Design and security of an instant messaging service using Tor

Marvin FOURASTIE

Spring 2021

1 Introduction

Tor [1] is a communication service based on the Onion Routing principle which provide anonymity to the users. Tor works on the Internet, which allow any user to run TCP-based applications without reveling his real IP address.
This anonymity is guaranteed by the design of the service which works as follow: each client choose a path through the network and build a circuit with nodes, called "onion router"(or "OR"), that know only its predecessor and successor, but no other nodes in the circuit [1].
As now most of the chat applications want to provide secure and private communications to their users, we are interested in the implementation of an instant messaging system on the Tor network. The main motivation is to exploit Tor design to keep the users anonymous on the network and get rid of a centralized server. Furthermore, we want to emphasize the pros and the cons of such an implementation by pointing out what are the security features provided and the attacks possible.
Through this paper, we will first have a look at the existing applications of instant messaging to highlight the differences and similarities between "classical" messaging systems and the ones running on Tor. Then, we will present a small proof of concept to understand better the design of such a service. And to finish we will see which security features this type of application provides and what are the possible attacks.

2 Background

In order to have a better understanding of Tor, we introduce some definitions of the main concepts of Tor that are used to build a instant messaging system. The following explanations are highly inspired by the ones given in [2].

- **Onion Proxy (OP)**: This is a small piece of local software that needs to be installed on the user's device. It enables communication with the directory servers (DSs), establishes connections in the Tor network, and handles connections from the user's applications.

- **Directory Server (DS)**: These are a small set of trusted and known servers in the network that actively keep details about the complete network status. DSs produce a consensus document with the status of the network relays, bandwidth availability, exit policies, etc. The OPs can download this document from a DS and select three suitable relays to establish the communication circuit to a destination.
- **Entry Node/Guard**: This is the relay in the Tor network that the client directly connects to and hence, it knows the identity of the client.
- **Exit Node**: This is the final hop of the Tor circuit. Therefore, it knows the IP address of the destination server accessed via the Tor network.
- **Hidden services (HS)**: This is a web server that can be hosted in a node inside the Tor network or outside of the Tor network. These have a top-level domain name called .onion.
- **Introduction Points**: These are random nodes selected by the HS at the start of the connection establishment process. Once the HS has selected an introduction point, it provides the HS's public key to the introduction point.
- **Rendezvous Points (RPs)**: This is a random Tor node selected by the client OP before the client initialises a connection with any of the introduction points advertised by the DS. The client establishes a circuit to the RP and informs the HS to meet at the RP. The HS then creates a connection to the RP.

3 Related work

Nowadays a lot of social network applications provides its own instant messaging service. Moreover, some chat applications that use Tor already exist even if they are less popular. In order to provide an overview of these works, we separate them in three categories.

3.1 "Classical" instant messaging

One of the common point of the most popular chat applications like WhatsApp, Signal or Telegram is that they use end-to-end encryption. The protocol used by the first two is the Signal protocol [3] that provide perfect forward secrecy by using a different key for each message. In Signal, like in most of the protocols used by instant messaging applications, all the messages are sent to a server (which potentially can collect metadata). However, these chat applications are considered secure, because the messages are encrypted by a shared key. But they does not provide anonymity.

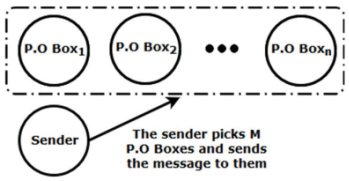
3.2 Synchronous messaging over Tor

As already mentioned earlier, the use of Tor provides anonymity to the users on the network. One of the first application of instant messaging over Tor is TorChat [4] which aims to provide a chat service between two users without revealing any information about the users. This is possible by the use of hidden services (or "onion services") which allow a user to create a TCP service without revealing IP address.
Ricochet [5] uses Tor and end-to-end encryption to make the chat secure and anonymous. Furthermore, the use of Tor permits to have an autonomous application as the routing is ensured by the Tor network. Ricochet is a real-time messaging system that follows these properties [5]:

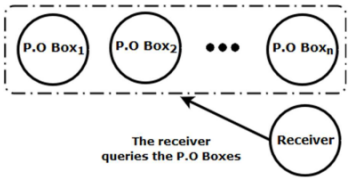
- Users aren't personally identifiable by contacts or their address
- Communication is authenticated and private
- No person or server can access contact lists, message history, or other metadata
- Resist censorship and monitoring at the local network level
- Resist blacklisting or denial of service against users
- Accessible and understandable for non-technical users
- Reliability and interactivity comparable with traditional instant messaging services

Each user is represented by a hidden service as its connection point. We will see later that our proof of concept aims to simulate this kind of system.
Even if TorChat and Ricochet seems to have all the required features for a secure and anonymous chat, the users need to be online at the same time to chat.

