HTor: Enabling Unlinkable Communication on the Internet

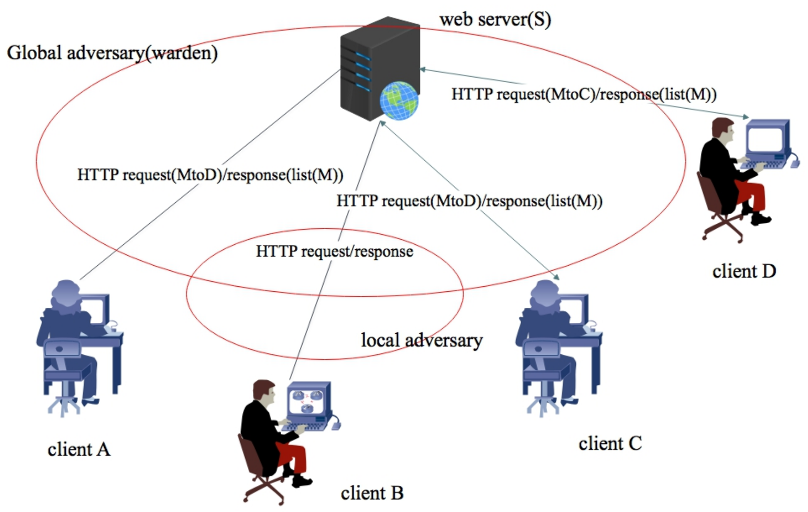
A HTor demo

August 19, 2017

Contents

1 HTor basic structure

Figure 1: HTor basic structure


web server(S)
Global adversary(warden)
HTTP request(MtoC)/response(list(M))
HTTP request(MtoD)/response(list(M))
HTTP request(MtoD)/response(list(M))
HTTP request/response
client D
local adversary
client A
client C
client B

HTor consists of HTor server S and clients. As shown in Fig. 1, clients can be classified into two categories: normal clients and HTor clients. A normal client B is a client who requests S for web pages. On the contrary, a HTor client A intends to send/receive messages covertly by sending requests to S and receiving responses from S.

By the hand of Covert Channels, we modulate HTTP requests/responses containing messages the same with normal HTTP requests/responses. The details are stated in HTor paper and are omitted here for simplicity. As the requests/responses of HTor clients are indistinguishable from those of normal clients, all clients interacting with S is indistinguishable. In this way, we achieve a covert and unlinkable communication through single one web server with negligible overhead on clients. What's more, by designing a robust algorithm over HTTP for constantly updating keys, HTor clients can deny previous communication easily.

2 HTor server in demo

A HTor server in demo is built on amazon EC2 web service. We own a perfect domain name `http://htor.tech/`. The core code for supporting a HTor server is less than 200 lines and, as stated in HTor paper, supporting HTor puts little overhead on a web server. The demo client code is named 'simple_htor.py' in the same directory and the procedures to experiencing HTor is presented later.

3 Set up

Demo client code relies on python 2.7, HTor client code for python 3 is not available yet. We here present how to install needed packages to start HTor client GUI. Instructions for Windows are similar, anyone who helps to give detailed instructions for Windows will be appreciated.

3.1 Ubuntu

Follow instructions below to install a GUI library and 'gmp' library:

```
sudo apt-get install python-tk libgmp3-dev
```

Install newest pip library:

```
sudo apt install python-pip
pip install --upgrade pip
```

Install some required python packages:

```
pip install tornado seccure
sudo pip install BeautifulSoup4 lxml
```

At last, run command below to start chatting using HTor.

