Home Assignment 2

November 26, 2017

1 A-2) Why is it important to have a large volume of traffic in anonymous communication?

The term anonymity set is important when talking about anonymity. The anonymity set is the set of people that a user is anonymous in. This means that when a user makes an action or sends a message it is not possible to determine a particular user in the anonymity set. This means that if the anonymity set is small then it is more likely go guess right by simply guessing. In the same way it is harder to guess right if the set it large. From this discussion it follows that it is important to have a large anonymity set in order to provide strong anonymity.

In order to illustrate why it is important to have large volume of traffic in anonymous communication lets consider a Chaum Mix. The mix works by first buffer a bunch of incoming messages where the goal is to deliver these messages in an anonymous way to the right recipients. When a large volume of messages have been received by the mix the mix changes the order of the messages when sending out the messages. Since the messages are encrypted and the order is changed it is hard to guess the mapping between senders and receivers. If the buffer is small then it is easier to guess the mapping between senders and receivers than when the buffer is large. This means that the anonymity set increases when the buffer is increased. If the traffic is sparse it means that we will have to wait longer in order to generate a large anonymity set then when there is more traffic. This means that it is important to have large volumes of traffic. The same principle holds for the Tor network where synthetic traffic is generated.

2 A-7) When sending a mail through several Mixes, there are several public keys involved: K_1, K_2, . . . , K_n and K_a. What happens if one does not use K_a? Does this risk the anonymity of the sender?

When using several mixes we send out $K_n(R_n, K_{n-1}(R_{n-1}, ..., K_1(R_1, K_a(R_0, M), A_y)...))$. Lets assume that the public key K_a is not used. This means that instead $K_n(R_n, K_{n-1}(R_{n-1}, ..., K_1(R_1, (R_0, M), A_y)...))$ is sent out. This means that the message exiting the last mix will become $(R_0, M), A_y$ and since M is not encrypted an eavesdropper can see every message sent to each recipient. However

since the sender still encrypts the message with the public keys of the different mixes an eavesdropper can not trace the message sent from an sender. But messages sent by an sender can be read by an eavesdropper so the conclusion is yes in some sense the sender looses some anonymity but unless the sender does not leak personal information in the message it can not be linked to the sender.

3 A-13) It is straightforward to generalize the N - 1 attack to an N - k attack, 0 < k < N. Describe the N - k attack.

The idea behind the $N-1$ attack is that an attacker controls all senders but one in a batch. This means that since the attacker know the recipients for the $N-1$ messages sent into the mix the attacker can filter out the sender-recipient pair for the one message sent by the other sender. This may be somewhat hard to do and the idea behind the $N-k$ attack is that the attacker controls all but k senders and therefore there is more uncertainty since there are now k input and k outputs that should be matches together. However it should be noted that the anonymity set is reduced so it is easier than the original problem. The $N-1$ attack is an extreme of the $N-k$ attack.

4 A-21) In August 2013, a large botnet used Tor Hidden Services to communicate with the Command and Control server. This resulted in an increase from 1 million to 6 million daily clients using Tor in just three weeks. As a result, the time required for downloading a 50 KiB file doubled from 1.5 seconds to 3.0 seconds. However, the amount of traffic on the network did not change very much? Why is that?

A Command and Control server(C&C) is typically used in order to provide commands to the different infected machines. These commands are typically not very large measured in bytes which is why the traffic was not increased so much. However when using Tor there is much cryptography operations involved such as key negotiation where symmetric keys are generated for the different onion routers. There is also work associated with encryption and decryption of data. These computations must be done for each user using Tor and if the number of users increase drastically as in this case it means that the number of operations will also increase drastically. In this case the performance was affected since the new users generated much work for the onion routers which lead to growing buffers.

5 A-23) Several users can use the same exit node in Tor, but different intermediate nodes. How can the exit node know where to send the response from the target?

When using Tor a circuit between the user and the target is created. The user shares a symmetric encryption key with each onion router. Each two adjacent nodes shares an circuit ID for their connection. This means that if we assume that three onion routers are used for the circuit one ID named ID_1 will be used for the connection between Alice and OR_1, ID_2 for the connection between OR_1 and OR_2 and ID_3 for the connection between OR_2 and OR_3. Each router keeps an table combining the circuit ids for the connection before and after. Lets take OR_2 as an example the circuit ID before the router is ID_2 and the circuit id after is ID_3. These two ids are stored together in an table in order to be able to know that traffic with circuit id ID_3 should be relayed to circuit id ID_2 and in the same way traffic with circuit id ID_2 should be relayed to circuit id ID_3.

Now lets consider an response from the target to the exit node. The exit node knows from which circuit id that the request originated from and when the response comes back from the target to the exit node the exit node will remember the circuit id. The exit node OR_n now just relays the response to the onion router OR_{n-1} where OR_n and OR_{n-1} shared the circuit id from which the request originated. Onion router OR_{n-1} now just uses its table to know which onion router OR_{n-2} that the response should be relayed to. This procedure is repeated until it reaches the end user. It can be said that each node just sends the response to the previous node in the circuit that is then responsible for sending the response closer to the end user. It should also be said that each node encrypts the response with its symmetric key shared with the end user. This means that when the response reaches the user it is encrypted with all the shared symmetric keys for the routers that is used in the circuit that the user has established. The user knows all the shared symmetric keys so its just to decrypt this response.

