Atividade de Segurança de Redes II

Thiago Teodoro Pereira Silva

Centro Federal de Educação Tecnológica Celso Suckow da Fonseca - CEFET/RJ

18 de agosto de 2021

Conteúdo

Introdução

- A rede Tor é uma rede de sobreposição (*overlay network*).
- O usuário roda localmente um *proxy Onion* responsável por montar circuitos, e gerenciair conexões vindas da camada de aplicação. Essa interface é feita via *SOCKS*.
- Um roteador Onion matém uma conexão TLS com outros roteadores e com *proxies* Onion.
- A rede só trabalha com TCP.

Conteúdo

Estrutúra de uma célula I

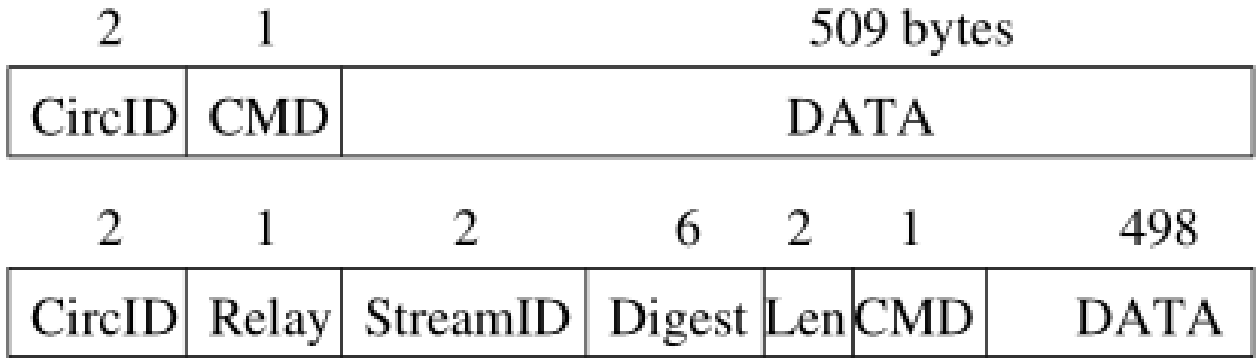


Figura: Estrutura de uma célula. A segunda corresponde à uma célula de Relay.

- Cada célula possui um cabeçalho e um *payload*;
- O campo *CircID* indica a qual circuito essa célula se refere;
- E *CMD* indica o que fazer com o *payload*;
- Dependendo do comando, células podem ser tanto de controle quanto de *relay*;
  - Células de controle são interpretadas no nó que recebe elas;
  - Células de *relay* carregam dados de ponta a ponta.

Estrutúra de uma célula II

Comandos de controle podem ser:

- *padding*;
- *create* e *created* (para criar um cricuito)
- *destroy* (para destruir um circuito)

Células de *relay* possuem um cabeçalho adicional:

- *StreamID* (muitos fluxos TCP podem ser multiplexados em um mesmo circuito);
- Um *checksum* para checagem de integridade;
- O tamanho do *payload*
- e um comando.

Estrutúra de uma célula III

Os comandos de *relay* são:

- *relay data*, para dados sendo transmitidos em um fluxo;
- *relay begin* para abrir um fluxo;
- *relay end* para fechar um fluxo;
- *relay extend* e *relay extended* para extender o circuito em um salto.
- e outros...

