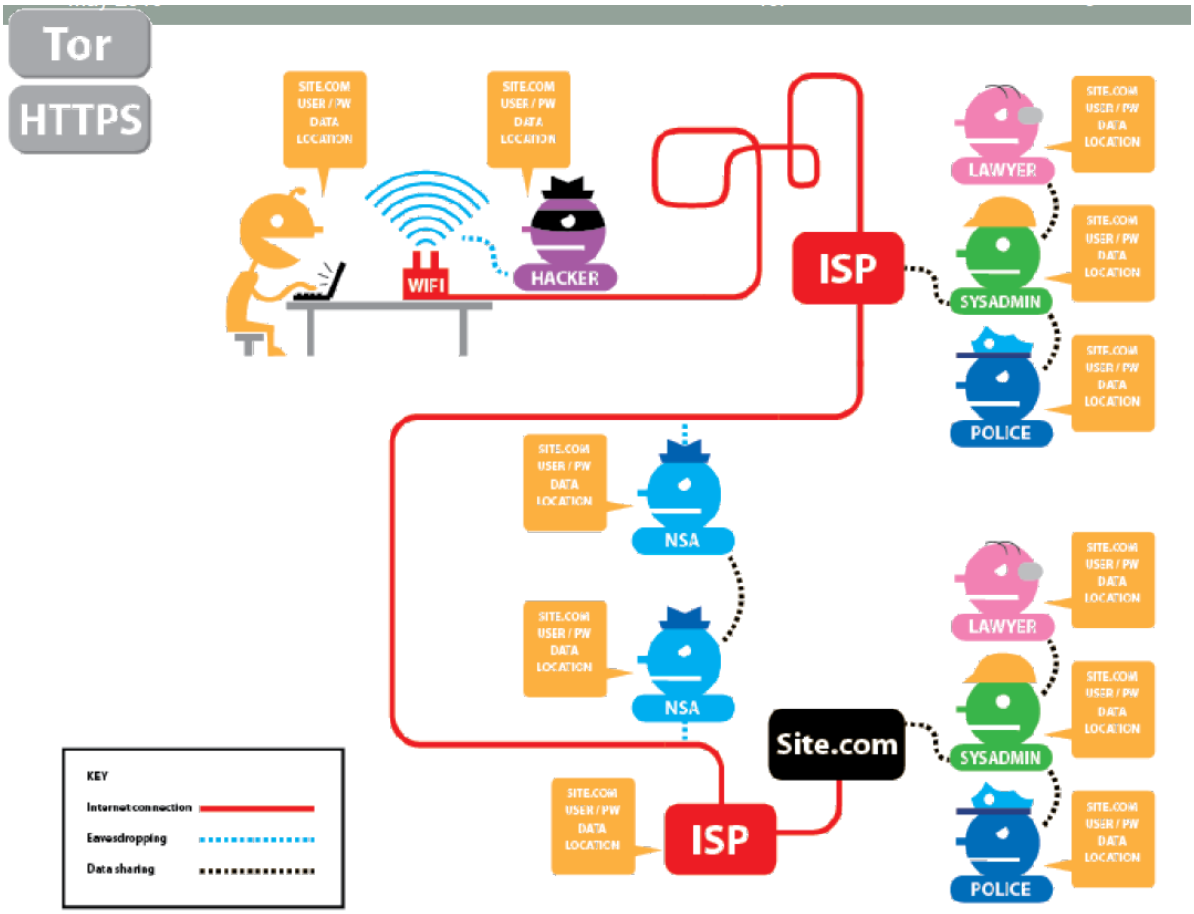
The Onion Router (Tor)

