Master in Computer Engineering (MEI)
Integrated Master in Informatics Engineering (MiEI)

Specialization Profile **CSI**: Cryptography and Information Security

Engenharia de Segurança

jose.miranda@devisefutures.com

Tópicos

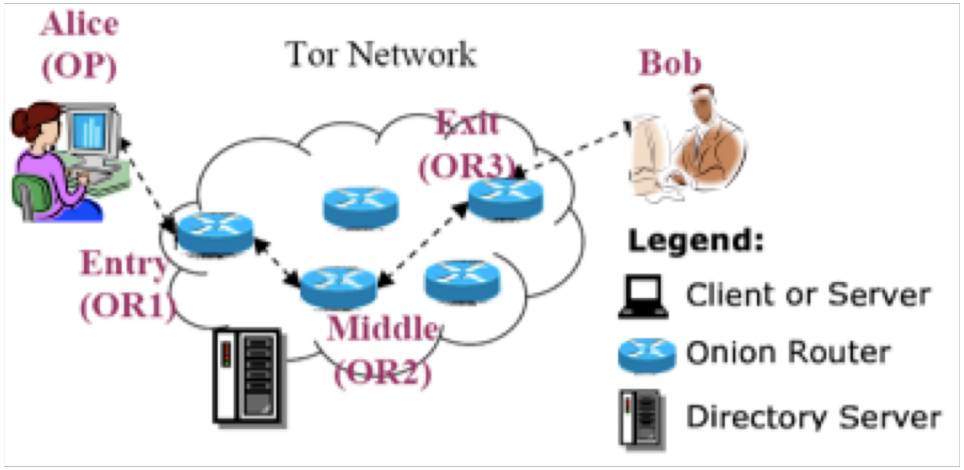
- Applied Cryptography
  - Cryptographic protocols / applications
    - TOR (The Onion Router)

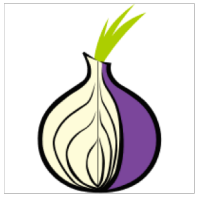
Bibliography (English):

- TOR overview - https://www.torproject.org/about/overview.html.en
- Tor: The Second-Generation Onion Router - https://svn.torproject.org/svn/projects/design-paper/tor-design.pdf

TOR (The Onion Router)

- **Cryptographic network protocol** whose purpose is:
  - Ensure **peer-to-peer anonymity** (at the non-application level) of an Internet user;
    - Note that all the protocols we have seen so far establish secure / private / confidential tunnels, but do not allow point-to-point anonymity (at the non-application level), as packet headers (TCP / UDP / IP) still reveal information about the user.
  - Allow anonymous services (hidden services) - www and other services - without revealing their location.





TOR (The Onion Router)

- Internet overlapped network consisting of ***Onion Routers*** (**OR**)
  - Each OR performs as a normal user process without the need for special privileges;
  - Each OR connects to other ORs over a TLS connection;
  - There are "more reliable" ORs that act as Directory Server - they provide signed lists of known ORs and their current state, which are periodically downloaded by TOR users –;
  - Each OR has a pair of long-lived keys (*identity key*) used to sign the TLS certificates, the OR descriptor (contains public keys, address, bandwidth, exit policy, etc.) and directory (when the OR is a Directory Server);
  - Each OR has a periodically rotated pair of short-lived keys (*onion key*) used to establish session keys with the user (via Diffie-Hellman).
- Each user runs local software: **Onion Proxy** (**OP**)
  - OP obtains data from the directory, establishes circuits through the TOR network and manages user application connections.

How Tor Works:

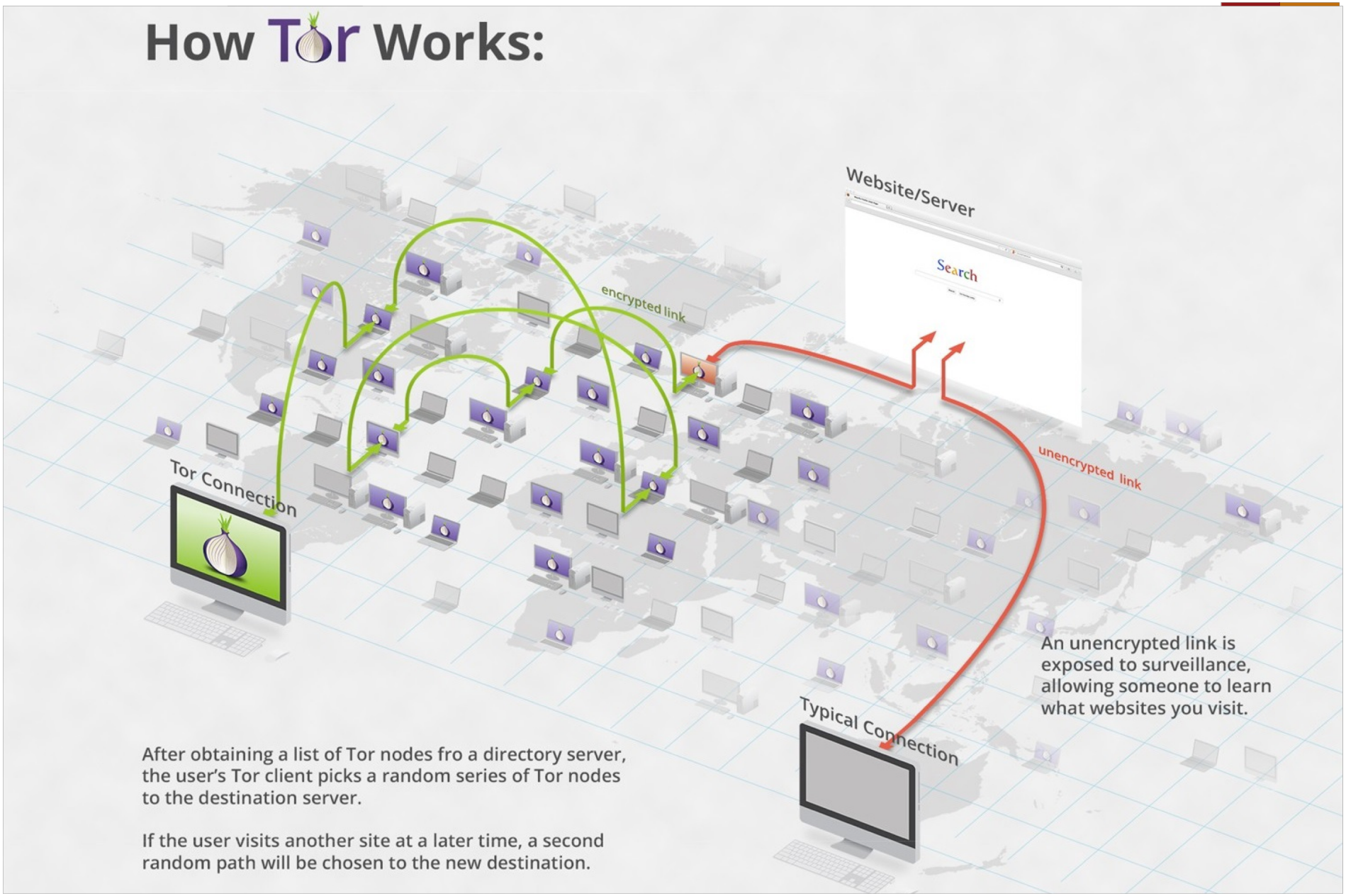
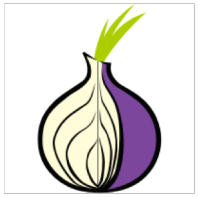


Image: https://www.extremetech.com/internet/226106-onionscan-tests-dark-web-sites-to-see-if-they-really-are-anonymous



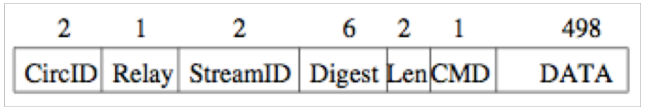
TOR (The Onion Router)

- OR communicate with each other and with OP in fixed-length packets (cells), over TLS connections.
  - Each cell is 512 bytes and consists of a *header* and a *payload*;;
  - The *header* includes the circuit identifier circID which indicates which circuit the cell refers to (multiple circuits can be multiplexed over the same TLS connection), and a CMD command that describes what to do with the *payload*;

  - According to the CMD, the cells can be control cells (always interpreted by the receiving OR) or relay cells (take data point/OR to point/OR);
  - The CMD of the control cells can be :
    - padding (used for *keepalive*),
    - create/created (used to create a new circuit), ou
    - destroy (used to close a circuit);

TOR (The Onion Router)

- OR communicate with each other and with OP in fixed-length packets (cells), over TLS connections.
  - Relay cells have an additional header with streamID (stream identifier: multiple streams can be multiplexed over the same circuit), point-to-point hashing (for integrity checking), relay payload size, and relay CMD;



  - All relay header and relay payload content (i.e., cell payload) is encrypted or deciphered (AES 128 bits) sequentially as the cell moves along the circuit;
  - The relay CMD can be :
    - data (for data to be communicated in the stream),
    - begin (to open new stream), end (to close existing stream),
    - teardown (to close a "damaged" stream),
    - connected (to notify the OP of a successful begin),
    - extend/extended (to extend the circuit by one more OR),
    - truncate/truncated (to destroy only part of the circuit),
    - sendme (to control congestion in an OR), ou
    - drop (to test a stream);

