Shallot routing

---

1 Goal

By groups of 4, you will be in charge of coding a TOR-like network. The project will have to be finished by December 22nd, 12:00. The grade obtained for the project will count for 25% of the final grade for the course. In addition to the source code, you will make a report explaining your solution. The choice for the programming language is left to you, but we highly recommend to use Python (version 3).

You will implement 2 peers (Alice and Bob) and a set of intermediary relays (R1, R2, R3, etc...). The messages flowing from Alice to Bob will make use of these relays in a random way and will cipher the communication in order to provide content protection and anonymity. The Fig. 1 presents a generic shallot network made of Alice, Bob, and 6 relays.

The project is divided in 4 steps :

1. implement the relays
2. randomize the routing
3. secure the communications on each link by building a shallot
4. relaying the messages

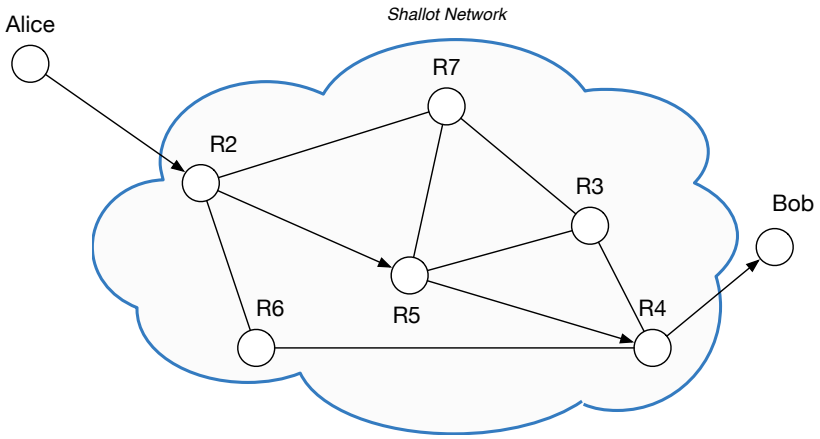


Figure 1: Shallot architecture

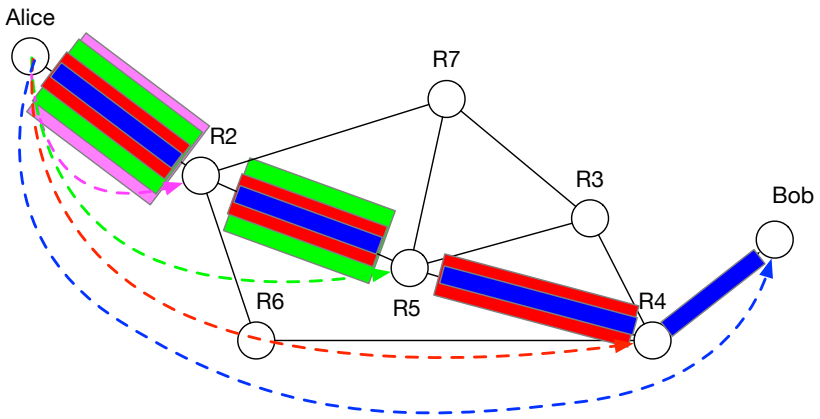


Figure 2: Shallot security

2 Shallot routing in a nutshell

A anticipated in Fig. 1, the *shallot network* uses several relays to route a message from Alice to Bob. The path is randomized in order to minimize the risks of interception and to increase the anonymity. The confidentiality of the *shallot network* is implemented as folllows: the relay nodes are unable to decrypt the original message. They can only know about the next hop in the route and the (ciphered) message they have to relay. Alice picks up the route so that it can never be predicted by the relays.

In practice, the security model is presented in Fig. 2. It can be observed that Alice wraps its original message with multiple encryption layers. All these layers form the *shallot*. At each relay, a layer of the shallot is peeled by using a pre-defined ciphering key. It allows to reveal the next hop and the message to forward. Note that this peeled message cannot be read by the relay itself: it has been ciphered with the key of the next hop. For instance, Alice will wrap a message with the blue key (Bob), then the red key (R4), then the green key (R5), and finally the magenta key (R2). The so-constructed shallot is sent to R2. R2 uses the magenta key to reveal the next hop and the green-ciphered message. It cannot it (only R5 can) and forwards it to R5. Ultimately, the message reaches Bob. It uses the blue key and reveals the original message.

