



# Mestrado em Engenharia Informática (MEI)

# Mestrado Integrado em Engenharia Informática

## (MiEI)

Perfil de Especialização **CSI** : Criptografia e Segurança da Informação

Engenharia de Segurança





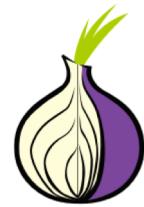
# Tópicos

- Criptografia Aplicada
  - Protocolos/aplicações criptográficas
    - TOR (The Onion Router)

## Bibliography (English):

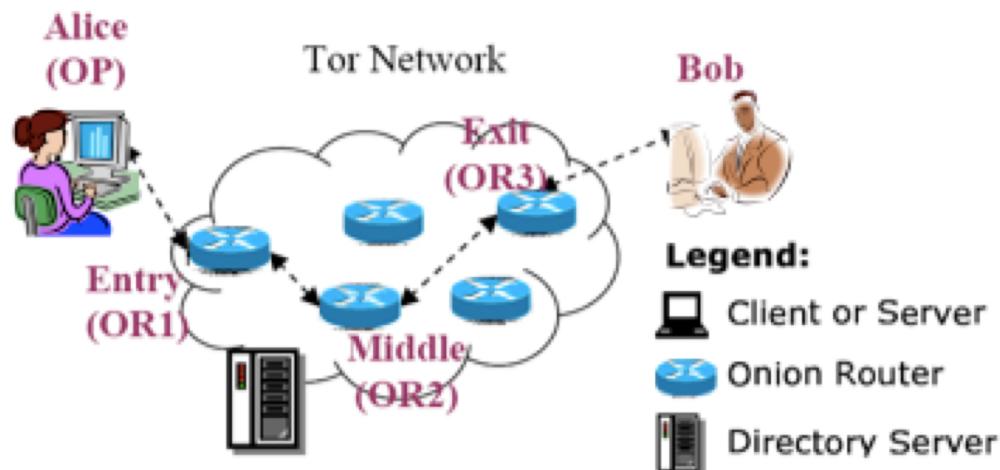
- TOR overview - <https://www.torproject.org/about/overview.html.en>
- Tor: The Second-Generation Onion Router -  
<https://svn.torproject.org/svn/projects/design-paper/tor-design.pdf>

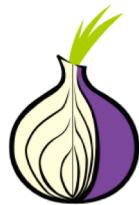




# TOR (The Onion Router)

- **Protocolo criptográfico de rede** cujo objectivo é:
  - Garantir a **anonimidade ponto-a-ponto** (ao nível não aplicacional) de um utilizador na Internet;
    - Note que todos os protocolos que vimos até agora estabelecem túneis seguros/privados/confidenciais, mas não permitem anonimidade ponto-a-ponto (ao nível não aplicacional), na medida em que os *headers* dos pacotes (TCP / UDP / IP) ainda revelam muita informação sobre o utilizador.
  - Permitir disponibilizar **serviços anónimos** (*hidden services*) – www e outros serviços – sem revelar a localização dos mesmos.





# TOR (The Onion Router)

- Rede sobreposta à Internet constituída por ***Onion Routers (OR)***
  - Cada OR executa como um processo normal do utilizador sem necessidade de privilégios especiais;
  - Cada OR conecta-se a outros OR através de uma conexão TLS;
  - Existem OR “mais confiáveis” que actuam como *Directory Server* – fornecem listas assinadas dos OR conhecidos e seu estado actual, que são descarregadas periodicamente pelos utilizadores do TOR –;
  - Cada OR tem um par de chaves de identidade de longo tempo (*identity key*) utilizado para assinar os certificados TLS, o descritor do OR (contém chaves públicas, endereço, largura de banda, política de saída, etc.) e a directoria (no caso dos *Directory Server*);
  - Cada OR tem um par de chaves de curto prazo (*onion key*), rodadas periodicamente, utilizado para estabelecer chaves de sessão com o utilizador (através de Diffie-Hellman).
- Cada utilizador executa um software local: ***Onion Proxy (OP)***
  - OP obtém dados da directoria, estabelece circuitos através da rede TOR e gere conexões das aplicações do utilizador.