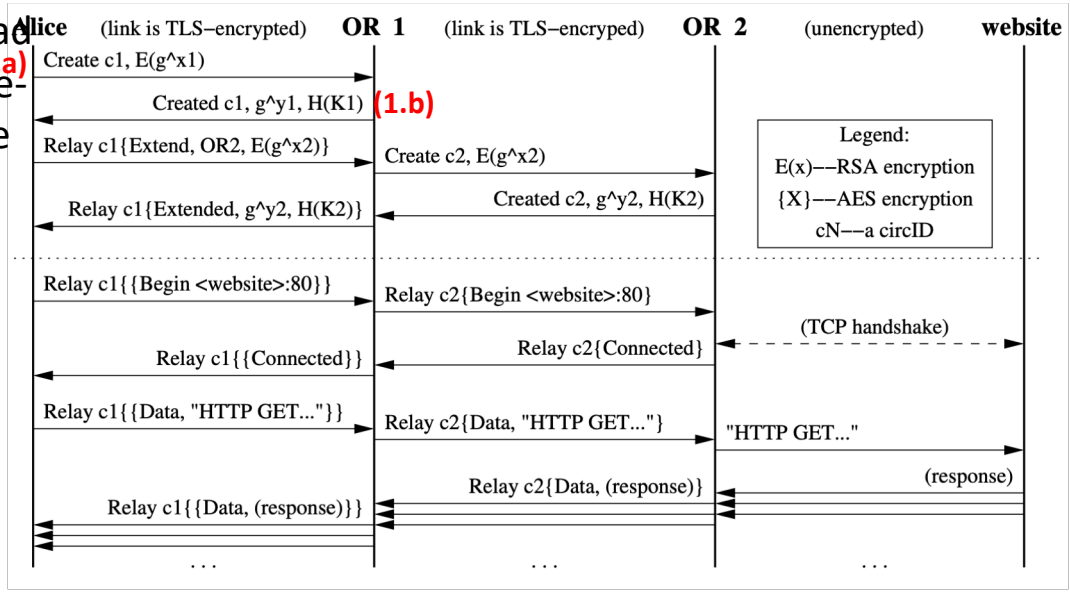
TOR - Anonymity

- OP pre-sets circuits (usually 3 OR) and switches to a new circuit once per minute, ensuring that only a limited number of requests can be connected to each other in the output OR;
- OP constructs the circuit incrementally, negotiating a symmetric key with each OR of the circuit, OR to OR. Note that the Ors are chosen from the OR list provided by the Directory Server that has the long-term (*identity key* to sign TLS certificates) and short-term keys (*onion key* to establish Diffie-Hellman session keys).

Step 1a: The OP sends a *create* control cell to the first node (OR1) of the path chosen by the OP, choosing a new circID and with the payload of the cell containing the first half of the Diffie-Hellman key exchange (g^{x1} encrypted with the public key of the OR1 *onion key*).

Step 1b: The OR1 responds with a *created* control cell, containing g^{y1} as well as the hash of the negotiated key K1 ($K1 = g^{x1.y1}$).

From this moment, OP and OR1 can communicate the relay cell with the payload encrypted with the key K1.



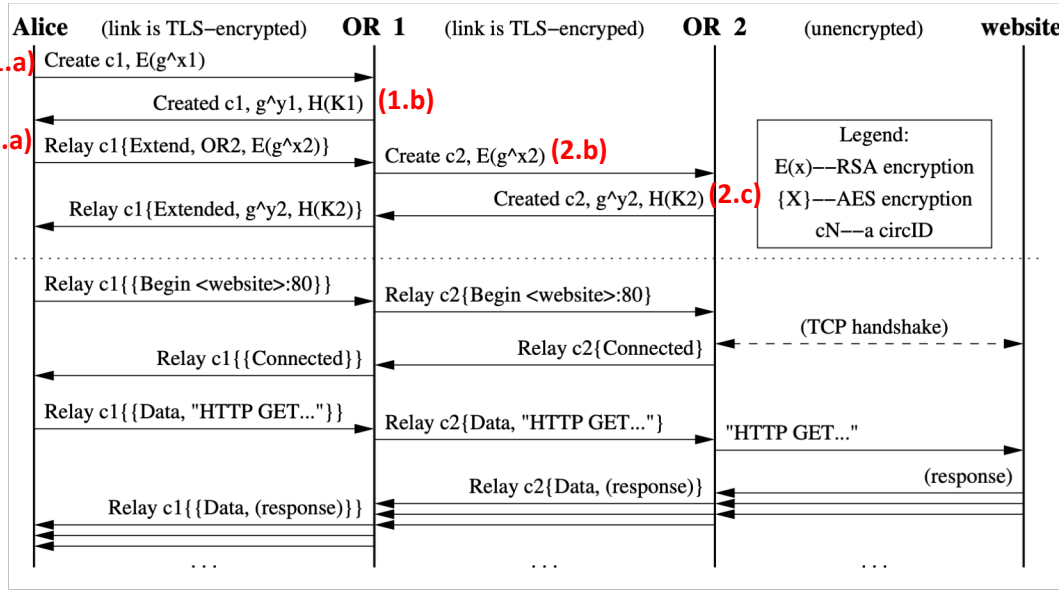
TOR - Anonymity

Step 2a: To extend the circuit, the OP (Alice) sends a *relay extend* cell to OR1, identifying the next OR (OR2) and with g^{x2} encrypted with the *onion key* public key of OR2 (E (g^{x2})).

Step 2b: OR1 chooses a new CircID, copies E(g^{x2}) to the payload of a *create* control cell, and sends it to OR2 .