3 Description of the Steps

3.1 Step 1 – implement the relays

Relays –All entities need to read the configuration file config/host.ini to know their IP address and port number where they can be contacted. Next, Alice will load the topology in the file `config/topology.ini`. The entities will listen for incoming requests (on their IP address and port number):

- If the request is malformed (invalid message format), they will send back an `ERROR` (see Appendix) message with the error code `INVALID_MESSAGE_FORMAT`.
- If the request is `KEY_INIT` or `KEY_REPLY` , the relay will negotiate a session key with the contacting peer using the Diffie-Hellmann algorithm.

- If the request is `RELAY_MESSAGE`, it deciphers the message and relays it to the next hop.

Alice – Before any communication can take place, Alice will be in charge of (1) preparing a route to the destination, (2) negotiate all keys with the relays, and (3) construct the shallot, layer by layer.

3.2 Step 2 – randomized routing

The topology is known by Alice: it is contained in the file `config/topology.ini`. The file enumerates (1) a list of relays and the TCP port to contact them and, (2) for each relay, what is its list of neighbours.

In order to select the routing path from Alice to Bob, Alice will make use of a modified Dijkstra algorithm. After loading the topology, the cost of each link will be randomly selected between 1 and 16. Next, Dijkstra's algorithm is used to compute the least cost path. The routing path contains the ordered list of relays to be used in that session.

A packet will always contain the next hop to be used and the payload.

3.3 Step 3 – secure the communications

The communication between Alice and Bob is secured end-to-end and at each relay. In order to do so, Alice will negotiate a symetric key with each relay node. This is done by means of the Diffie-Helman algorithm. Each key is uniquely identified by a unique, 32-bits key ID selected by Alice.

3.4 Step 3 – build the shallot

The path to Bob is considered in reverse order. For instance, in Fig. 2, the reverse path is {Bob, R4,R5,R2}. With each relay R of the path, the following step is performed to build the shallot:

1. The current shallot becomes the payload of this message.
2. The next hop and the key ID are added to the header of the message.
3. The new message is encrypted using the AES algorithm and the key negotiated by Alice and R.
4. These steps are repeated for each relay in the reverse path.

Note that if the next hop is the host itself, it means that the message has reached its destination (e.g., the first message has Bob has next hop, uses Bob key).

For instance, let us agree that the select path from Alice to Bob is Alice, R2, R5, R4, and Bob. The first message M1 will be constructed as follows:

- **Next Hop:** Bob
- **KeyID:** value of the key ID generated by Alice (key: Alice-Bob).
- **payload:** original message from Alice ciphered with Alice-Bob key

The second message M2 is:

- **Next Hop:** R4
- **KeyID:** value of the key ID generated by Alice (key: Alice-R4)
- **payload:** message M1 ciphered with Alice-R4 key

The third message M3 is:

- **Next Hop:** R5
- **KeyID:** value of the key ID generated by Alice (key: Alice-R5)
- **payload:** message M2 (ciphered with Alice-R2 key)

4 Report Content

You report will contain:

- Functional description of your solution.
- One sequence diagram for each step.
- Difficulties encountered and how you solved them.

5 Topology

The `config/topology.ini` file contains the list of all relays and the topology. Each line of the topology contains the IP of a relay and the IP of its neighbours. For instance:

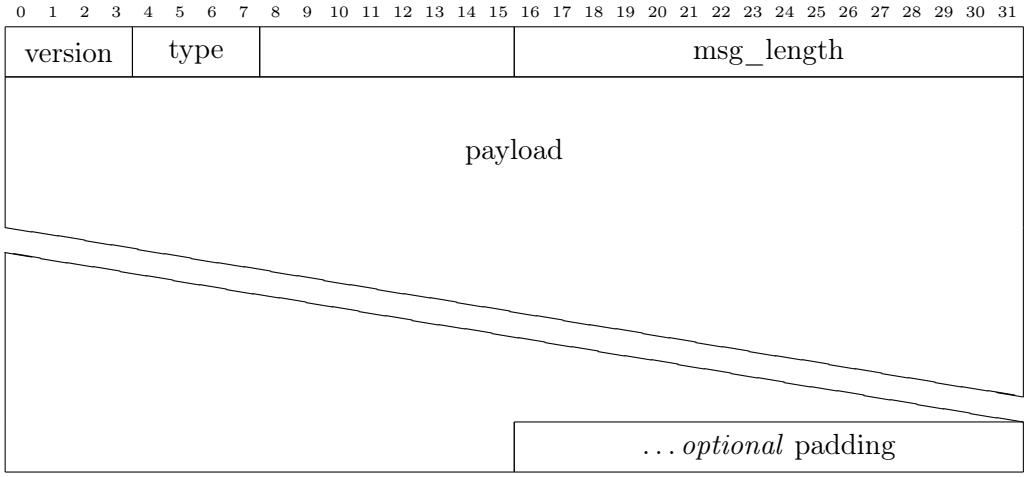
```
[relays]
172.16.1.1 9001
172.16.2.1 9009
172.16.3.2 9012

[topology]
172.16.1.1 172.16.2.1 172.16.3.2
172.16.2.1 172.16.1.1 172.16.3.2
172.16.3.2 172.16.2.1 172.16.1.1
```

The `config/host.ini` file (deployed at each relay) contains its IP and port number:

```
[host]
17216.2.1
9001
```

6 Messages Format

The general format of a packet is comprised of a header of 4-bytes length and a body of variable length.

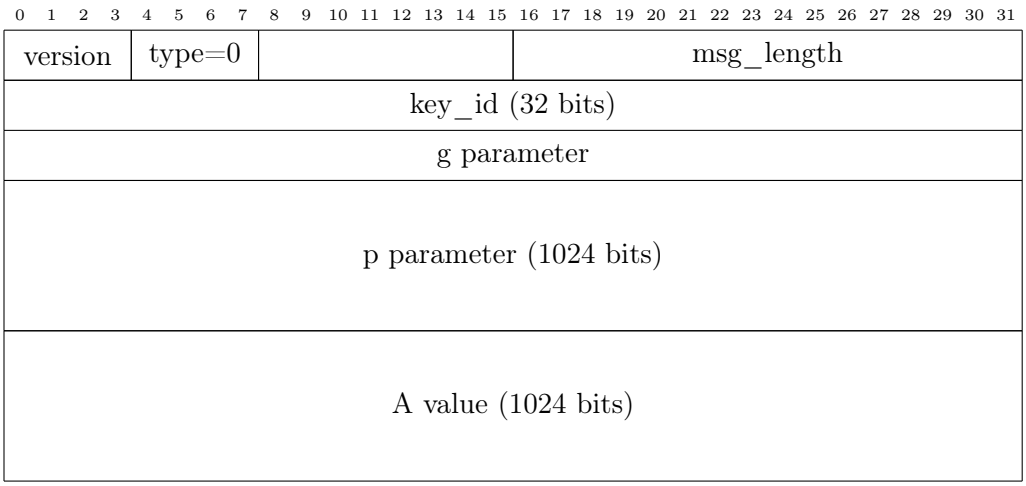


- version must be 1.
- msg_type must be between 0 and 2 (included) depending on the body type.
- msg_length is the total length of the message in dwords (e.g., if the message has a body of 12 bytes, the msg_length will be (4 + 12)/4 = 4). It means that the message length in bytes must be a multiple of 4.