# How Tor Works:

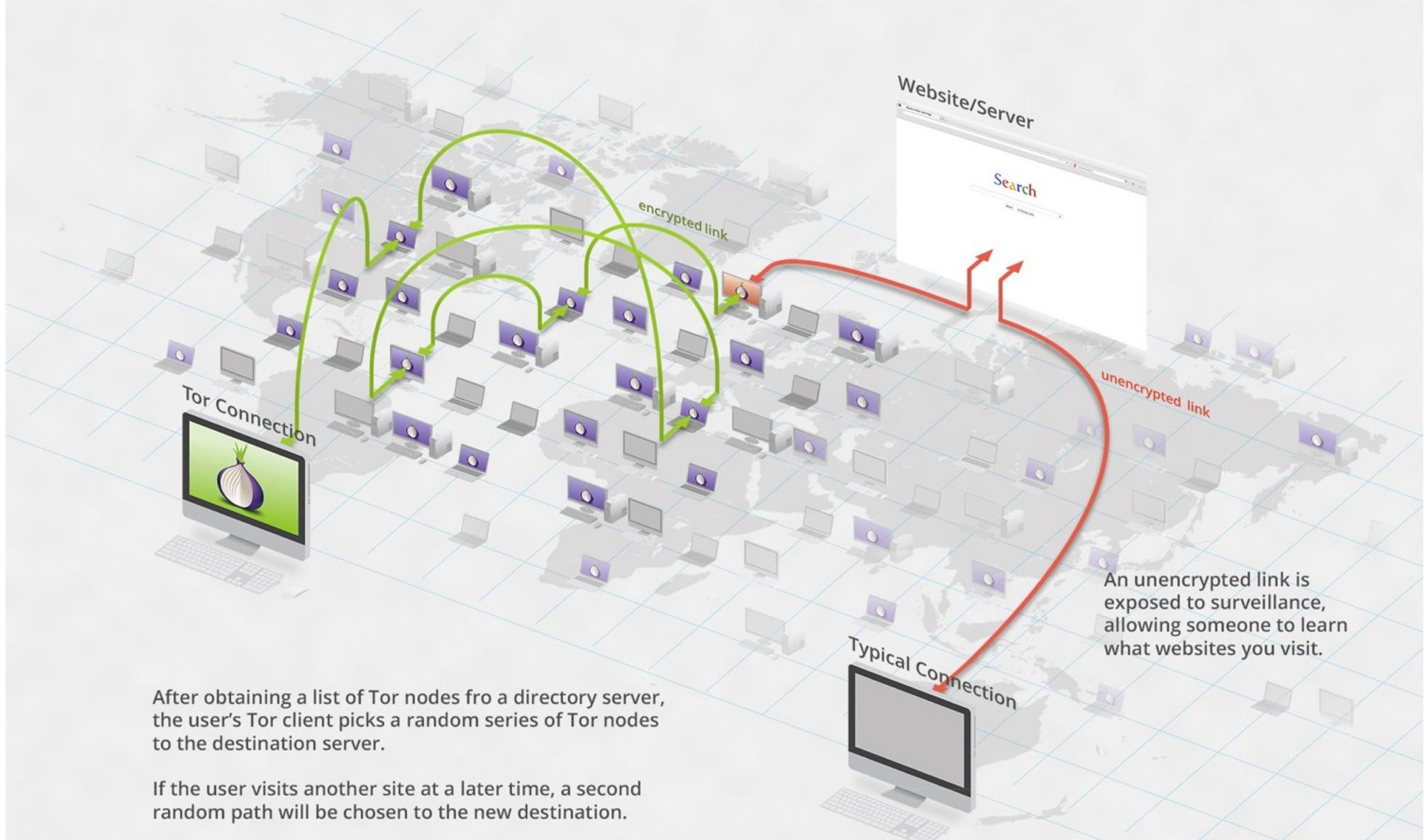
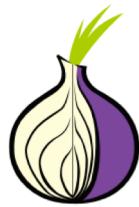


Imagen: <https://www.extremetech.com/internet/226106-onionscan-tests-dark-web-sites-to-see-if-they-really-are-anonymous>





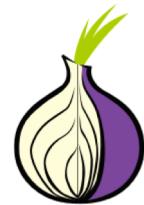
# TOR (The Onion Router)

- OR comunicam entre si e com OP através de conexões TLS em pacotes (células) de tamanho fixo.
  - Cada célula tem 512 bytes e é constituída por um *header* e um *payload*;
  - O *header* inclui identificador do circuito circID que indica a que circuito a célula se refere (vários circuitos podem ser multiplexados sobre a mesma ligação TLS) e um comando CMD que descreve o que fazer com o *payload*;



- De acordo com o CMD, as células podem ser *control cells* (sempre interpretadas pelo OR que as recebe) ou *relay cells* (levam dados ponto a ponto);
- O CMD das *control cells* pode ser:
  - padding (utilizado para *keepalive*),
  - create/created (para criar novo circuito), ou
  - destroy (para finalizar circuito);

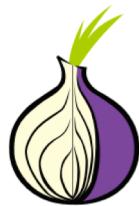




# TOR (The Onion Router)

- OR comunicam entre si e com OP através de conexões TLS em pacotes (células) de tamanho fixo.
  - As *relay cells* têm um *header* adicional com o streamID (identificador do stream: vários streams podem ser multiplexados sobre o mesmo circuito), hash ponto-a-ponto (para verificação de integridade), o tamanho do payload do relay e o CMD do relay;
- Todo o conteúdo do relay header e relay payload (i.e., o payload da célula) é cifrado ou decifrado (AES 128 bits) sequencialmente à medida que a célula se move ao longo do circuito;
- O CMD do relay pode ser:
  - data (para dados a serem comunicados na stream),
  - begin (para abrir nova stream), end (para fechar stream),
  - teardown (para fechar uma stream “estragada”),
  - connected (para notificar o OP que foi efectuado um begin com sucesso),
  - extend/extended (para extender o circuito por mais um OR),
  - truncate/truncated (para destruir apenas parte do circuito),
  - sendme (para controlo de congestionamento num OR), ou
  - drop (para testar stream);