Step 2c: OR2 responds with a *created* control cell and OR1 copies the payload from that cell to an *relay extended* cell that sends to the OP. The circuit is extended to OR2 and the OP and OR2 share the common key K2 = $g^{x2.y2}$).

From this moment, OP and OR2 can communicate the *relay* cell with the payload encrypted with the key K2.

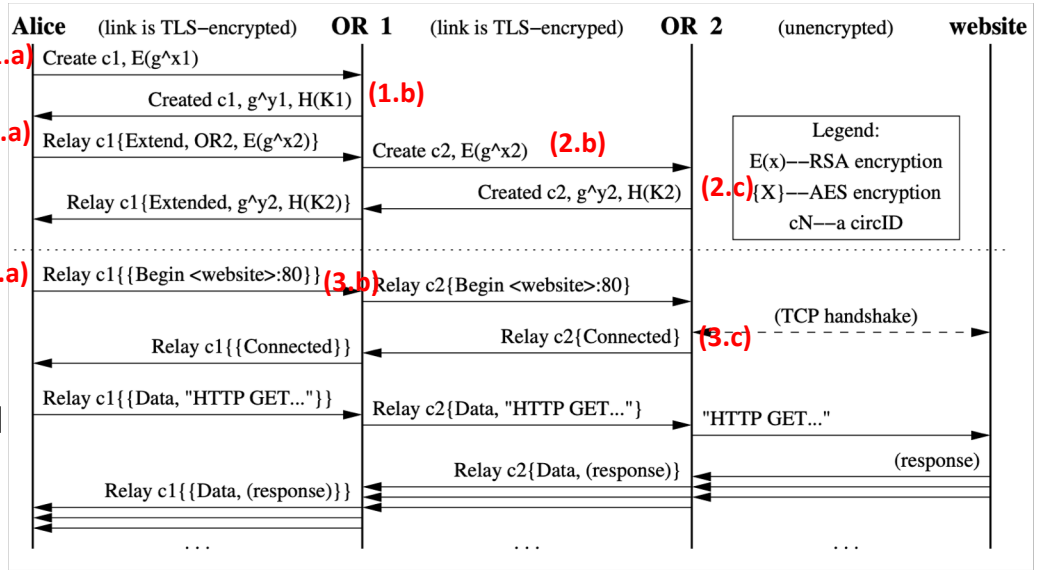


To extend the circuit to additional nodes (OR_{n+1}), OP (Alice) performs the previous steps, always directing the last OR (OR_n) in the circuit to extend to the new OR (Or_{n+1}).

TOR - Anonymity

The circuit establishment protocol guarantees unilateral authentication (OP knows that it is exchanging keys with the OR, but the OR does not know who is opening the circuit - i.e., OP does not use its public key and remains anonymous).

Once the circuit is established, OP can send *relay* cells to the last OR of the circuit.

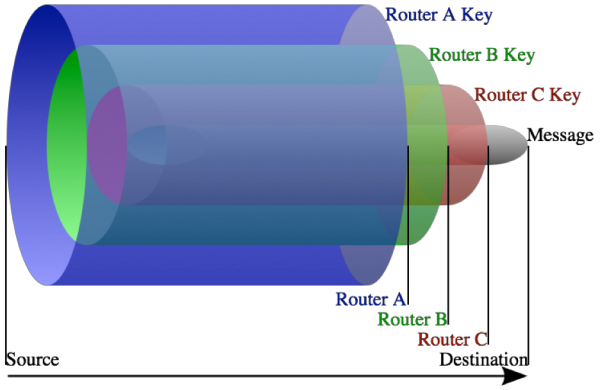


Step 3a: OP (Alice) sends the *relay* cell to the destination OR. To construct this cell, it inserts all the necessary data, generates the hash / digest and cipher with each symmetric key exchanged with each of the ORs (in this case it is enciphered with OR1 and then with OR2).

Step 3b: The first OR (OR1) decrypts the payload and checks to see if the hash is correct. If it is, it performs the command requested by the OP. If not, it picks up the decrypted payload and send a relay cell to the next OR in the circuit.

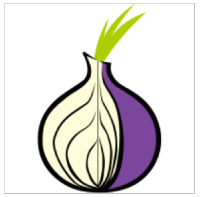
Stopping condition, i.e., reached destination OR

Step 3c: The final OR responds through the established circuit with a *relay* cell with the encrypted payload for the OP. Intermediate ORs add new levels of cipher as they "forward" it to the OP.