3.3 Asynchronous messaging over Tor



(a) The sender sends the messages to receiver's P.O Boxes



(b) The receiver retrieves the messages from P.O Boxes

Figure 1: Asynchronous messaging using P.O Boxes [6]

One of the first attempts to develop an asynchronous and anonymous messaging service was made by Tox [7] which is an encrypted instant messaging

protocol, which provides peer-to-peer communication. Even if Tox does not provide real asynchronous communication, it implements "pseudo-offline" messages, where a message is stored locally at the user, until both are online.
Recently, a new protocol named ATHiCC [6] (for Asynchronous Tor Hidden Chat Communication) was proposed in 2019 to provide an anonymous, asynchronous and serverless instant messaging protocol. The main idea is to send messages into "P.O boxes" (see Figure 1) which allow the receiver to collect the messages at any time the P.O box is online. The first step is to associate to several P.O Box. The P.O Boxes are node that agrees to provide message hosting service. Once this is done, the receiver node informs his contacts of the P.O Boxes he is using. Then, the sender and the receiver can chat asynchronously using these P.O Boxes.

4 Proof of concept

To illustrate the feasibility of an instant messaging system on Tor, we realized a simple proof of concept which show the construction of a circuit initialized by two clients, and how they build their hidden services to communicate with each other. The goal here was not to offer an exhaustive design of a real Tor network but to extract the most important features which ensure anonymity of each user. This work will also allow us to illustrate the security issues and potential solutions of such kind of system.

4.1 Design and implementation

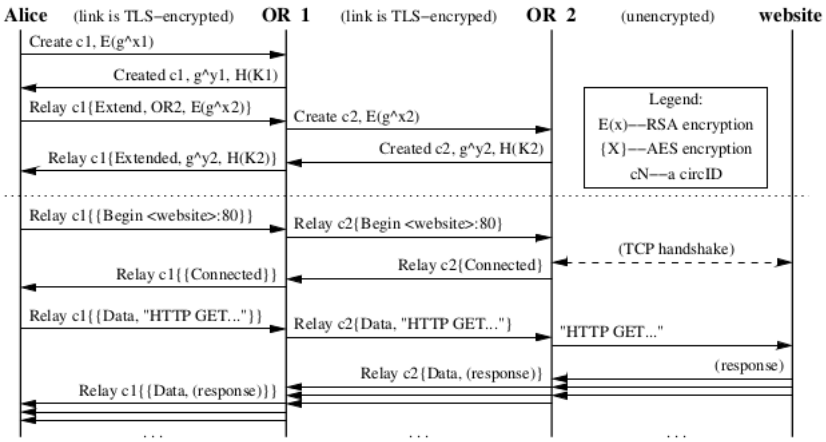


Figure 2: Alice builds a two-hop circuit to a server using Tor [1]

The design of our proof of concept is inspired by the figure 2. We used Python sockets and multi-threading to simulate the communications between

nodes. The nodes of Tor communicate using cells [1] which contains a command and a payload. Here, we chose to implement two types of cells (actually used by Tor but simplified in our proof of concept):

- **"create" cell**: set up the circuit by negotiating a symmetric key with an OR of the circuit. In our work, if an OP sends a cell "create" to OR1, they will do a Diffie-Helman handshake to build a shared key.
- **"extend" cell**: this cell has for goal to make the last node of the circuit extend by one hop the current circuit. The last node send a "create" cell to the next OR to complete the circuit.