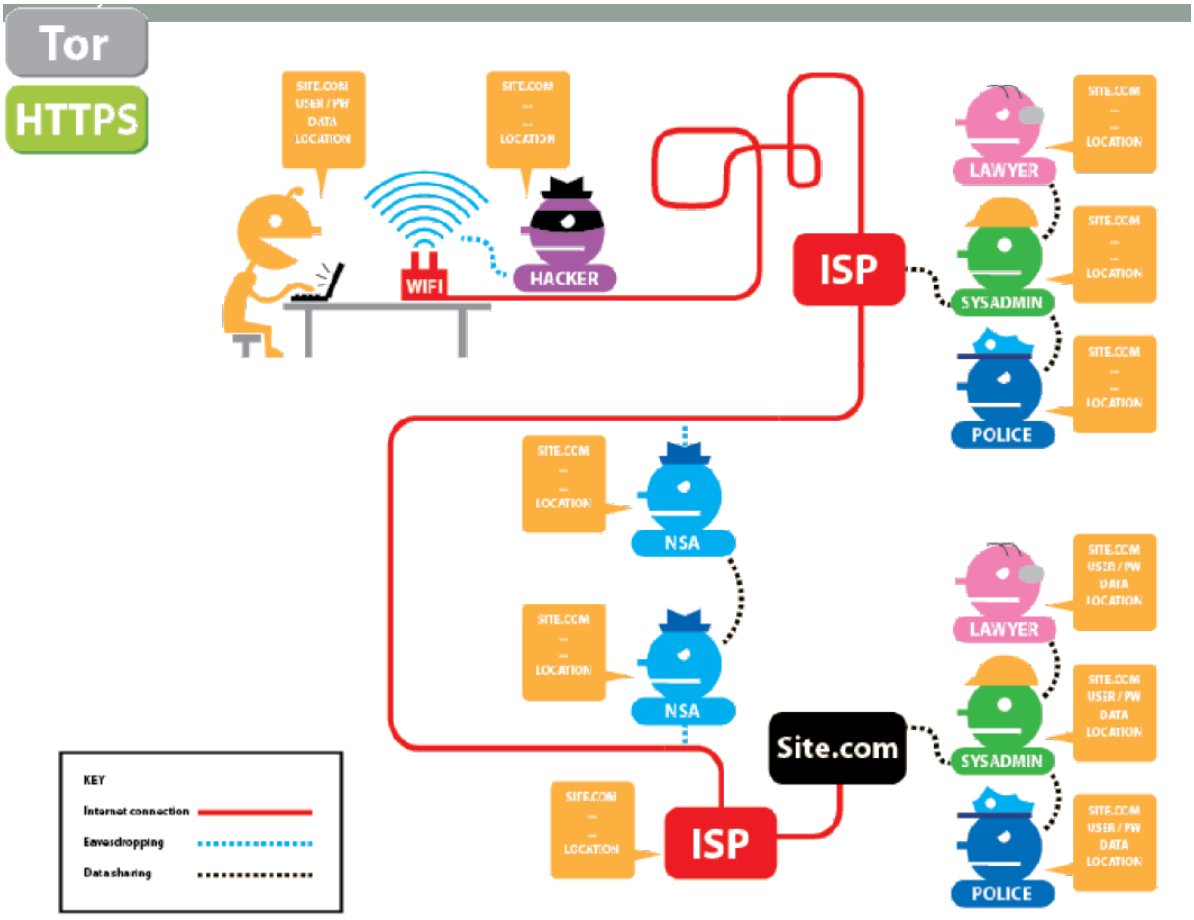
As we already know, Internet is designed as a public network. All the metadata associated to a network communication is visible from any network devices, indeed, routing information is public (source and destination of IP packets). Moreover, encryption does not hide identities, it hides only payloads. So, even IPSec cannot protect IPSec gateways addresses.

Tor is an open project that tries to achieve Anonymity of users. It is well integrated into browsers. It is a distributed anonymous communication service using an overlay network that allows people and groups to improve their privacy and security on the Internet. It is used to deny the tracking mechanisms of web sites or to connect to those internet services blocked by their local Internet providers. Tor allows users to publish their web sites and other services without needing to reveal their location.