6 A-25) Explain what the point of the recognized field in a Tor cell is and how it makes communication more efficient.

When sending cells in the Tor network control cells are always interpreted by the receiving node and relay cells are just interpreted by the last node. The intent behind the recognized field is to make it easier to determine if the cell should be interpreted or not.

This is carried out by the user setting the recognized field to zeros before encrypting it with the symmetric keys. Each onion router receiving the cell decrypts the cell with its private key and checks the recognized field. If its nonzero the onion router knows that the cell should not be interpreted and can just relay the cell without having to compute any hash value. If the field is zero

then the cell needs to compute the hash value and compare it against the value of the digest field. This gives an significant speedup since it means that many cost full hash computations can be avoided.

7 A-26) A TCP handshake consists of the client and the server exchanging three messages: SYN, SYN-ACK and ACK. Explain why, in Tor, Alice can connect to a webserver and expect the TCP handshake with the server to be performed with low latency.

Lets consider the case where Alice wants to connect to the webpage eit.lth.se. Alice suspects that the admin behind the web page is sneaky and therefore wants to connect to the web page using Tor. Using two onion routers she sends a command basically saying that she wants to connect to the web server to onion router OR_1 that then sends this command to the exit node OR_2. It is then OR_2 that performs the TCP handshake exchanging the three messages SYN, SYN-ACK & ACK with the web server. The tcp handshake is between OR_2 and Website and can therefore be considered an ordinary TCP handshake performed with low latency. The exit node then sends an "connected" message back trough the circuit to Alice that can then begin to send an ordinary HTTP request back trough the circuit to the exit node. When using Tor you typically experience delays when trying to visit different web pages. These delays typically comes from the fact that the information from and to the web pages are relayed trough the network[1].

8 A-28) Show that the SSL/TLS handshake, when RSA is used, does not provide perfect forward secrecy.

Lets summarize the steps of SSL handshake using RSA[2].

1. Client sends ClientHello. Containing random client generated 28bytes used for calculating master secret, suggested cipher suit(RSA) and suggested compression algorithms.
2. Server sends ServerHello. Containing random server generated 28bytes used for calculating master secret, decided cipher to use(RSA) and decided compression algorithm.
3. Server sends its certificate chain. Possible sending also client authentication request.
4. Server sends server hello done.

[1] It is obvious that if the server serving the web page is slow then there will also be delays. But this delay would also exist when connecting directly to the web page.

[2] Following description of http://www.eit.lth.se/fileadmin/eit/courses/eit060/lect/Lect9b-10.pdf

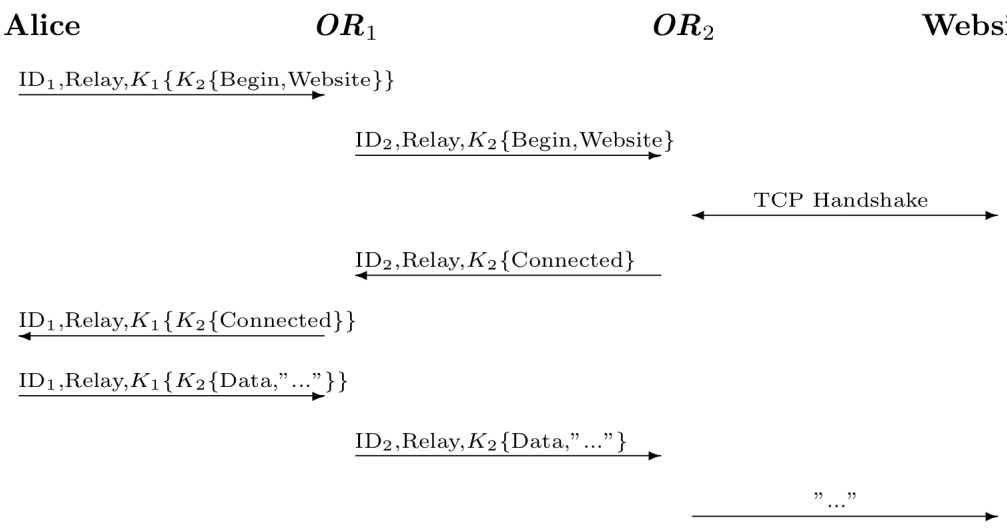

Figure 1: Connecting to a webpage(from lecture notes)

5. Client checks the certificate chain received from the server. If everything checks out the client continues otherwise it aborts.

6. Client generated pre-master secret and encrypts it with the public key of the server(found in certificate). This encrypted pre-master secret it sent to the server and both the client and the server can extract the key to be used from this premaster secret and the previously sent messages.

Looking at the above protocol from Mallory's point of view we see that the problem is that the client sends information needed to derive the session keys encrypted using the public key of the server. If Mallory could obtain the private key of the server then Mallory could read the handshake in plain text and therefore derive the session keys. If Mallory knows the session keys Mallory could read the traffic in plain text. Lets now assume that Mallory simply stores all SSL handshake traffic and all session traffic. Lets assume that Mallory now obtains the private key used by the server(bad ssh password etc). Since Mallory has stored SSL handshake traffic Mallory can now obtained the session keys used previously. This means that Mallory can also read the encrypted traffic by simply decrypt it by using the obtained session keys. Perfect forward secrecy states that it should not be possible to obtain the session keys even if the private assymmetric key is obtained. It therefore means that simply using RSA for SSL handshake does not provide perfect forward secrecy.