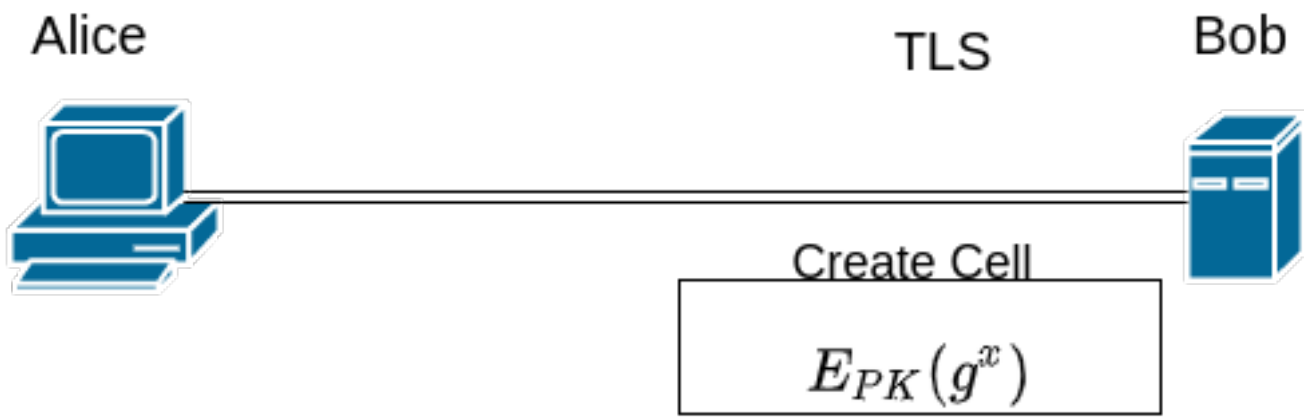
Conteúdo

Criando um novo circuito I

O *Onion Proxy* do usuário constroi um cicuito incrementalmente, negociando uma chave simétrica com **OR** no circuito, um salto por vez.

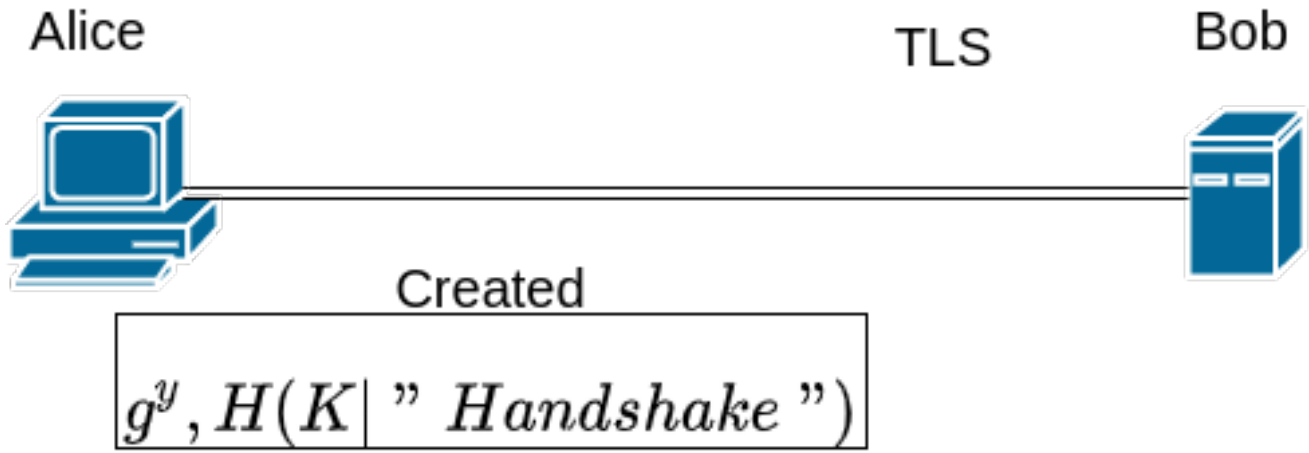
Criando um novo circuito II

- Alice envia à Bob uma célula *create*
- Escolhe um novo *circID* C_{AB}
- O *payload* da célula possui a primeira metade *Diffie-Helman Handshake* (g^x)