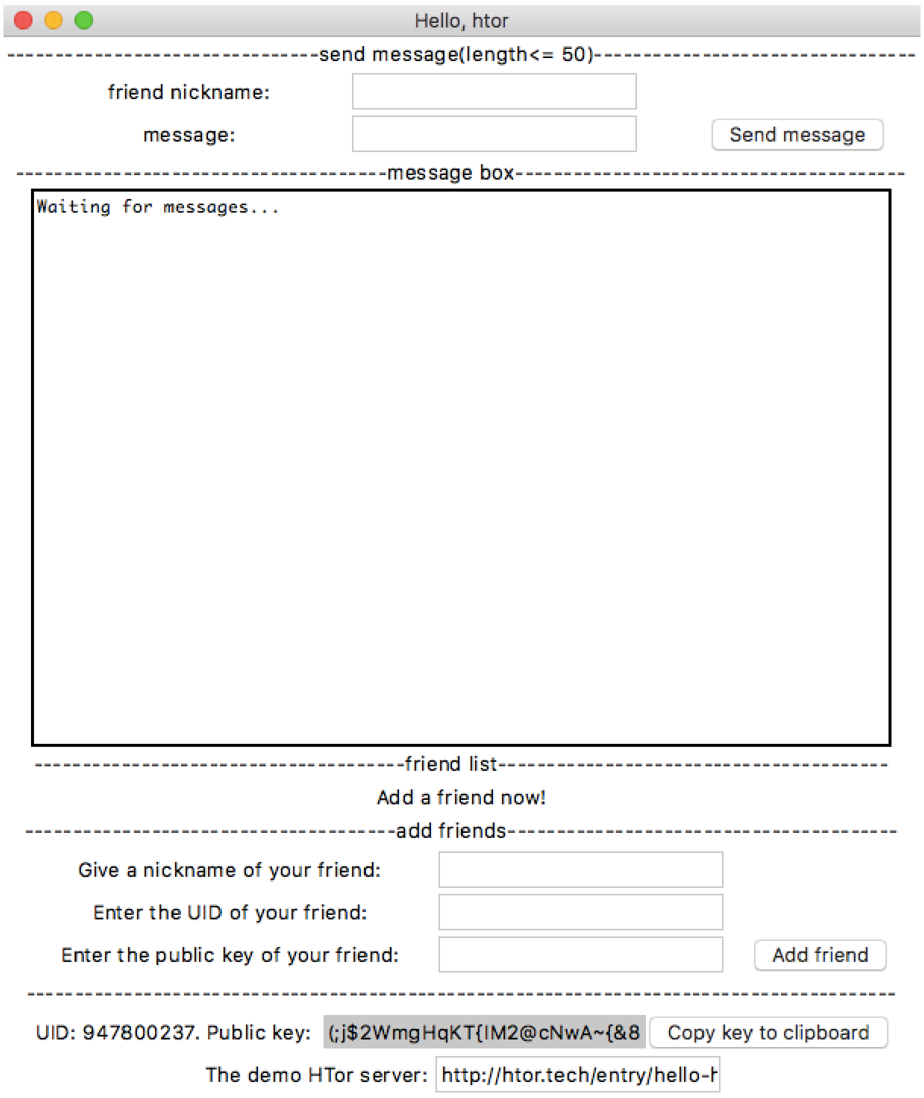
```
python simple_client.py
```

3.2 MAC OS

To be filled.

4 Hello HTor

Figure 2: main interface



Hello, htor

-------------------------------send message(length<= 50)-------------------------------

friend nickname:

message:

Send message

------------------------------------message box------------------------------------

```
Waiting for messages...
```

------------------------------------friend list------------------------------------

Add a friend now!

------------------------------------add friends------------------------------------

Give a nickname of your friend:

Enter the UID of your friend:

Enter the public key of your friend:

Add friend

UID: 947800237. Public key: (;j$2WmgHqKT{IM2@cNwA~{&8 Copy key to clipboard

The demo HTor server: http://htor.tech/entry/hello-

As shown in Fig. 2, the main interface consists of five parts:

- message send.
- message box.
- friend list.
- add friends.
- Your profile: your HTor id and pub.

In the reminder of this doc, we show the function of each part and how to chat step by step.

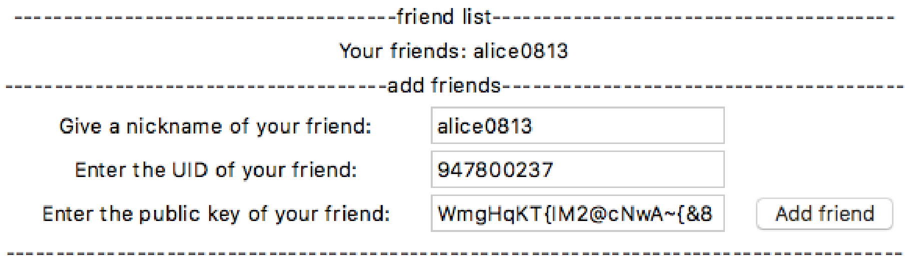
4.1 Your profile

When you run command "python simple_client.py ", the script will: 1) check the server works or not. 2)find a unique id(uid) for you to register in the HTor server. 3)generate a random public key(pub) of fixed length for communicating with friends. Note that each normal client also has an uid. By the hand of covert channels, the differences between normal clients and HTor clients are only accessible to the HTor server.

4.2 add friends

The first step is adding some friends. To add a friend, the uid and public key of your friend are needed to create a quasi end-to-end encryption over HTTP. The uid and public key of a friend is acquired by ways other than HTor.