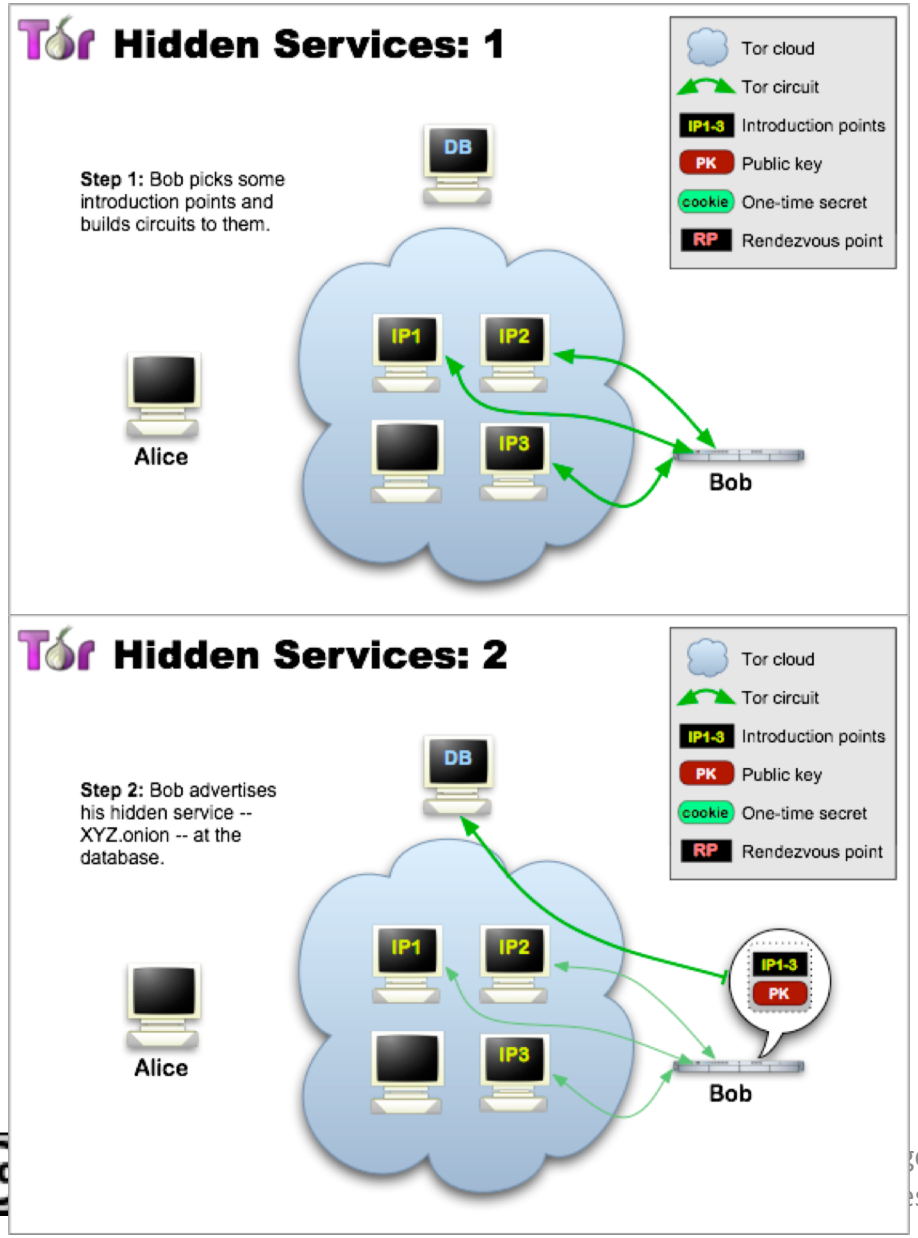


TOR – Rendezvous Points and Anonymous Services

- **Rendezvous Points** are the support for the provision of anonymous services (also called *responder anonymity*).
  - In the TOR network, the provision of **anonymous services** allows an OP (Bob) to provide TCP services (for example, web server) without revealing its IP address.
  - Since the OP (Alice) accessing the anonymous service is also anonymous, both the accessing OP and the OP that is accessed are anonymous.



TOR – Rendezvous Points and Anonymous Services

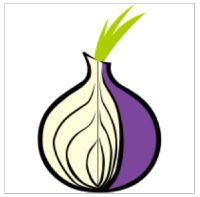


Steps 1 and 2:

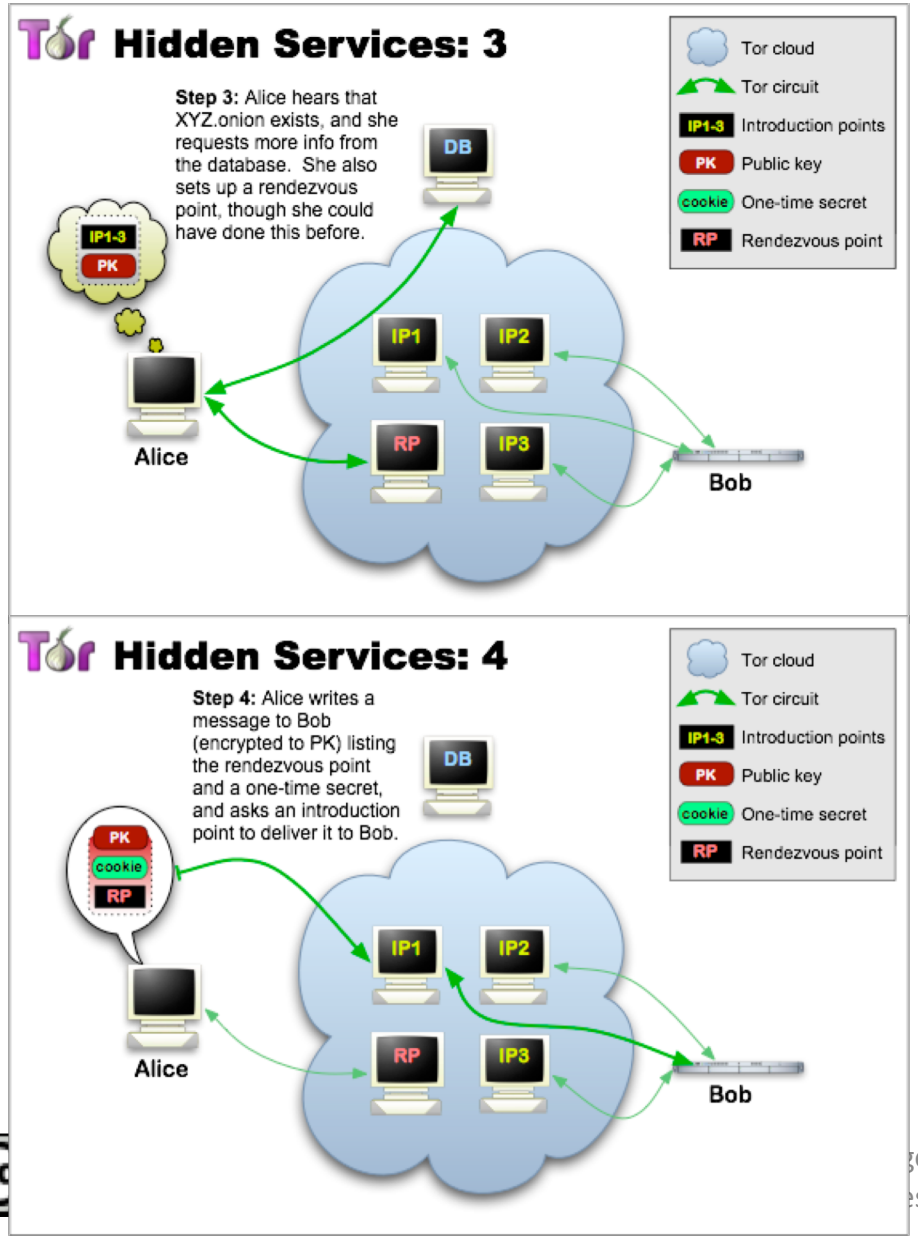
a. Bob generates a long-time key pair to identify its Web service (the public key becomes the service identifier);

b. Bob chooses some *introduction points* and announces them in the *Directory Server* by signing the advertisement (service descriptor) with his private key;
   - The anonymous service descriptor contains the service public key and a summary of each entry point;
   - The descriptor / service will be found by clients accessing XYZ.onion on the TOR network, where XYZ is a 16-character name derived from the public service key.

c. Bob creates a TOR circuit (as seen in the last slides) for each of the entry points and asks them to wait for orders.



TOR – Rendezvous Points and Anonymous Services



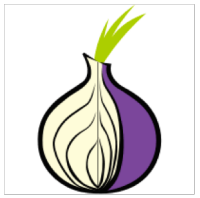
Steps 3 and 4:

a. Alice knows about the XYZ.onion service and accesses its details (public key and entry points) in TOR through the Directory Server;

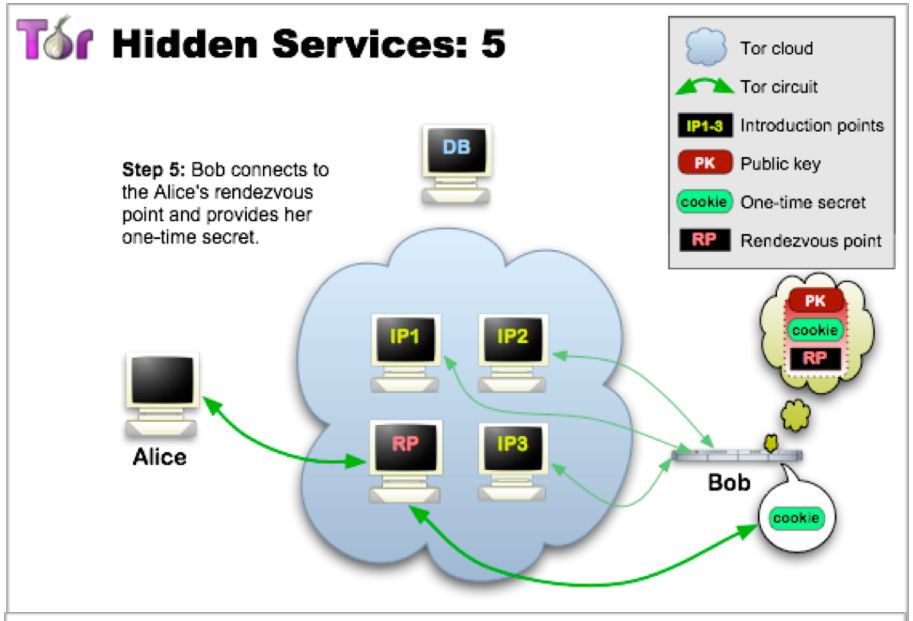
b. Alice chooses an OR as a *rendezvous point* (RP) for the connection to the XYZ.onion service;