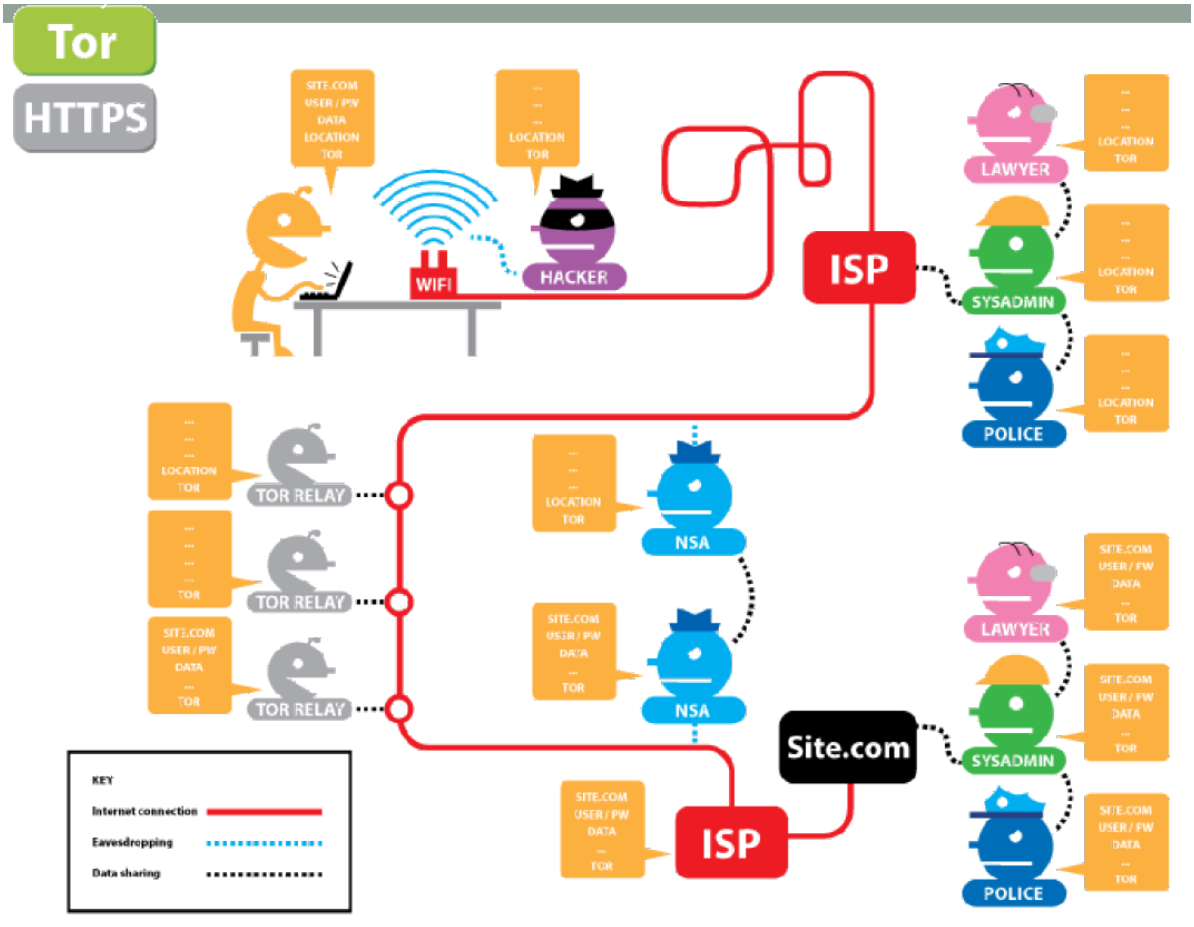
In the following images, we try to understand where Tor is useful (where means at what points of the communication).



