_packet | Input: nodes details, data, id\_key, public keys…  Output: the function returns an onion packet. | This function role is generating a full onion packet by using the 'Generate\_packet' function. |

Odidion Tools:

|  |  |  |
| --- | --- | --- |
| Function (client class) | I/O | Description |
| Ask\_to\_service | Input: service name, Boolean flag of – is web service.  Output: the function returns the routing path from the directory server. | This function role is handling the communication between the client to the dir server about the routing path. |
| Send | Input: sent data  Output: the function returns noting | This function role is to handle the communication with the service through the network  If web service -> the browser handles the communication. |
| Session | Input: nothing  Output: the function iterates the received data from the service | This function role is to iterate the received data from the service  If web service -> the browser receives the data. |

|  |  |  |
| --- | --- | --- |
| Function (service class) | I/O | Description |
| Bind\_and\_set\_service | Input: the name, IP, and port of the service  Output: the function returns if the operation succeeded or not | This function role is to bind and create all the relevant sockets of the service |
| Register | Input: nothing  Output: the function iterates the received data from the service | This function role is to handle the registration of the service to the network by communicating with the directory server |
| Send\_to\_client | Input id\_key, data  Output: None | This function role is to handle the onion routing to the client. |
| recieve\_from\_clients | Input: nothing  Output: the function iterates the received data from the client and its id\_key | This function role is to iterate the received data from the clients |

# 6 Testing Scenarios

## Testing emphases

* Fast communication.
* Well half-duplex communication (request - response).
* Parallel sessions through the network.
* Keep alive – the directory server must know which of the onion routers are online.
* Big data and web support.

## Main Testing scenarios

* 6 online onion routers.
* 2 services that are registered to the network – FTP server, website.
* 2 clients who are communicating with one of those 2 services.

# 7 Reflection

## Time table

|  |  |  |
| --- | --- | --- |
| Month | Implementation | Research |
| December | * Architecture optimization:   + JSON support * Bug and issues fix according to the POC   + Optimize the encryption/decryption methods * Transferring "Big data" support * Generic "Ren Point" | * Directory server roles * Encryption of the databases * TCP/UDP * Anonymity:   + Acknowledges   + Duplex communication * Scapy limitations |
| January | * Duplex communication:   + Acknowledges   + start of developing duplex communication * Client-Server module (API for developers) * Verify performance (zero sleep) * Graphics   + Connect and send manually messages   + Show acks on the messages | * HTTPS, HTTP * Duplex communication * Find the balance between the anonymity to the duplex communication * Client-server support |
| February | * Multi-user Platform allowing multiple send/receive activities * Upgrade client graphic * Basic Admin graphic * Client-server module | * Client-server support |
| March | * Developing example services such as websites, chat servers, cloud services * Web support - graphic * Full EMS support * Loading an exist network (directory servers, onion routers, services) | * Automated testing |
| April | * Complete gaps if necessary * Optimization and upgrades * Bug fixes |  |
| May | * Bug fixes |  |

## Challenges and personal contribution

The process of designing, programming, reviewing, and fixing my project was a new challenge for me, it was my first time working on a long-term project and besides, through my project, I had to learn and research about a lot of technologies, theories, and modules.

Most of my challenges were the complex implementations of my project such as: building the onion packet, half-duplex communication. I had no problems with the long-term scheduling because I work hard to make sure I am not missing any deadline, and when I understood that I am going to miss it, I asked to postpone the deadline to another date

## Insights

As I wrote in the introduction, I chose this project because I was really curious about the implementation of deep-web communication because of its benefits, so this curiosity helped me stay focused and disciplined all the way to the goal – develop my deep-web network. I learned a lot about cryptography, Scapy, web framework -flask, python, and also about myself.

What did you learn about your self?

What about writing in English?

# 8 Instructions for installation and operation

## Configuration and prerequisites

There are no special configuration and prerequisites yet

## Installation

* Scapy
* Install JSON module
* Install pycryptodome and fernet modules

# 9 Bibliography

Missing

Tor technology - <https://gitlab.torproject.org/tpo/team/-/wikis/home>

THERE WERE NO MUCH SOURCES EXCEPT TO THE OFFICIAL TOR PROJECT WEBSITE

# 10 Appendix