Now, let us have a look at the nodes of the network. We know that a client who want to connect to Tor network needs to have an OP in order to initialize a circuit. This OP negotiates a symmetric key with each OR on the circuit. We choose to implement this OP like a particular OR who stores all the symmetric keys of each OR. The ORs of the circuit are able to send cells and share only a key with the OP.
As we said earlier, the main goal of building an application on Tor is anonymity. We will now show that our proof of concept preserves anonymity for each nodes of the network. First of all, the OP (call her Alice) chooses a path and sends a "create" cell to the first OR (call him OR1). By receiving a "create" cell, OR1 will share a key only known by Alice and him (call it K_1). To extend the circuit by one hop, Alice will send an "extend" cell encrypted with K_1. OR1 can decrypt it and then send a "create" cell containing the half-handshake of Alice to the next OR (called OR2) to establish a shared key K_2 between Alice and OR2. Now, Alice wants to connect to an hidden service. She will send another "extend" cell (in the real Tor, she has to send "relay begin" cell) but as she wants OR2 to decrypt the cell, she will encrypt the cell two times: one time with K_2 and another time with K_1. The successive encryptions and decryptions of the cells works as the layers of an onion. When OR1 will receive the cell, it will remove the first "layer" by decrypting with K_1 and then forward the cell. Then OR2 will remove the last "layer" using K_2 to read the cell. As we connect to a server, he will establish a TCP connection and will become the exit node of the circuit.
We can notice that at each time, Alice knows she is handshaking with the ORs but the ORs do not care who is opening the circuit. So, Alice remains anonymous to every nodes of the network. Furthermore, each OR knows only its predecessor and successor as the cells flow node by node in the circuit.
For now we can build a circuit from a client using an OP to a hidden service established by the client. As the goal of our project is to show the feasibility of the implementation of a instant messaging system on Tor, we want two clients (Alice and Bob) to be able to communicate with each other. The idea we want to illustrate here is to allow Alice and Bob to communicate through their hidden services. To implement it, we made two scripts which build a hidden service for Alice and another one for Bob. When the circuits are built, Alice and Bob can chat through their hidden services, preserving their anonymity.

With this proof of concept, we focused on the anonymity between the nodes of the network, omitting some additional features of Tor, in particular the connection between the hidden services to establish a chat session (see 6).

5 Security of the system

Tor is mainly used to remain anonymous on the network, but it exists some attacks to break this anonymity and defenses to some of them. However, in our work we focused on the perspective of an instant messaging system which bring also other security considerations. In this section we will not give an exhaustive list of all the possible attacks on Tor but we will emphasize the strengths and weakness of such an architecture for an instant messaging.

5.1 Defenses

The design of Tor is made to prevent some attacks. Here are some of them [1]:

- **Perfect forward secrecy**: The use of different session keys for each successive hops in the circuit guarantees this property. A single hostile node cannot record traffic and later compromise successive nodes in the circuit and force them to decrypt it.
- **Rendezvous points and hidden service**: Hidden service allows a client to remain anonymous. Indeed, it allows the user to offer a TCP service without revealing his IP address. This anonymity protects against some kinds of DoS attacks because the attackers do not know the client's IP address. Furthermore, Rendezvous point and Introduction points add an extra level of indirection.
- **End-to-end integrity checking**: Integrity checking is essential for a chat service. Tor verify data integrity at several level in the network. At the establishment of the shared keys, the cells are signed using RSA to ensure integrity during the Diffie-Helman handshake. Tor checks also at the edges of each stream, before it leaves the network.

5.2 Attacks

Unfortunately, Tor has security issues that are well-known by the security community. The fact that Tor is used by criminals who can stay anonymous makes governments active to find attacks on Tor, especially de-anonymity [2]. We now will have a look to some attacks that are a real problem for an instant messaging system that claims to be anonymous.

- **Entry and exit routers**: The entry and exit nodes are one of the weaknesses of Tor. Indeed, a attacker which controls both the entry and the

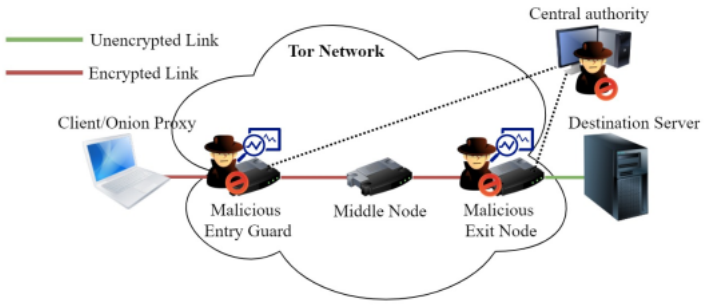


Figure 3: Attack Scenario with compromised Entry and Exit nodes (from [2])

exit nodes of the circuit can break the anonymity of the user. To success, the attacker has to compromise existing Tor nodes or introduce new attacker controlled nodes into the Tor network.

- **OP/Server**: The principle of this attack is to control one Tor node, a Tor client or a server. Even if Tor network has scaled with time, making it difficult to execute attacks by controlling Tor nodes [2], some of them are actually possible. If the OP is used to execute the attack, its default functionality will usually be altered to match the requirements of the attack. If the attack is on the server, it can be accomplished by hosting a server or taking control of a targeted server. These attacks leads to de-anonymise an OP or a HS.
- **Side channels**: The most common type of side-channel attack is the ones which use to intercept the traffic between the Tor client and the entry node.