2	1	2	6	2	1	498
CircID	Relay	StreamID	Digest	Len	CMD	DATA



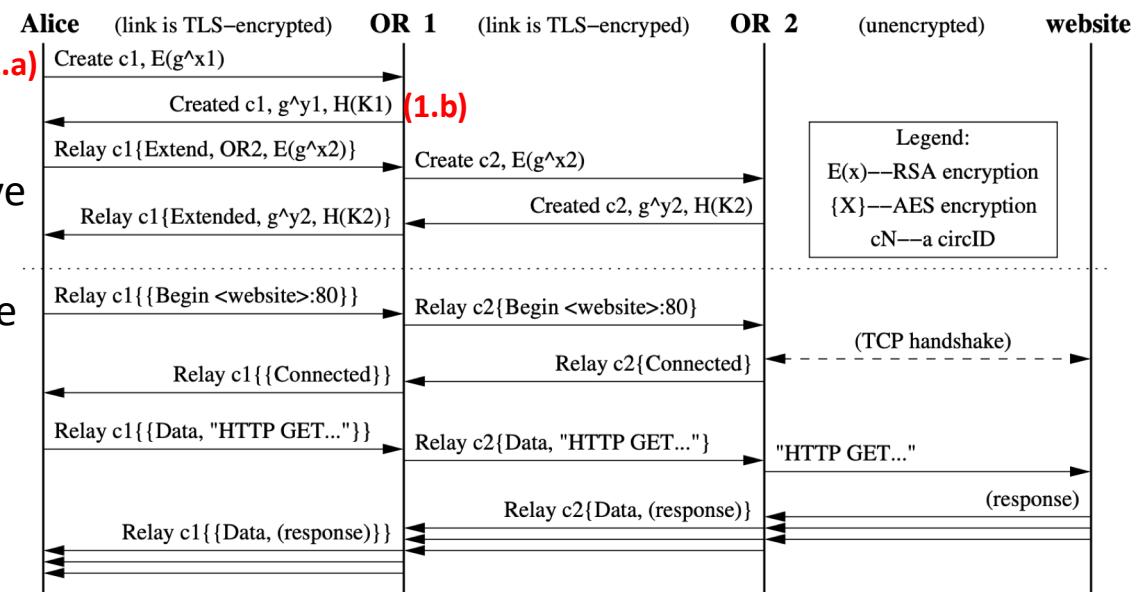
# TOR - Anonimização

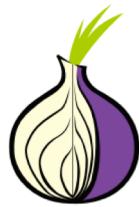
- OP pré-estabelece circuitos (normalmente de 3 OR) e muda para um novo circuito uma vez por minuto, garantindo que apenas um número limitado de pedidos podem ser ligados uns aos outros no OR de saída;
- OP constrói o circuito incrementalmente, negociando uma chave simétrica com cada OR do circuito, OR a OR. Note-se que escolhe os OR a partir da lista de OR fornecida pelo *Directory Server* que tem associado as chaves de longo termo (*identity key* para assinar certificados TLS) e de curto termo (*onion key* para estabelecer chaves de sessão por Diffie-Hellman).

Passo 1a: O OP (Alice) envia uma célula de controlo *create* para o primeiro nodo (OR1) do caminho escolhido pelo OP, escolhendo um novo circID e com o payload da célula contendo a primeira metade da troca de chaves Diffie-Hellman ( $g^{x1}$  cifrado com a chave pública da *onion key* de OR1). (1.a)

Passo 1b: O OR1 responde com uma célula de controlo *created*, contendo  $g^{y1}$  assim como a hash da chave  $K_1$  negociada ( $K_1 = g^{x1,y1}$ ).

A partir deste momento, OP e OR1 podem comunicar a célula de *relay* com o payload cifrado com a chave  $K_1$ .





