39. Anonymity and Tor

39. Anonymity and Tor

39.1. Overview

Imagine you wanted to use a group messaging app to share a public message but did not want your name attached to your message. How would you go about using the chat to achieve that?

One possible solution would be to randomly choose one member of the group on the participant list, send them a private message with your message and have them post the message with their name instead. This way your message gets posted to the forum but without your name attached to it.

This is the basis of *anonymity*, essentially a methodology that enables you to conceal your identity. In the context of the Internet, we may want *anonymous communications* wherein the identity of the source and/or destination are concealed. Note that anonymity is not the same as confidentiality, which is about keeping the message private, whereas anonymity is concerned with keeping one's identity private. As we will soon come to understand, anonymity on the internet is difficult without help; baked into the internet protocol is placing the IP address into the packet header, meaning that any message that is sent to a server reveals your IP address (which reveals quite a bit about your identity and is often one of the primary means of internet tracing). Malicious users, however, have it easier when it comes to anonymity due to the existence of *botnets*, which come about when someone hacks a set of machines and then controls those machines for various purposes (each machine is essentially a bot in the hands of the person who hacked it). The key takeaway here is that for the good guys to gain anonymity, we generally require some help, and what this comes down to is to simply ask someone else to send the message for you (as we saw in the previous example).

A majority of anonymity techniques require the use of a *proxy*, an intermediary (usually a somewhat trusted 3rd party) who relays our traffic for us. Say for example you wanted to visit a website, *www.example.com*, but you don't want to reveal to that server that you are visiting them. To hide your identity, you can route your traffic through a proxy server, which visits *www.example.com* on your behalf, gets the page, and returns the page to you. An issue that you might be thinking about right now is that you have to place your trust into this proxy server as they are now able to have access to your identity, so you are *not quite* completely anonymous, and you would be completely correct. We will see a couple of ways to circumvent this further down the road.

39.2. Anonymity in Cryptography

Let's try to understand in a little more detail how anonymity works when dealing with cryptography. Say that Alice wants to send a message M to Bob, but wants to conceal her identity when doing so. This means that not only should Bob not know that M is from Alice, but an evesdropper, Eve, should also not be able to determine that Alice is communicating with Bob. Using an intermediate proxy server, Alice can encrypt (M, Bob) using the proxy server's public key (K_{PS}), and sends this along to the intermediary. Note that the information packet needs to include the message and the person who the intermediary has to forward the information along to, in this case, Bob. Since Alice wants to send this information privately, she utilizes public key encryption. The proxy then decrypts the packet from Alice, retrieves the message, M, and the intended recipient, Bob, and then forwards the message to the intended destination. Essentially, all the proxy server is doing is accepting encrypted messages, decrypting them to extract the message and the destination, and then forwarding the message to the destination.

Notice that this method, as mentioned earlier, requires Alice to have complete trust in the proxy server in the hopes that they will not reveal her identity. To prevent having to place full trust in a third party server, we use the concept of onion routing.

39.3. Onion Routing

The key idea behind onion routing is the use of multiple proxy servers chained together in the hopes that at least one can be trusted. Again, Alice wants to send the message M to Bob, but this time she doesn't quite trust her proxy server, so instead decides to use onion routing to chain together 3 different intermediaries (let's call them Frank, Dan, and Charlie).

- Charlie is going to be the final proxy in the chain, so it needs to know the final recipient and the final message. So the packet Charlie receives will be (M, Bob) encrypted with Charlie's PK, $K_{Charlie}$ since Alice wants the message to be private. Charlie then decrypts the packet with his private key to obtain (M, Bob), then sends the message M to Bob.
- Since Alice doesn't trust Charlie to not reveal her identity, however, she wants to route the message through another proxy server, Dan. Again, the proxy requires the message and the recipient it forwards the message to. In this case, the message is $(M, Bob)_{KCharlie}$, and the recipient is Charlie. So the packet that Dan receives is $((M, Bob)_{KCharlie}, Charlie)_{KDan}$. Dan decrypts the packet using his private key to obtain $((M, Bob)_{KCharlie}, Charlie)$, then sends $M' = (M, Bob)_{KCharlie}$ to Charlie.
- Since Alice doesn't trust Dan either to not reveal her identity, she again wants to route the message through a different proxy server, Frank. Again, the proxy requires the message and the recipient that it forwards the message to. In this case the message is $((M, Bob)_{KCharlie}, Charlie)_{KDan}$ and the recipient is Dan. So the packet that Frank receives is $(((M, Bob)_{KCharlie}, Charlie)_{KDan}, Dan)_{KFrank}$. Frank decrypts the packet using his private key to obtain $(((M, Bob)_{KCharlie}, Charlie)_{KDan}, Dan)$, then sends $M'' = ((M, Bob)_{KCharlie}, Charlie)_{KDan}$ to Dan.

The overall routing scheme sends the packet from $Alice \to Frank \to Dan \to Charlie \to Bob$, where Frank, Dan, and Charlie are the three intermediaries, or proxies:

- Alice sends Frank $(((M, Bob)_{KCharlie}, Charlie)_{KDan}, Dan)_{KFrank}$
- Frank decrypts this using his private key and sends $((M, Bob)_{KCharlie}, Charlie)_{KDan}$ to Dan
- Dan decrypts this using his private key and sends $(M, Bob)_{KCharlie}$ to Charlie
- Charlie decrypts this using his private key and sends M to Bob
- For additional security, the message M could also have been encrypted using PK_{Bob}.

Note that this approach can be generalized to n intermediaries, and no one proxy knows both the sender and the recipient. In fact, even if $n - 1$ of the intermediaries were malicious and were colluding, as long as one recipient is honest, there is a low probability that they can connect Alice and Bob. If there were not a lot of people using the system however, if Frank and Charlie colluded, there would be a way to link the messages. In reality, though, each proxy server would likely be sending and receiving several thousands of messages, so it would be extremely difficult to link these two messages together.

Tor uses this type of onion routing for anonymous web browsing and censorship circumvention. While this example only depicted onion routing in one-direction, it can easily be scaled up, and Tor provides bidirectional communication.

39.4. Onion Routing Issues and Attacks

One of the downsides in onion routing is performance, as the message has to bounce off of several proxy servers before it reaches its destination, it takes a lot longer to get there. Each time the message goes through a proxy, there is an extra delay that is added to the latency. However, it should be noted that performance decreases linearly with the number of proxies added, and if you estimate that roughly half of the available proxy servers are honest then you only need to chain together a small number of proxies before you can be fairly sure that you have gained some level of anonymity (you can think of security going up exponentially, but performance going down linearly with the number of proxy servers added).

Another possible attack is one that was mentioned in the previous section, when the first and last proxy servers are under malicious control. The attacker can use timing information to link Alice and Bob, but, as mentioned earlier, this depends on the amount of traffic that is flowing through the proxy servers at that time. A possible defense is to pad messages (this is what Tor does but they note that it's not enough for defense), introduce significant delays, or if Bob is ready to accept encrypted messages, then the original message M can be encrypted.