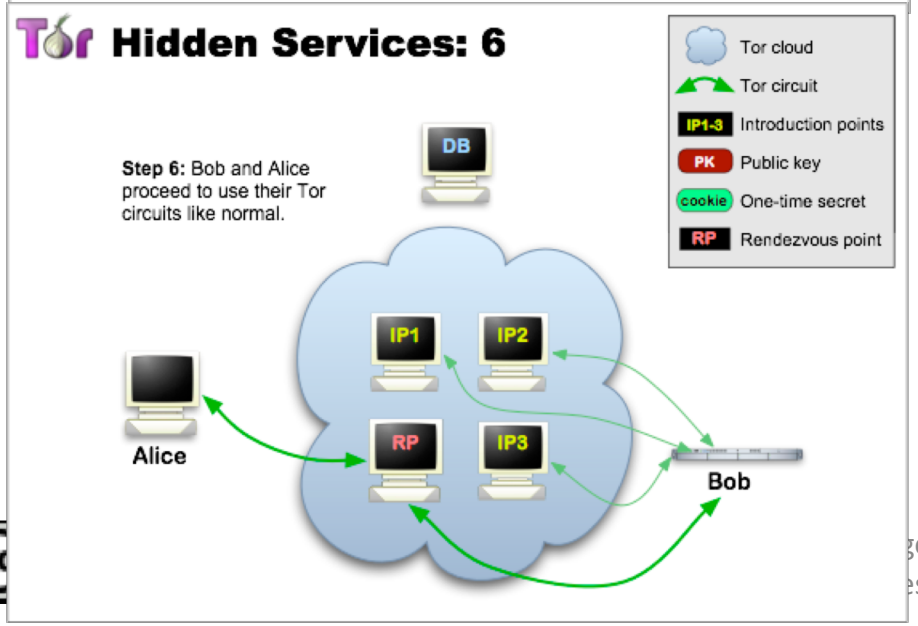
c. Alice builds a TOR circuit up to the RP and provides it with a "*rendezvous cookie*" for later recognition of the XYZ.onion service;

d. Alice opens an anonymous *stream* to one of the entry points and provides a message (encrypted with the XYZ.onion public key) with information about the RP, the "*rendezvous cookie*", and the Diffie-Hellman key exchange start. The input point forwards the message to the XYZ.onion service via the TOR circuit created in the previous steps.



TOR – Rendezvous Points and Anonymous Services





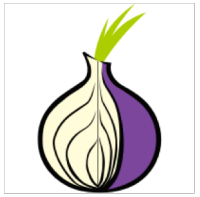
Steps 5 and 6:

a. If Bob (XYZ.onion) wants to talk to Alice (OP), Bob creates a TOR circuit to the RP and sends the "*rendezvous cookie*", the second part of the Diffie-Hellman key exchange, and a session key hash to share with Alice;

b. The RP connects the circuit of Alice with Bob circuit (usually the circuit consists of 6 ORs: 3 chosen by Alice being the third the RP and the other 3 chosen by Bob). Note that the RP can not recognize Alice, Bob nor the data they transmit;

c. Alice sends a *relay begin* cell through the circuit that, upon reaching Bob´s OP, connects to the service provided by Bob (for example, a Web server);

d. An anonymous *stream* has been established and Alice and Bob communicate in the normal way of a TOR *stream*.



TOR – Possible Attacks

What happens if our server acts as an input node?

- Knows that John's IP is using the TOR network (interesting for SIGINT - *signal intelligence* - but little else).

What happens if our server acts as an intermediate node?

- Nothing. It receives encrypted information from a TOR node that goes to another TOR node.