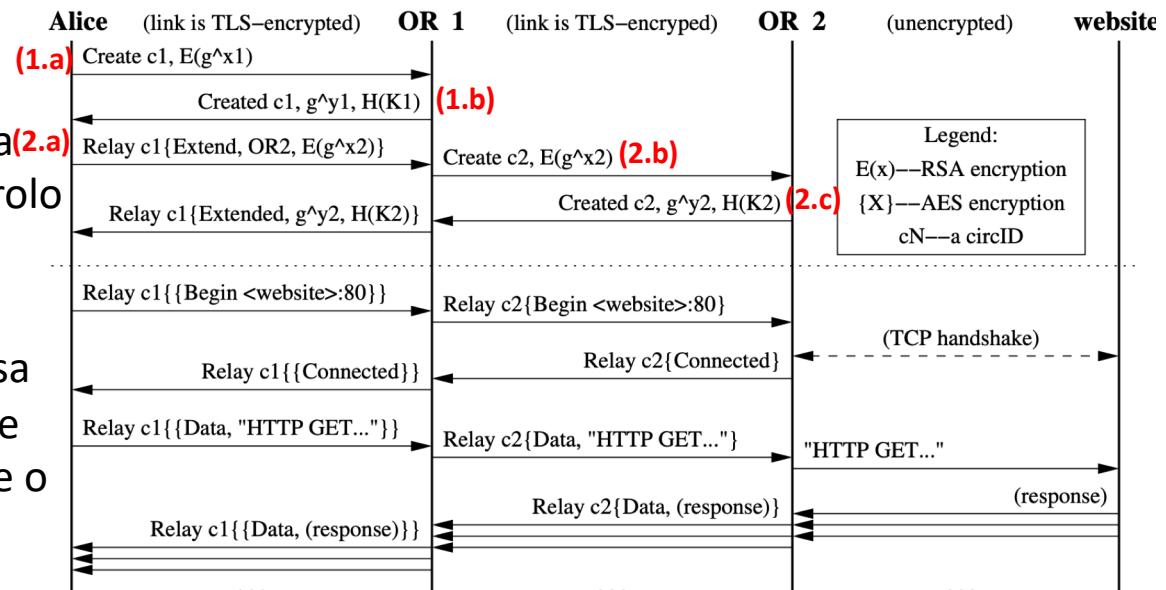
# TOR - Anonimização

Passo 2a: Para extender o circuito, o OP (Alice) envia uma célula de *relay extend* ao OR1, identificando o próximo OR (OR2) e com  $g^{x^2}$  cifrado com a chave pública da *onion key* de OR2 ( $E(g^{x^2})$ ).

Passo 2b: OR1 escolhe um novo CircID, copia(2.a)  $E(g^{x^2})$  para o payload de uma célula de controlo *create* e, envia-a a OR2.

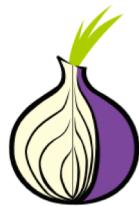
Passo 2c: OR2 responde com uma célula de controlo *created* e OR1 copia o payload dessa célula para uma célula de *relay extended* que envia a OP. O circuito está extendido a OR2 e o OP e OR2 partilham a chave comum  $K_2 = g^{x^2,y^2}$ .

A partir deste momento, OP e OR2 podem comunicar a célula de *relay* com o payload cifrado com a chave  $K_2$ .



Para extender o circuito para nodos adicionais ( $OR_{n+1}$ ), OP (Alice) efetua os passos anteriores, indicando sempre ao último OR ( $OR_n$ ) no circuito para extender ao novo OR ( $OR_{n+1}$ ).





# TOR - Anonimização

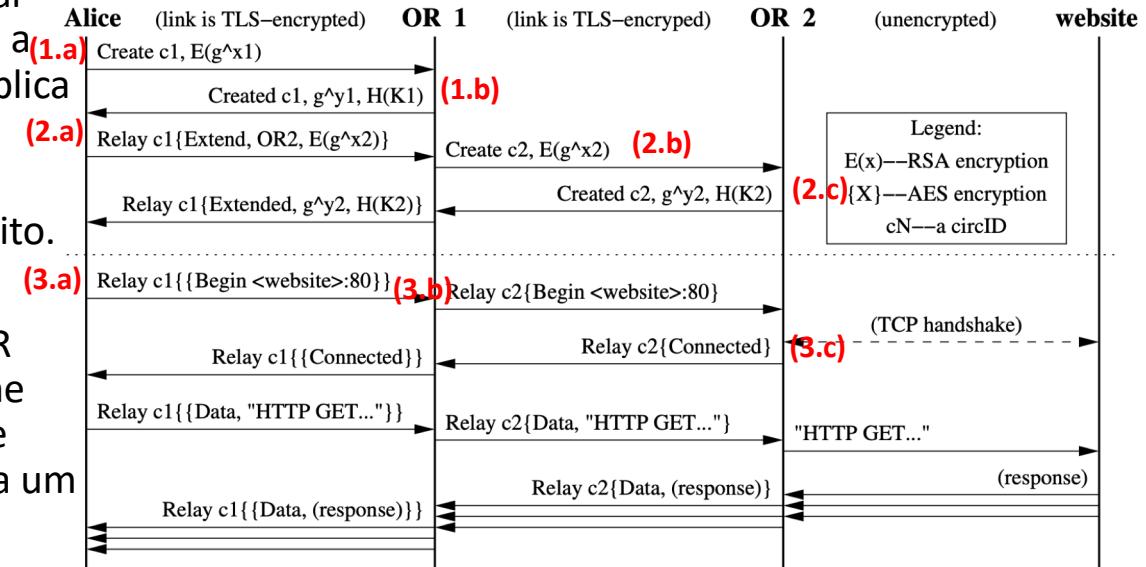
O Protocolo de estabelecimento do circuito garante autenticação unilateral (OP sabe que está a trocar chaves com o OR, mas o OR não sabe quem está a abrir o circuito – i.e., OP não usa a sua chave pública e mantém-se anónimo).

Assim que o circuito está estabelecido, OP pode enviar células de *relay* até ao último OR do circuito.

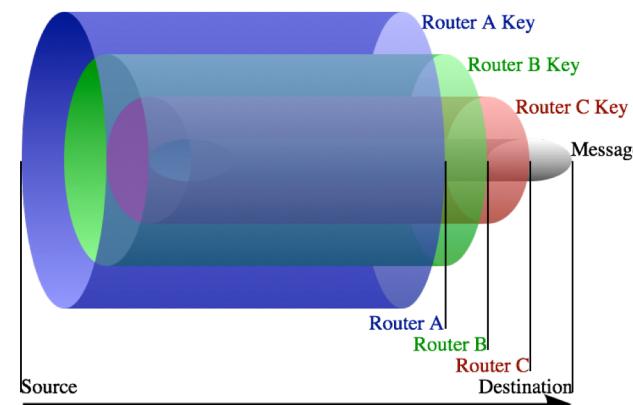
Passo 3a: OP (Alice) envia a célula de *relay* ao OR destinatário. Para construir essa célula, insere-lhe todos os dados necessários, gera o hash/digest e cifra com cada chave simétrica trocada com cada um dos OR (neste caso cifra com OR1 e depois com OR2).