The possible values for msg_type are:

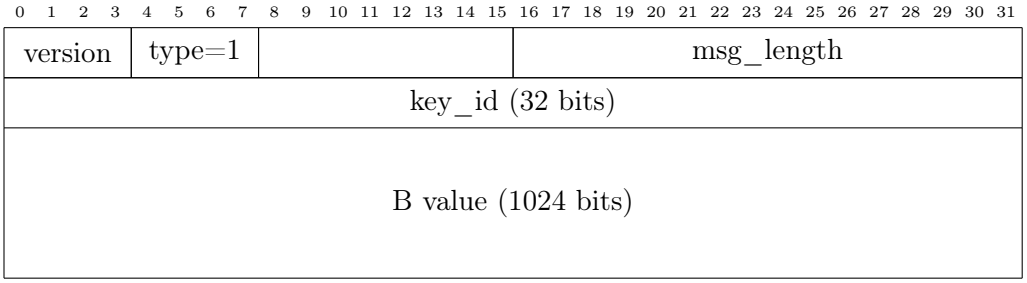
- 0 = KEY_INIT
- 1 = KEY_REPLY
- 2 = MESSAGE_RELAY

6.1 KEY_INIT



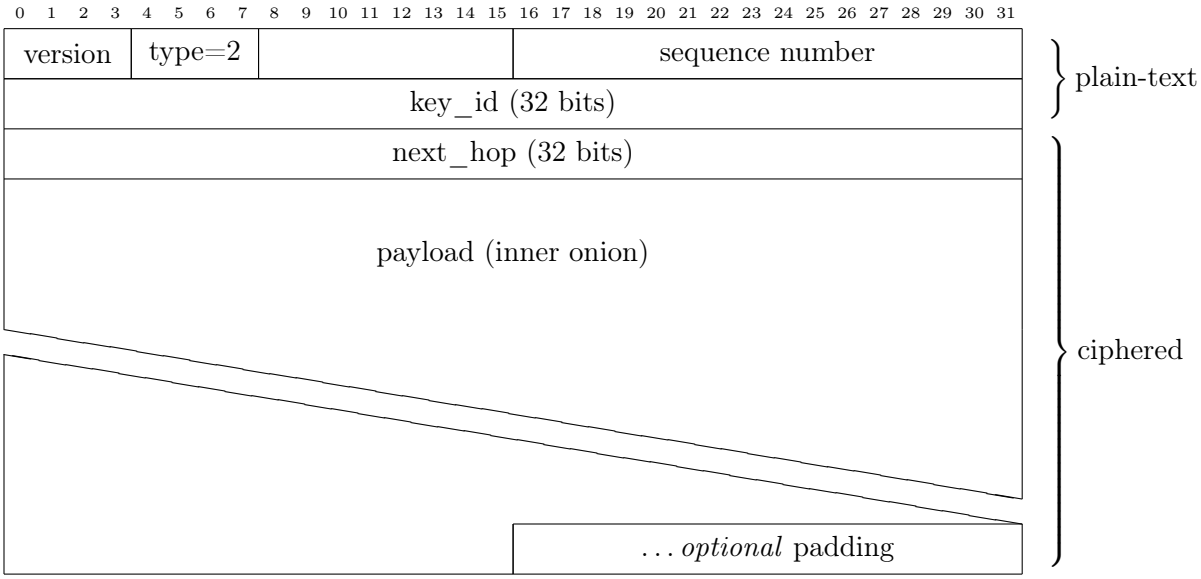
- key_id is a 32-bits number generated by Alice.
- g,p, A: the Diffie-Hellman parameters sent to the relay. In this project, we will set $g = 2$ and p of size 1024 bits

6.2 KEY_REPLY



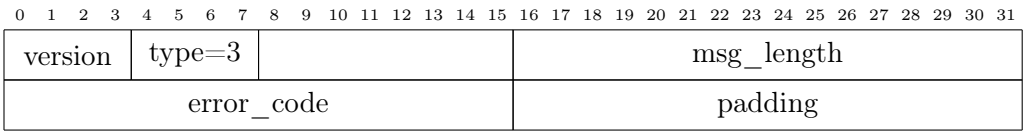
- key_id is a 32-bits number generated by Alice.
- B: the Diffie-Hellman parameters sent to Alice.

6.3 MESSAGE_RELAY



- key_id the ID of the key to decrypt the payload.
- next_hop: IP address of the next relay. If the next_hop is this host, the message has reached its destination (Bob)
- payload: the content of the message to be forwarded to the next_hop.

6.4 ERROR



The error codes are:

- 0 = `INVALID_MESSAGE_FORMAT`
- 1 = `INVALID_KEY_ID`