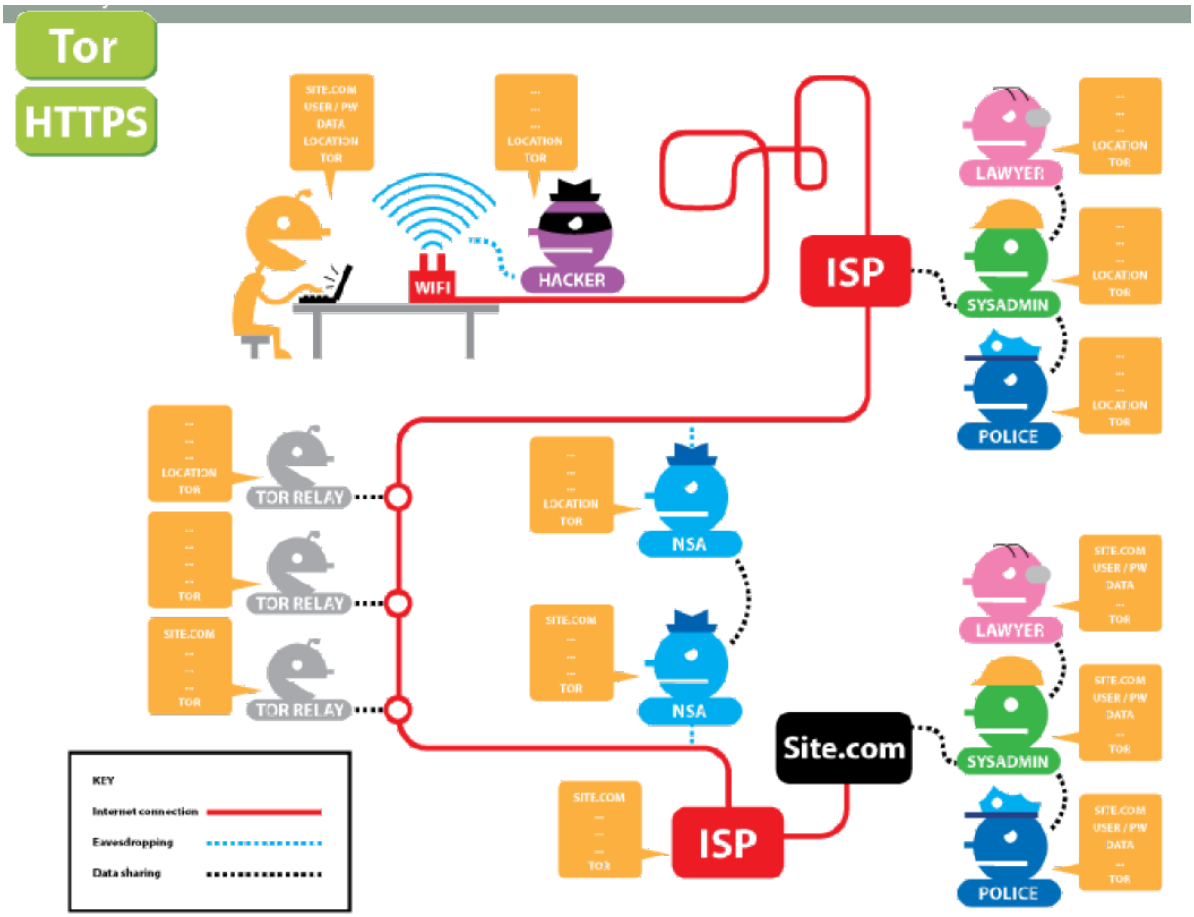
First of all, let's start with Tor and HTTPS disabled. Without HTTPS all the exchanged data between an user and a web site are left in clear, i.e. anyone at any point can collect the exchanged packets and read whatever information. Without Tor also the intention of contacting the web site is known.



HTTPS enabled only. It allows to encrypt, so to hide, the content of the exchanged payload, but the will of contacting the web site is still left in clear. Of course, the server where the web site is hosted must be able to decrypt the content of the payload in order to correctly process it.



Using Tor, instead, protects not only the content of the payload but also the intention of contacting the web site. Notice that these information are revealed when the packets reach the last Tor Relay (more on this later). Moreover, the location of the users are known between the user and the first Tor Relay. This is why, obviously, this is a simple connection between two hosts. the important thing to notice is that after the first Tor Relay, the location is somehow lost. This is due the random choice of the three Tor Relays that will be in charge of forwarding the packet to the correct web site.