We can notice in the literacy that the majority of the attacks are on anonymity, because it is the cornerstone of Tor design. The main weaknesses are on the entry and exit nodes because they send and received unencrypted data and so they need to be trusted entities.

6 Further works

If we had to build a large scale usable instant messaging application on Tor, we have to consider some important features for an efficient and secure onion routing. As our proof of concept does not show all the features of Tor, let us list some which missing:

- **Data encryption**: Of course, all the data are not exchanged in clear on the network. Each onion router maintains a TLS connection to every onion router [1]. Furthermore, data are encrypted with RSA keys and are

signed to prevent security issues. As already said, we focused for our work mainly on symmetric shared keys which provide anonymity.

- **Directory server**: In Tor, an OP has to connect first to a directory server to select the nodes of its circuit. Here, as we show an example with two relays we presume already chosen.
- **Rendezvous Points**: In order to preserve anonymity, two clients who want to communicate will use their hidden services. But each hidden service will use intermediates nodes (introduction points) as contact points. If two clients want to communicate after one advertise the other through its introduction point, they have connect to a particular OR (the rendezvous point) that cannot recognize the clients or the data they transmit.
- **Circuit identifier (circID)**: As many circuits can be multiplexed over the single TLS connection, we add an information called circID which specifies which circuit the cell refers to. This feature can be added if we want to make a node handle multiple connections.

The proof of concept is a good basis to add features in order to have a more complete simulation of an instant messaging on Tor. Additionally, we can also make some enhancements to make the simulation more scalable and complete. For example, being able to use it on several machines (works currently on the localhost), make a user interface, make the code suitable for testing... We could also apply some new approaches to improve Tor design like key management as an alternative to directory servers [8].
Here, we focus on synchronous messaging, but it could be interesting to consider asynchronous messaging. Indeed, this last represent a more realistic application of instant messaging. The use of the ATHiCC [6] protocol is a pertinent direction for this project.

7 Conclusion

Nowadays, privacy on the Internet is a preoccupation for a lot of user especially on instant messaging applications. Even if the most popular ones ensure end-to-end encryption of the data, anonymity is not guaranteed by these software. We saw here that Tor can provide such kind of anonymity and provide also integrity and a non centralized server. In a simple proof of concept, we saw that a client cannot be identified on the network thanks to the onion routing design and implementation. Moreover, it already exists some applications like TorChat or Ricochet that use Tor to anonymize messaging
However, Tor has some limitations that make a large scale utilisation complicated. To begin, the anonymity of the users leads unfortunately to abuses that can motivate some governments to potentially block the deployment of these kind of application. Then, as a direct consequence of the first point, Tor suffer of its image of "network of criminals" that is still in most of people mindset. Finally the last limitation is the security issues that leads to the de-anonymization

of the users.
To conclude, Tor is a good option to provide anonymity for an instant messaging system. Our proof of concept can be use as a starting point for testing and developing additional features like asynchronous messaging. Active research are still made on Tor and maybe, one day, it will leads to a larger utilisation of such kind of application ensuring anonymity.

References

[1] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-Generation Onion Router:," tech. rep., Defense Technical Information Center, Fort Belvoir, VA, Jan. 2004.

[2] I. Karunanayake, N. Ahmed, R. Malaney, R. Islam, and S. Jha, "Anonymity with Tor: A Survey on Tor Attacks," *arXiv:2009.13018 [cs]*, Sept. 2020. arXiv: 2009.13018.

[3] J. Alwen, S. Coretti, and Y. Dodis, "The double ratchet: Security notions, proofs, and modularization for the signal protocol," in *Advances in Cryptology – EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings*, pp. 129–158, Springer Verlag, 2019.

[4] B. K, "prof7bit/TorChat," May 2021. original-date: 2012-02-05T16:13:46Z.

[5] "ricochet-im/ricochet," May 2021. original-date: 2010-07-07T23:01:23Z.

[6] D. F. Balchasan, M. Ozaniak, S. Khajuria, Y. Schwartz, N. S. Steffensen, and L. T. Sørensen, "ATHiCC: An Anonymous, Asynchronous, Serverless Instant Messaging Protocol," p. 10.

[7] "TokTok/c-toxcore," May 2021. original-date: 2016-07-06T08:15:22Z.

[8] P. Palmieri and J. Pouwelse, "Key Management for Onion Routing in a True Peer to Peer Setting," in *Advances in Information and Computer Security* (M. Yoshida and K. Mouri, eds.), vol. 8639, pp. 62–71, Cham: Springer International Publishing, 2014. Series Title: Lecture Notes in Computer Science.