Passo 3b: o primeiro OR (OR1) decifra o payload e verifica se o hash está correcto. Se estiver, efectua o comando pedido pelo OP. Se não estiver, pega no payload decifrado e envia uma célula de *relay* para o OR seguinte no circuito.

Passo 3c: O OR final responde, através do circuito estabelecido, com uma célula de *relay* com o payload cifrado para o OP. Os OR intermédios adicionam novos níveis de cifra à medida que a “reencaminham” para o OP.

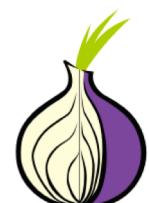


Condição de paragem, i.e., chegou ao OR de destino

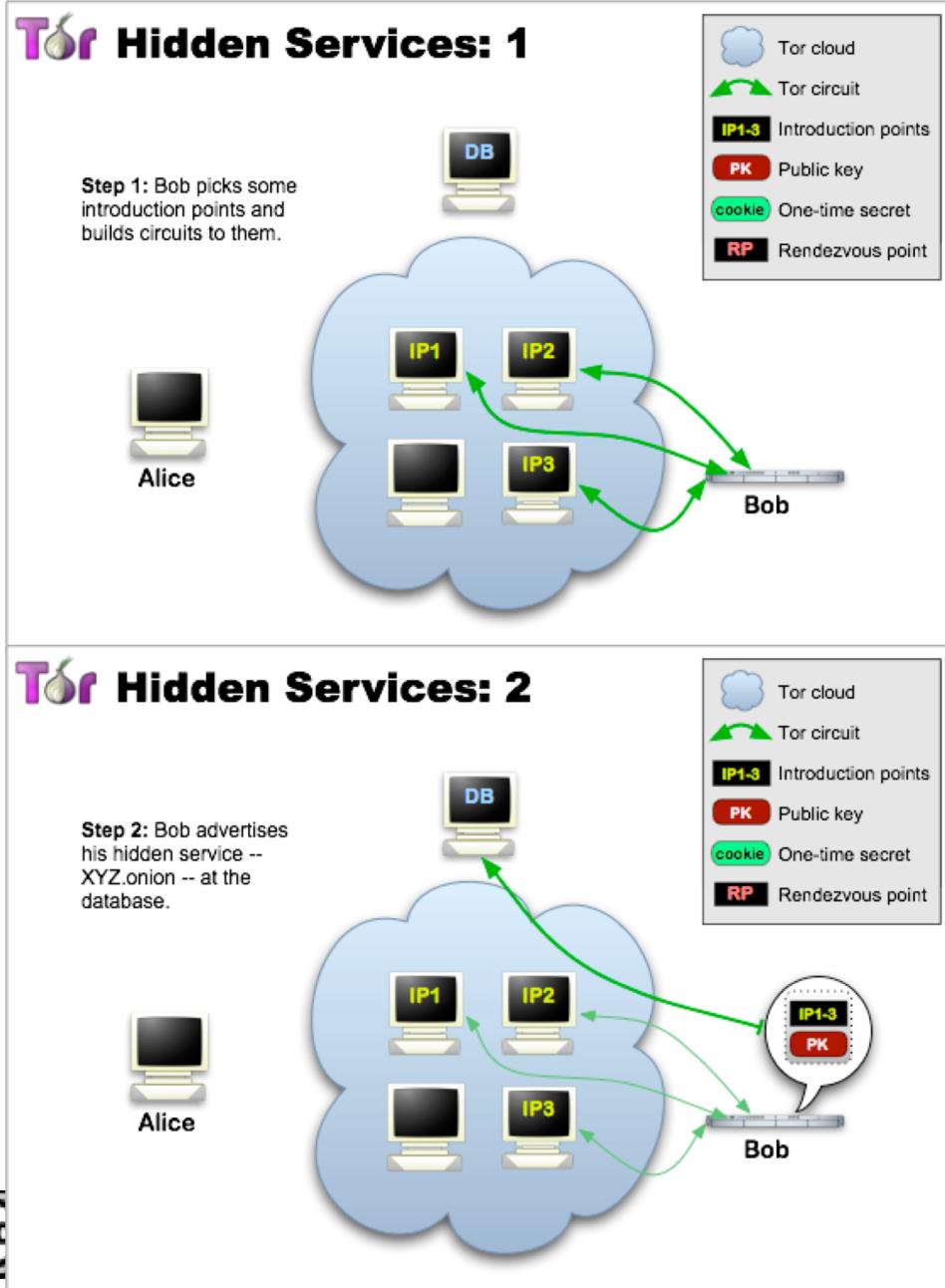


# TOR – Pontos de *Rendezvous* e serviços anónimos

- **Pontos de *Rendezvous*** são o suporte para a disponibilização de serviços anónimos (também designados por *responder anonymity*).
  - Na rede TOR, a disponibilização de **serviços anónimos** permite a um OP (Bob) disponibilizar serviços TCP (por exemplo, servidor web) sem revelar o seu endereço IP.
  - Como o OP (Alice) que acede ao serviço anónimo também é anonimizado, tanto o OP que acede como o OP que é acedido são anónimos.