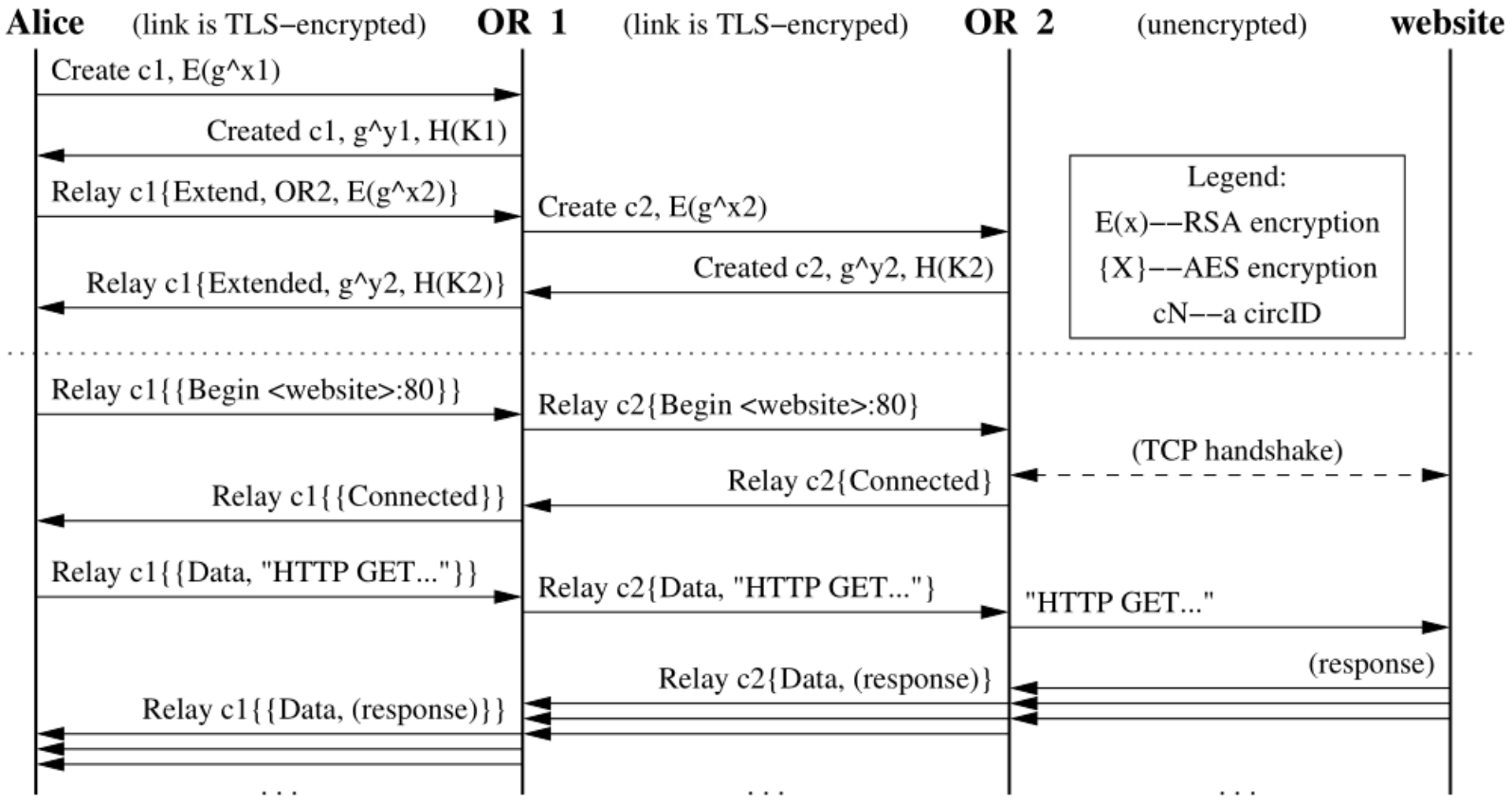


Criando um novo circuito III

- Bob então responde com uma célula *created* contendo g^y junto com um *hash* da chave negociada g^{xy}.