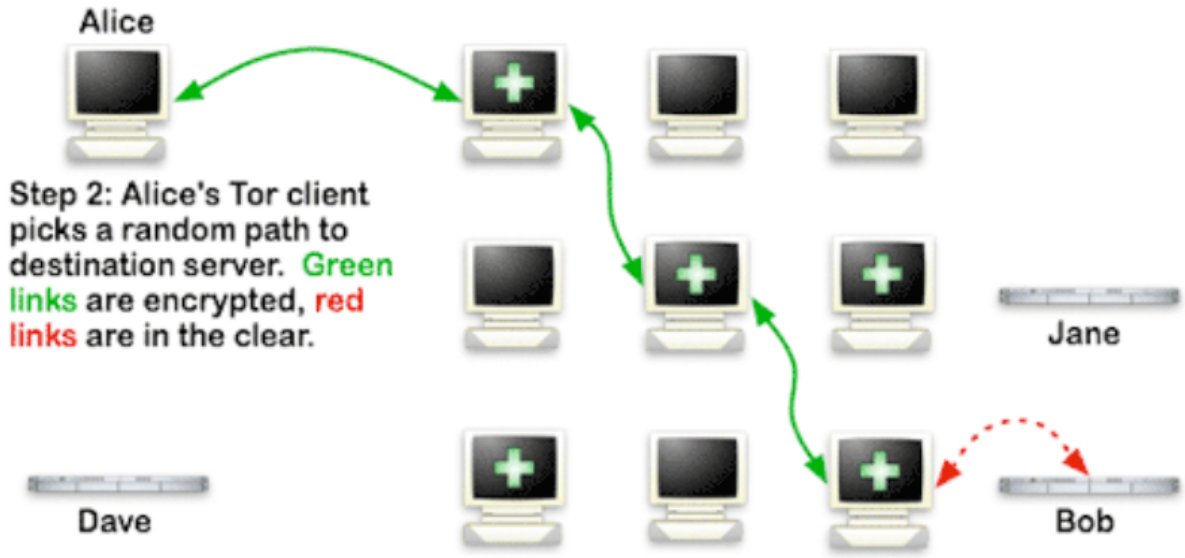


Now let's discuss the presence of both technologies. The only thing that is relevant to notice is that after the packets reach the last Tor Relay, the content still remains hidden. This is due to the HTTP use that encrypts the connection between two hosts, in this case the third Tor Relay and the web site.

The general scheme is:

- An user A uses a Tor Browser in order to contact the service B.
- The request is not directly sent to B, but it is forwarded for exactly three Relays in the Tor Network.
- The last Relay will be in charge to perform the last forwarding to the requested server B.

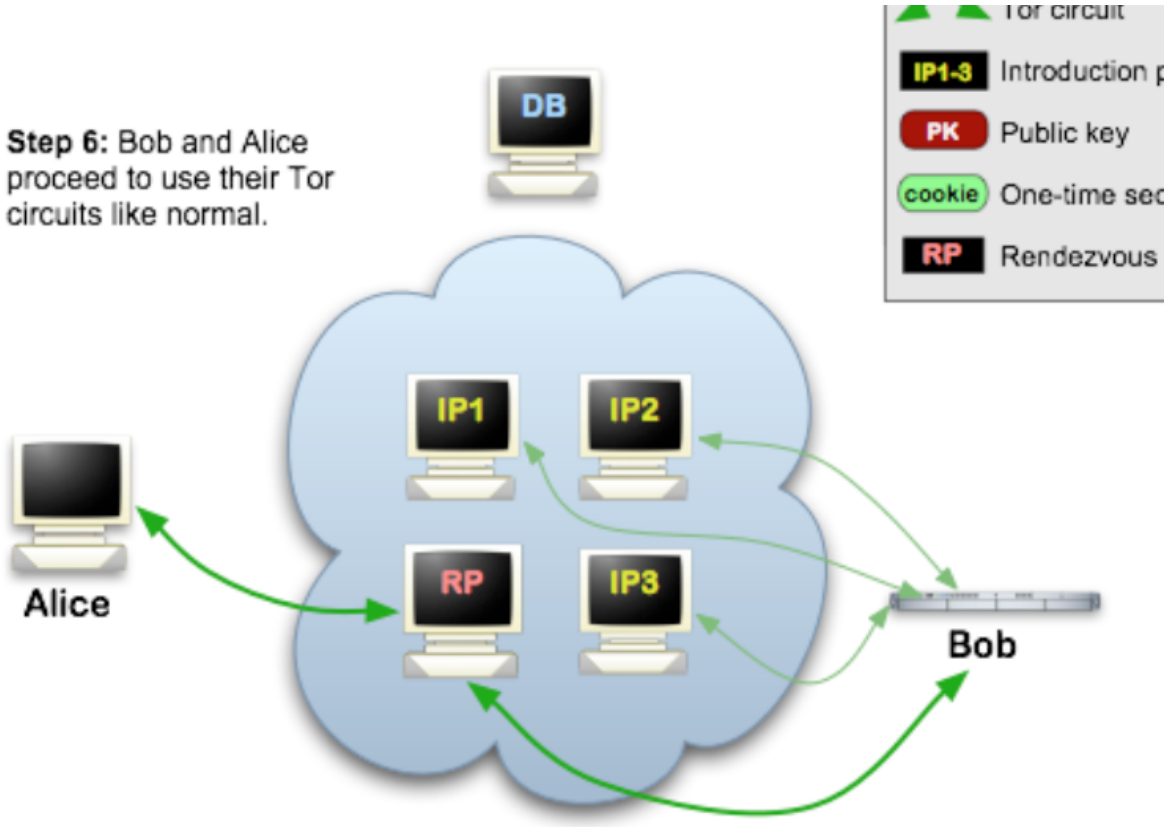
The first operation that A must perform is to contact a Directory Server, a fundamental entity in the Tor network, in order to retrieve a list of Tor clients that are up and running.
Then Alice randomly chooses three nodes in order to build a path towards the service that she want to contact. Notice that at each visit the three-relays path is chosen randomly from scratch.