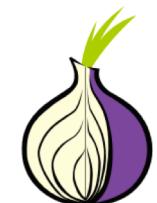


# TOR – Pontos de *Rendezvous* e serviços anónimos

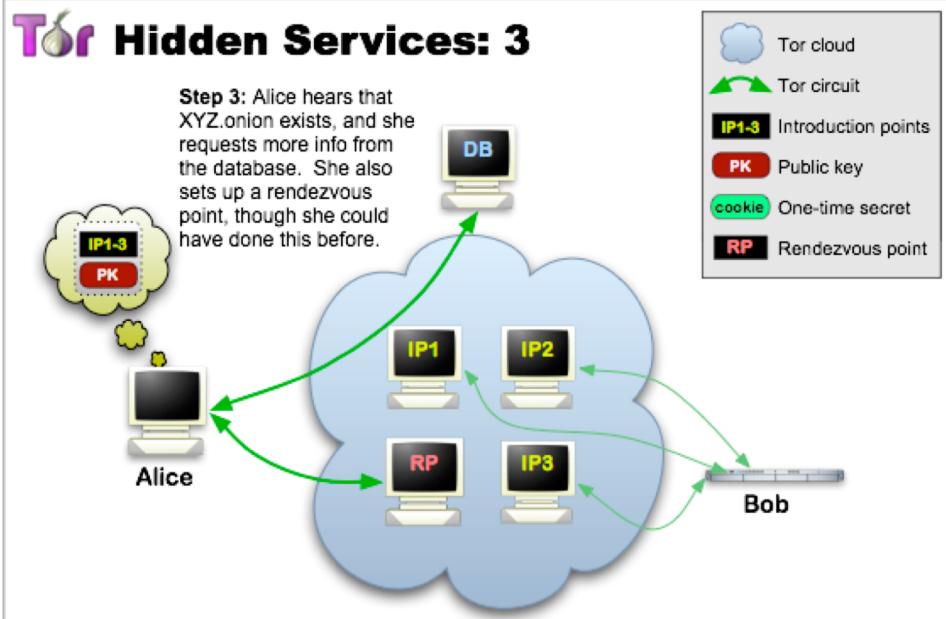


## Passos 1 e 2:

- Bob gera um par de chaves de longo tempo para identificar o seu serviço Web (a chave pública passa a ser o identificador do serviço);
- Bob escolhe alguns pontos de introdução (*introduction points*) e anuncia-os no *Directory Server*, assinando o anúncio (descriptor do serviço) com a sua chave privada;
  - O descriptor do serviço anónimo contém a chave pública do serviço e um sumário de cada ponto de introdução;
  - O descriptor/serviço será encontrado pelos clientes que acederem a XYZ.onion na rede TOR, onde XYZ é um nome de 16 caracteres derivado da chave pública do serviço.
- Bob cria um circuito TOR (conforme visto nos últimos slides) para cada um dos pontos de introdução e pede-lhes para esperarem por pedidos.