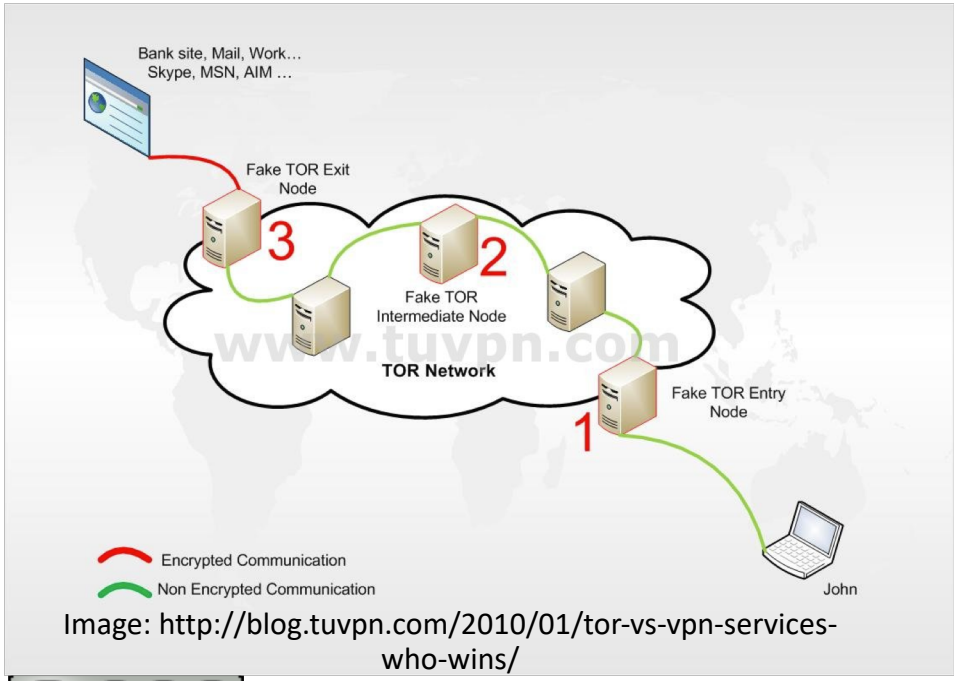


Image: http://blog.tuvpn.com/2010/01/tor-vs-vpn-services-who-wins/

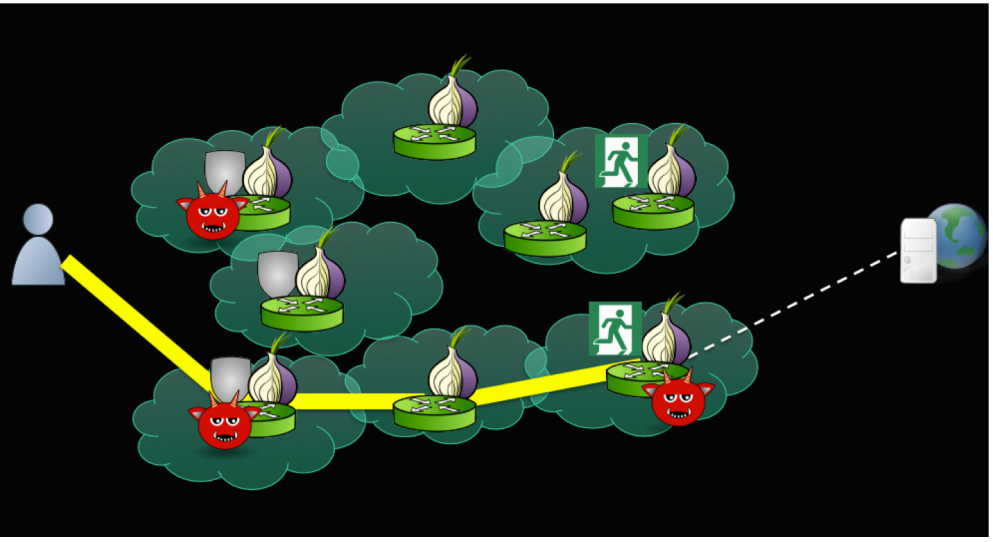
What happens if our server acts as an outbound node?

- The mission of the outbound node is to decrypt the communication, and send it to the destination (web) service, so it does not know who was the originator of the communication but knows what the communication is.
- We can argue that if the destination is SSL, there is no way to access John's plaintext communication.
- Wrong. It has been shown that a *MitM attack* on SSL is possible on the output node.

TOR – Possible Attacks

What happens if one of our servers acts as an input node and another acts as an output node?

- It is possible to calculate correlation coefficients - through mathematical formulas of the area of probability and statistic - in the analysis of packages in the two nodes, based on the frequency, timing and size of the package.
- It is assumed that 80% of users can be de-anonymised within 6 months.



- High cost, since a large percentage of ORs (TOR has about 7,000 ORs) must belong to the entity that wants to perform a correlation attack.
- Used by government agencies (BND, GCHQ, NSA).
- It is assumed that "Silk Road 2.0" was de-anonymised based on this attack.

Image: https://www.deepdotweb.com/2016/10/25/tors-biggest-threat-correlation-attack/

TOR

- Example carried out in less than ten minutes, from the same machine, at www.whatismyip.com

Tor Browser

IP Address: 87.106.148.90
City: Karlsruhe
State/Region: Baden-wurttemberg
Country Code: DE
Postal Code: 76229
ISP: 1&1 Internet Ag
Time Zone: +01:00 UTC/GMT
Latitude: 49.0047
Longitude: 8.3858

IP Address: 92.222.172.41
City: Roubaix
State/Region: Nord-pas-de-calais
Country Code: FR
Postal Code: 59689
ISP: Ovh Sas
Time Zone: +01:00 UTC/GMT
Latitude: 50.6942
Longitude: 3.1746

IP Address: 94.242.206.170
City: Steinsel
State/Region: Luxembourg
Country Code: LU
Postal Code: I-7349
ISP: Root Sa
Time Zone: +01:00 UTC/GMT
Latitude: 49.6769
Longitude: 6.1239

Firefox

IP Address: 94.132.113.217
City: Porto
State/Region: Porto
Country Code: PT
Postal Code: -
ISP: TVCABO Portugal, S.A.
Latitude: 41.1496
Longitude: -8.611

- What's the explanation?