After choosing the three Tor Relays, the path must be established through the so called Onion Routing, i.e. one hop at time. The Alice's goal is to share a symmetric key with all the three chosen nodes. Alice starts to negotiate a key with the first relay node of the path. Then she proceeds negotiating another session key with relay 2 THROUGH the tunnel between herself and the first relay. Finally, using the tunnel composed by the first and the second relay, A establishes the last session key with the third node. It must be noticed that each node knows ONLY its predecessor and successor: when A wants to establish a session key with relay 2, she doesn't contact directly that relay, instead, she asks to the first relay to do that. After the session key is exchanged between the two relays, the first one (the one which knows A identity) will encrypt that key with the session key between itself and A in order to forward that to A. The same thing is done between the second and the third relay. Location of the user A is knows only by the first relay.
After establishing the session keys, Alice starts to send her packets to the first Relay. The packet is encrypted as follow: let m be the message to send to the service B, Alice will send to the first relay this packet $E_{k_1}(E_{k_2}(E_{k_3}(m)))$, i.e. the message is encrypted with each session key starting from the third to the first (from this the Onion name). That packet is only decryptable by the first relay, so it will forward to the second relay the following packet $E_{k_2}(E_{k_3}(m))$. The third relay will receive this packet $E_{k_3}(m)$. The third packet, finally, is in charge to forward the message m to the service B.
The B's response is encrypted following the same order: firstly with K_3, then with K_2, finally with K_1.
Every few minutes the circuit is changed. For performance reasons, the first hop is usually kept fixed.
Notice that the messages have a fixed size, Tor uses padding to uniform them. Indeed, Anonymity is reached when all the users appear to be the same!

Directory servers maintain lists of active relay nodes, their location and their public keys. They check when a new node joins the network. The keys of the directory servers are ship within the Tor code.

Tor is based on the concept of hidden services. The goal is to deploy a server without sharing its physical location. It must be accessible by anyone from anywhere. This property protects the server to denial of service, censorship or physical attack.



In this picture there are all the basic elements to explain how an hidden service is made available for Tor users without knowing its IP address.

- Bob generates a long-term public key in order to allow Tor users to communicate confidentially with it.
- Bob chooses 3 Tor nodes as his Introduction Points, i.e. he builds three circuit to three different Tor nodes and teaches them to listen requests for Bob service.
- Then Bob publishes these information (public key + IP1-3) in the shared database called lookup service.
- Alice learns that Bob is hosting an hidden service and so retrieves details about Bob's service from the lookup service (public key + IP1-3).
- Then Alice builds a Tor path towards a Tor node called Rendezvous Point (RP) and gives to him a secret (nonce) to recognize Bob.
- Then Alice anonymously contact an Introduction Point of Bob (building a three-relays path) and gives it a message, encrypted with the public key of Bob, telling that she wants to contact him, the rendezvous point RP, the secret nonce and the start parameters for a Diffie-Hellman handshake protocol.
- IF Bob wants to talk with Alice, he builds a circuit to Alice's RP, sending the secret nonce, in order to be authenticated, the second parameter for the DH exchange and an hash of the actual secret shared (for message integrity).
- The RP now connects the two circuits. Notice that between Alice and RP and between RP and Bob there are other Tor Relays, i.e. RP doesn't know who is talking whom with.
- Now after acknowledging the correctly setup circuit, Alice and Bob can talk with each other normally.