Extendendo um cicuito I



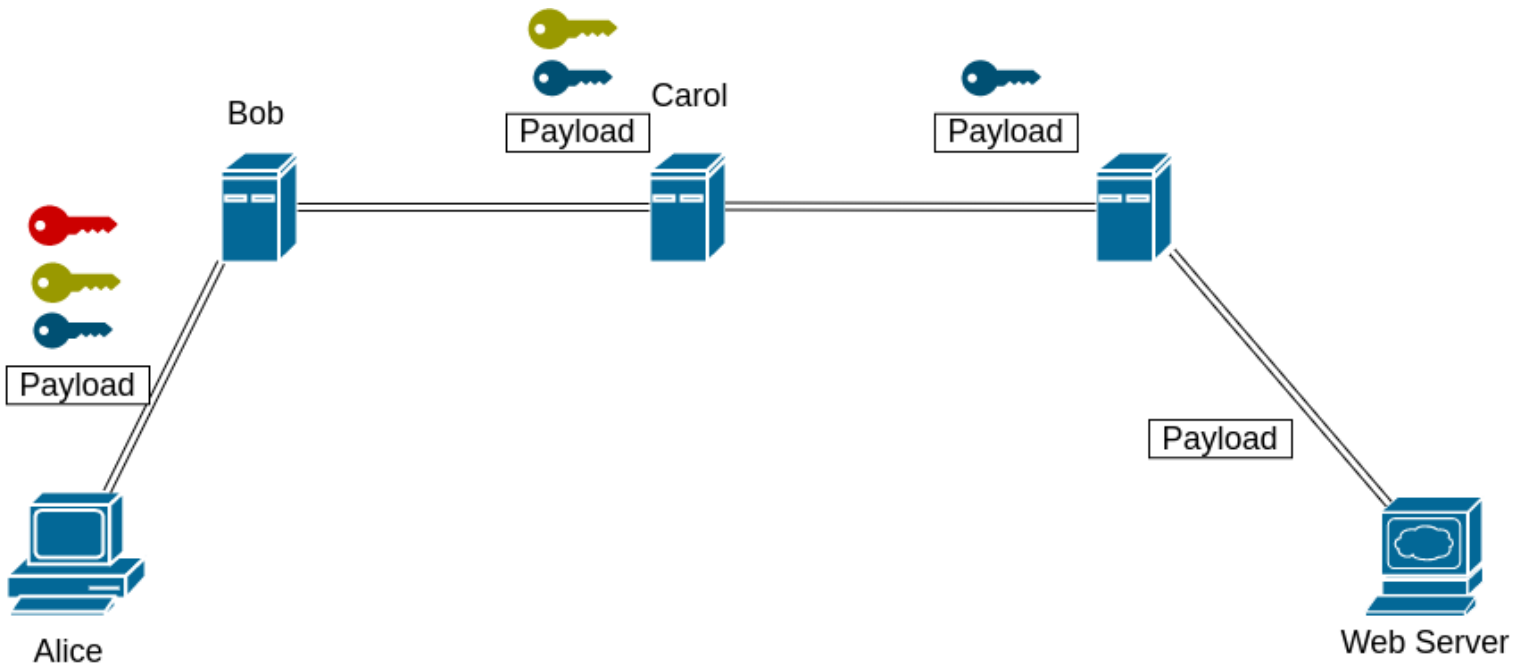
Extendendo um cicuito II

Para extender o cicuito...

- Alice envia uma célula *relay extend* para Bob, especificando o endereço do novo OR (Carol);
- A célula contém um g^{x2} criptografado para Carol;
- Bob então copia o *half-handshake* em uma célula *create* e envia para Carol.
  - Bob escolhe um novo *circID* C_{BC} para identificar sua conexão com a Carol;
  - Somente Bob associa o circuito C_{AB} com C_{BC}.
- Quando a Carol responde com uma célula *created*, contendo g^{y2}, Bob encapsula o *payload* em um *relay extended* en envia para Alice;
- Agora Alice e Carol compartialham uma chave simétrica g^{x2y2} só conhecida por elas.

Conteúdo

Células de Relay I



Células de Relay II

Quando um OR responde Alice com uma célula *relay*, ele criptografa o cabeçalho e o *payload* com a chave que ele compartilha com Alice, e então envia para o próximo nó no circuito. Subesequentes ORs acrescentam mais camadas de criptografia.

Conteúdo

Abrindo e Fechando Fluxos I

Quando uma aplicação da Alice quer abrir uma conexão TCP, ela pede ao OP (via *SOCKS*) para fazer a conexão. O OP então escolhe o circuito mais novo (ou cria um) e escolhe algum OR como sendo o nó de saída.

Abrindo e Fechando Fluxos II

O OP então abre o fluxo TCP enviando um *relay begin* para o nó de saída, usando um novo e aleatório *streamID*. Quando o nó de saída faz a conexão, ele responde com uma célula *relay connected*. Agora o OP aceita dados do fluxo TCP da aplicação, enpacotando dentro de células *relay data* e enviando alongo do circuito para o nó de saída.

Abrindo e Fechando Fluxos III

Vazamento de DNS

- Algumas aplicações passam o *hostname* para o SOCKS enquanto outras resolvem primeiro em um endereço IP para em seguida enviar para o SOCKS;
- Se a aplicação resolve o *hostname*, o destino da comunicação será revelado para o servidor DNS;
- Além disso, endereços *.onion* precisam ser passados para a interface SOCKS para serem resolvidos;