# TOR – Pontos de Rendezvous e serviços anónimos

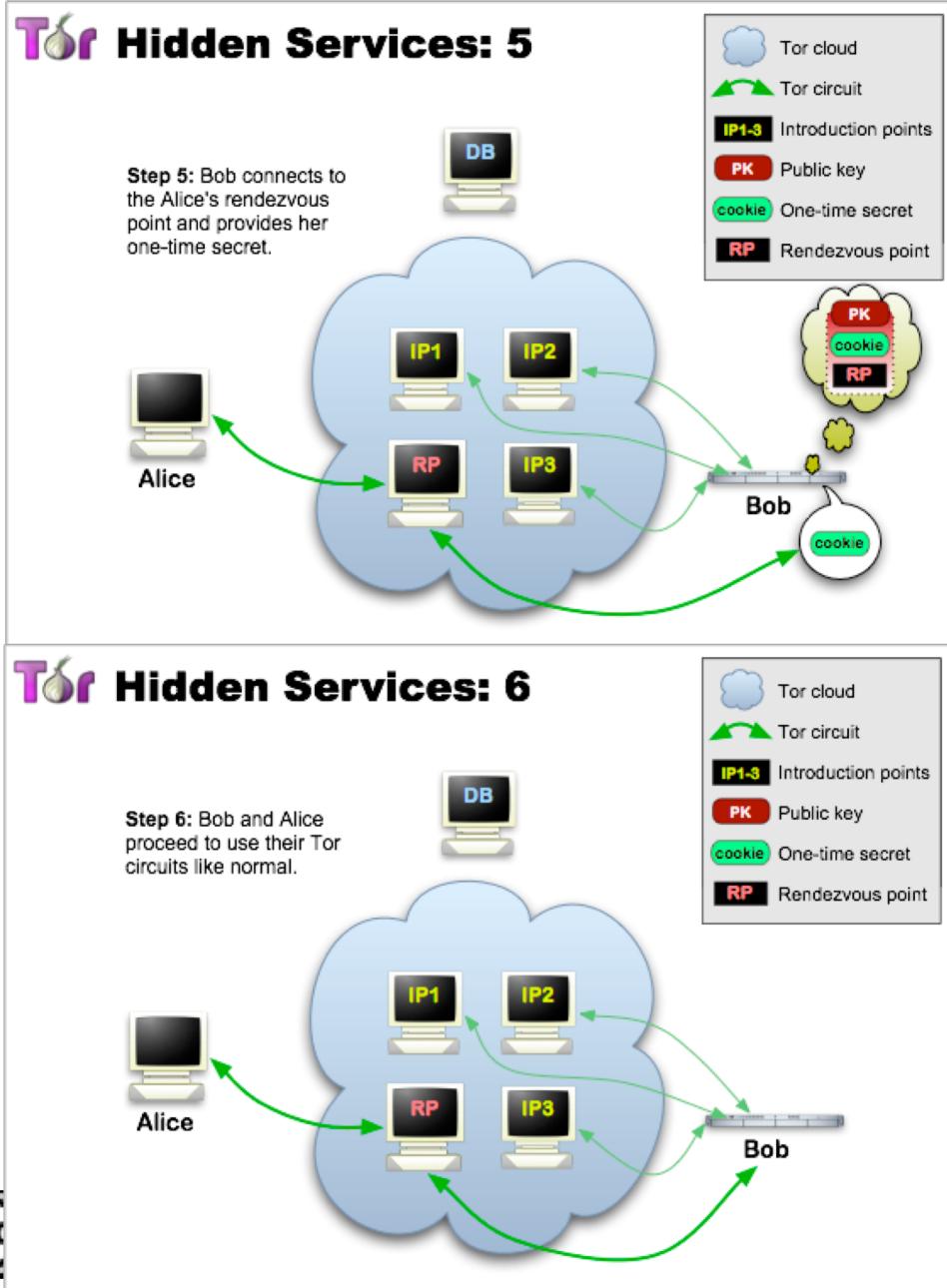


## Passos 3 e 4:

- Alice sabe da existência do serviço XYZ.onion e acede aos seus detalhes (chave pública e pontos de introdução) no TOR através do Directory Server;
- Alice escolhe um OR como ponto de *rendezvous* (RP) para a conexão com o serviço XYZ.onion;
- Alice constrói um circuito TOR até ao RP e fornece-lhe um “*rendezvous cookie*” (segredo aleatório único) para posterior reconhecimento do serviço XYZ.onion;
- Alice abre um *stream* anônimo até um dos pontos de introdução e fornece-lhe uma mensagem (cifrada com a chave pública de XYZ.onion) com informação sobre o RP, o “*rendezvous cookie*” e o inicio de troca de chaves Diffie-Hellman. O ponto de introdução reencaminha a mensagem para o serviço XYZ.onion através do circuito TOR criado nos passos anteriores.



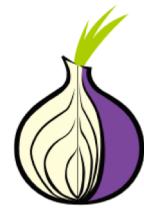
# TOR – Pontos de Rendezvous e serviços anónimos



## Passos 5 e 6:

- Se Bob (XYZ.onion) pretender falar com Alice (OP), Bob cria um circuito TOR até ao RP e envia o “rendezvous cookie”, a segunda parte da troca de chaves Diffie-Hellman e um *hash* da chave de sessão que agora partilha com Alice;
- O RP conecta o circuito de Alice com o circuito de Bob (normalmente o circuito consiste de 6 OR: 3 escolhidos por Alice sendo o terceiro o RP e, outros 3 escolhidos por Bob). Note-se que o RP não consegue reconhecer Alice, Bob nem os dados que transmitem;
- Alice envia uma célula de *relay begin* através do circuito que, ao chegar ao OP de Bob, conecta com o serviço disponibilizado por Bob (por exemplo, um servidor Web);
- Um *stream* anónimo foi estabelecido e Alice e Bob comunicam da forma normal num *stream* TOR.