Figure 3: add friends



```
--------------------------------------friend list---------------------------------------
                              Your friends: alice0813
--------------------------------------add friends---------------------------------------
      Give a nickname of your friend:        alice0813
       Enter the UID of your friend:         947800237
    Enter the public key of your friend:     WmgHqKT{IM2@cNwA~{&8      Add friend
-----------------------------------------------------------------------------------------
```

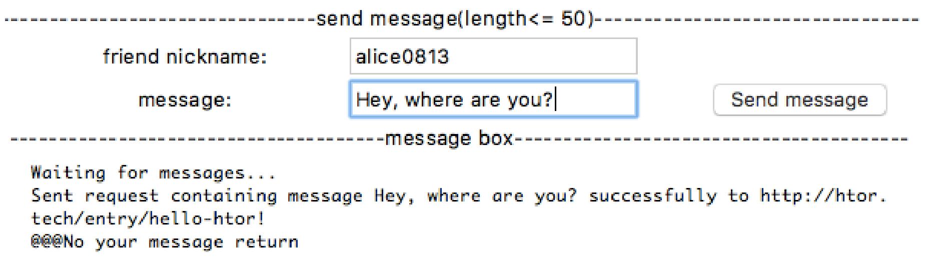
As shown in Fig. 3, except the needed uid and public key, you also need to input a nickname of your friend then HTor scripts can know to who each message sent. By clicking 'Add friend'. HTor will new a profile file for storing and updating keys for encryption between you and the friend.

4.3 friend list

'friend list' part shows all friends added. As shown in Fig. 3, after you add friends, name of all your added friends is shown in the 'friend list' part.

4.4 message send

Figure 4: send messages



-------------------------------send message(length<= 50)-------------------------------
friend nickname:
message:
Send message
-------------------------------------message box---------------------------------------

alice0813

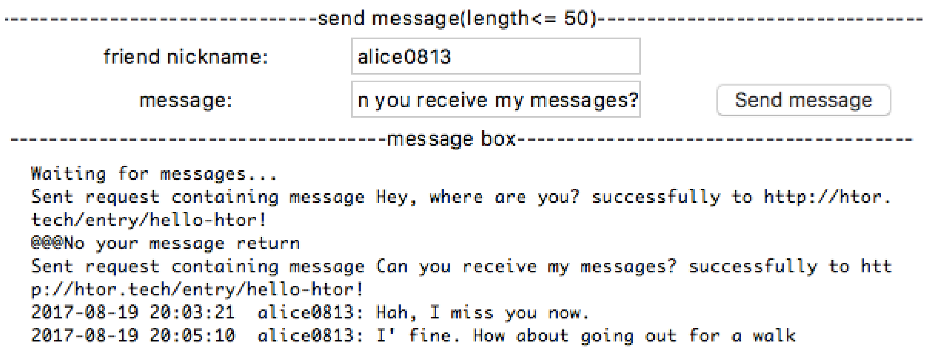
Hey, where are you?

```
Waiting for messages...
Sent request containing message Hey, where are you? successfully to http://htor.
tech/entry/hello-htor!
@@@No your message return
```

To send a message, as shown in Fig. 4. Once the button 'Send message' is clicked, a HTTP request wrapping the message "Hey, where are you?" is sent to the HTor server.

4.5 message box

Figure 5: chatting



-------------------------------send message(length<= 50)-------------------------------
friend nickname:
message:
Send message
-------------------------------------message box---------------------------------------

alice0813

n you receive my messages?

```
Waiting for messages...
Sent request containing message Hey, where are you? successfully to http://htor.
tech/entry/hello-htor!
@@@No your message return
Sent request containing message Can you receive my messages? successfully to htt
p://htor.tech/entry/hello-htor!
2017-08-19 20:03:21  alice0813: Hah, I miss you now.
2017-08-19 20:05:10  alice0813: I' fine. How about going out for a walk
```

After receiving the HTTP response, as shown in Fig. 5, the script will automatically analyze the secret messages in the HTTP response and output these messages to the message box.

5 A unusual communication style

Each HTor client can only receive messages from the HTor server by receiving HTTP responses. What's more, each HTTP response is received after the corresponding HTTP request is sent. Consequently, different from the common style – "I see your messages, then I answer you", the communication style over HTor is "I see what you answer after I send a new message". For example, Bob asked Alice "Hey, how are you?". After that, the HTor server received Alice's encrypted answer ”I’m fine. How about going out for a walk”. However, the HTor server can not inform Bob there is a message for him. Only when Bob send next HTTP request containing ”Can you receive my messages?” can Bob received Alice’s previous reply.

6 Conclusion

HTor presents a novel idea to design secure communication system by exploiting covert channels. This demo implements almost all ideas stated in the HTor paper. HTor is easy to implement and is very flexible as it can utilize various kinds of covert channels. Besides, HTor put negligible overhead on clients and HTor is stable because HTor clients don’t rely on other clients to function. The detailed comparisons between HTor and popular anonymous networks like Tor can be found in the HTor paper.