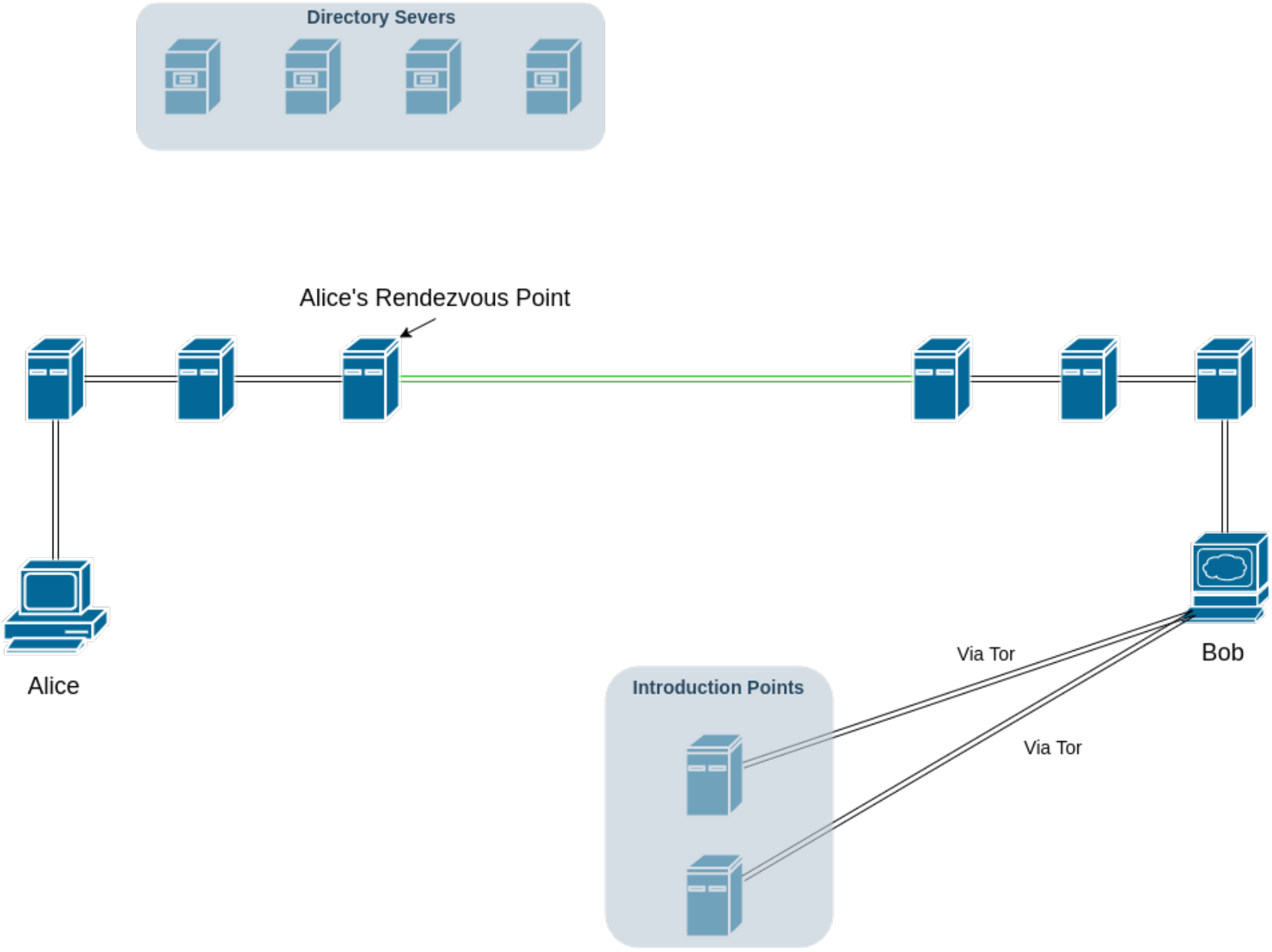
Conteúdo

Visão Geral

Suponha que Bob queira oferecer um serviço sem revelar a sua origem:

- Ele apresenta diversos ORs, que serão seu pontos de introdução, como pontos de contato;
- Alice, a cliente, escolhe um OR como seu ponto de encontro (*rendezvous point*);
- Ela conecta à algum dos pontos de introdução de Bob,
  - informa ele de seu ponto de encontro,
  - e espera ele se conectar ao seu ponto de encontro.

Serviços ocultos na rede Tor



Serviços ocultos na rede Tor

- Bob configura no seu OP o endereço local e porta de seu serviço (ou o local de seu *unix socket*);
- O OP publica as informações da chave pública de Bob e seus pontos de introdução no sistema de *lookup*;
  - O lookup é indexado pelo *hash* de sua chave pública.
- Domínios são da forma *x.y.onion*, onde **x** é o cookie de autorização (se estiver sendo usado) e **y** é o *hash* da chave pública de Bob;

Conteúdo

Referências


DINGLEDINE, Roger; MATHEWSON, Nick; SYVERSON, Paul. Tor: The Second-Generation Onion Router. In: PROCEEDINGS of the 13th USENIX Security Symposium. [S.l.: s.n.], ago. 2004.