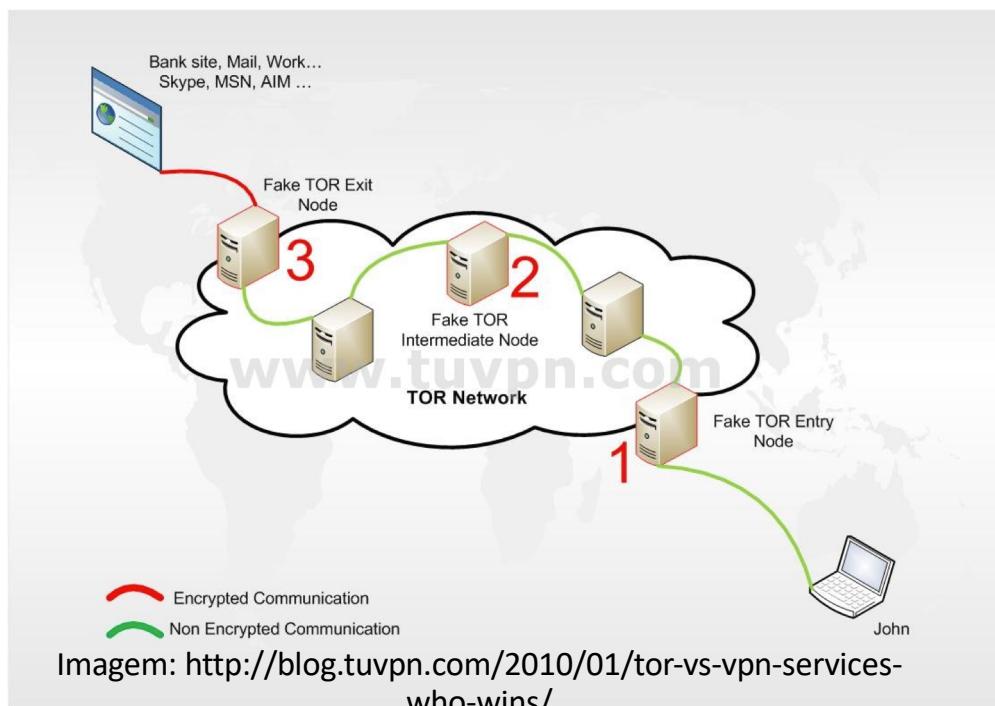
# TOR – Possíveis Ataques

**O que acontece se o nosso servidor actuar como nodo de entrada?**

- Sabe que o IP de John está a utilizar a rede TOR (interessante para SIGINT – *signal intelligence* – mas pouco mais).

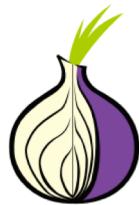
**O que acontece se o nosso servidor actuar como nodo intermédio?**

- Nada. Recebe informação cifrada de um nodo TOR que vai para outro nodo TOR.



**O que acontece se o nosso servidor actuar como nodo de saída?**

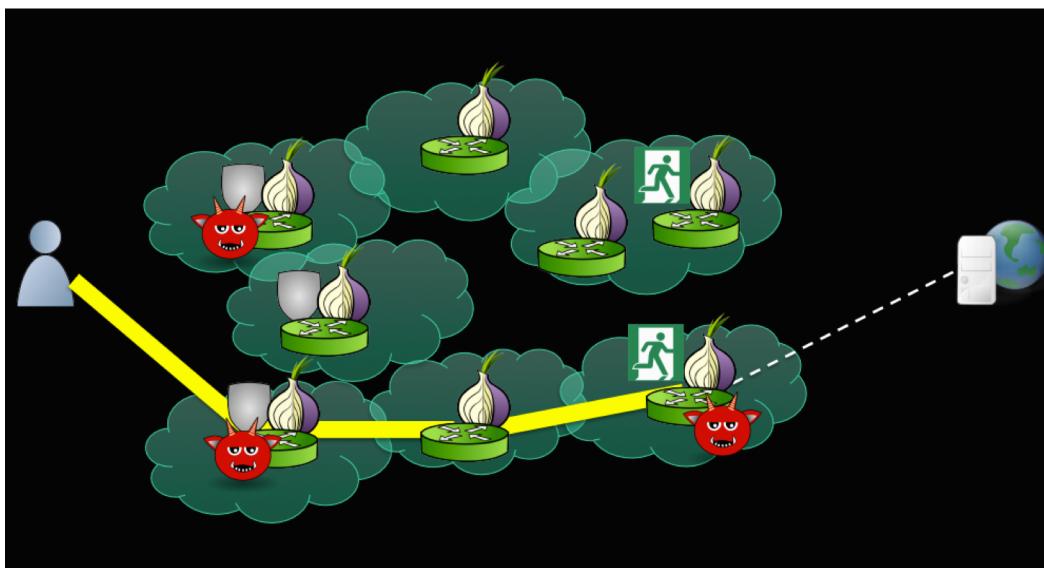
- A missão do nodo de saída é decifrar a comunicação, e enviá-la para o serviço (web) de destino, pelo que não sabe quem foi o originante da comunicação mas sabe qual é a comunicação.
- Podemos argumentar que se o destino é SSL, não há hipótese de aceder à comunicação em claro do John.
- Errado. Foi demonstrado que é possível um *MitM attack* sobre o SSL, no nodo de saída.



# TOR – Possíveis Ataques

O que acontece se um servidor nosso actuar como nodo de entrada e outro servidor nosso actuar como nodo de saída?

- É possível calcular coeficientes de correlação – através de fórmulas matemáticas da área da probabilidade e estatística – na análise de pacotes nos dois nodos, baseado na frequência, timing e tamanho do pacote.
- Supõe-se que 80% dos utilizadores podem ser de-anonimizados no período de 6 meses.



- Custo elevado, já que uma grande percentagem dos OR (TOR tem cerca de 7.000 OR) têm que pertencer à entidade que queira efetuar um ataque de correlação.
- Utilizado por agências governamentais (BND, GCHQ, NSA).
- Supõe-se que o “Silk Road 2.0” foi de-anonimizado com base neste ataque.

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





# TOR

- Exemplo efectuado em menos de dez minutos, a partir da mesma máquina, no site [www.whatismyip.com](http://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

## 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

- Qual a explicação?

