Alma Mater Studiorum · Università di Bologna

Scuola di Scienze
Corso di Laurea in Matematica

ONION ROUTING E ANONIMATO: IL SISTEMA TOR

Tesi di Laurea in Crittografia

Relatore:
Chiar.mo Prof.
Davide Aliffi

Presentata da:
Filippo Chini

Sessione Unica
2017/2018

Indice

Introduzione

La riservatezza

Il concetto di riservatezza, o privacy, significa assicurarsi che le proprie risorse siano accessibili solo alle parti autorizzate. In rete, senza accorgimenti, non esiste la privacy!
Infatti, tutte le attività su Internet sono rintracciabili e identificabili: la nostra navigazione online lascia una sorta di impronta digitale nel web, che può essere usata per raccogliere dati e quindi spiare il nostro uso di Internet.
La crittografia è un'elementare risposta a questo problema, ma essa a un livello base permette soltanto una protezione sui contenuti (per esempio messaggio o mail), mentre lascia all'avversario la possibilità di ricavare origine, destinazione, quantità e tipologia di dati che trasmettiamo.
Come fare quindi a non lasciare nessuna traccia di un nostro passaggio nella rete?

La soluzione: TOR

Il sistema TOR (acronimo di The Onion Router) è una risposta a questo problema, cioè è uno strumento in grado di creare una comunicazione anonima e riservata.
Supponiamo che Alice e Bob debbano scambiarsi dei dati: usando TOR, i due si tutelano dal fatto che qualcuno possa intercettare questa comunicazione, cioè si proteggono dai tentativi di *analisi del traffico*. In particolare non solo proteggono il contenuto, ma rendono questo loro contatto invisibile agli occhi della rete: nessuno (a parte loro) sa che sia avvenuta questa comunicazione!
Lo scopo di Tor è proprio quello di rendere difficile (anche se non del tutto impossibile) l'analisi dei flussi di dati nella rete e proteggere così la riservatezza delle comunicazioni e l'accessibilità ad alcuni servizi. Il funzionamento della rete Tor è concettualmente semplice: i dati che appartengono ad una qualsiasi comunicazione non transitano direttamente dal client al server, ma

passano attraverso i nodi Tor che agiscono da router costruendo un circuito virtuale crittografato a strati (per analogia con onion, che in inglese significa cipolla). Oggi Tor è un software libero, disponibile per tutti i sistemi operativi e per tutte le persone che desiderino una maggior protezione dei propri dati, ma nasce per scopi militari, come mezzo di antispionaggio.

Un po' di storia

Il nucleo di Tor fu sviluppato a metà degli anni novanta dal matematico Paul Syverson e da Micheal Reed per la US Naval Research Laboratory (la Marina Militare degli Stati Uniti), allo scopo di garantire che le conversazioni governative (ordini e disposizioni d'impiego) non fossero intercettate da entità nemiche o da servizi d'intelligence stranieri.
Nel 2002 la Marina rilascia il codice a due programmatori di Boston: Roger Dingledine, ex dipendente dell'NSA e Nick Mathewson. Costoro ne continuano pubblicamente lo sviluppo fino al 2004 quando, insieme a Syverson, presentano ufficialmente TOR. Nel Dicembre 2004 il progetto TOR viene adottato e sponsorizzato dall'Electronic Frontier Foundation.
Dingledine e Mathewson e altri 5 collaboratori che lavoravano originariamente al progetto, nel 2006 fondano The Tor Project, un'associazione senza scopo di lucro responsabile dello sviluppo di Tor e dei progetti correlati. La EFF continua ad essere il principale sponsor del progetto.

A chi è rivolto Tor?

Tor è rivolto a tutti quelli che vogliono tenere nascoste le proprie attività nella rete. Infatti, come si è già detto, la nostra navigazione online lascia tracce in rete: il nostro indirizzo IP, ad esempio, può fornire la nostra localizzazione e una precisa identificazione a chi sta osservando le nostre azioni. Questo succede perché sempre più persone usano Internet e la gamma di cose che possiamo fare si estende sempre di più. Di conseguenza, aumentano i dati che viaggiano in questo mondo virtuale dando a chi li gestisce il potere di influenzare la nostra esperienza in rete: annunci pubblicitari personalizzati o i suggerimenti nelle nostre ricerche sono solo due esempi di come questo possa essere attuato. Per tutelarsi da questo pericolo, l'uso di Tor risulta essere una buona contromisura.
Come si è già accennato, Tor nasce per nascondere comunicazioni degli agenti in missione visto che in territori ostili il controllo su certi strumenti informa-

tici è forte.
Un altra categoria che può beneficiare dell'uso dell'Onion Routing è quella dei giornalisti, i quali possono comunicare con le loro fonti in maniera perfettamente anonima evitando fughe di notizie, oppure possono pubblicare online articoli "scomodi" per alcuni governi o aziende senza timore di venire scoperti o censurati (un famoso esempio è Wikileaks). Tor è anche uno strumento fondamentale per la protezione delle fonti di informazione, che senza l'anonimato potrebbero correre rischi (i cosiddetti *whistleblowers*).
Infine, all'interno di un'azienda o di una grande società può essere necessario nascondere le risorse utilizzate per non far conoscere ai concorrenti le proprie strategie.

Capitolo 1

La tecnologia

Tor aiuta a ridurre i rischi derivati dall'analisi del traffico, distribuendo le transazioni attraverso molti nodi della rete Internet, in modo che nessun singolo punto possa collegare una transazione alla sua destinazione. L'idea è quella di costruire un percorso tortuoso e difficile come se si dovesse depistare un inseguitore, cancellando periodicamente le proprie orme. Infatti, anziché prendere un percorso diretto dalla sorgente alla destinazione, i pacchetti di dati nella rete Tor prendono un percorso casuale attraverso nodi particolari che ne coprono le tracce, in modo che nessun osservatore situato in un singolo punto possa stabilire l'origine e destinazione di un certo traffico.
Per creare un percorso di rete privato con Tor, l'utente esegue un software locale chiamato *Onion Proxy* (OP), il quale aggiungendo un nodo alla volta in maniera incrementale, costruisce un circuito di connessioni cifrate attraverso i server della rete TOR, detti *Onion Router* (OR). Il software negozia delle chiavi crittografiche per ogni aggiunta al circuito così quando deve mandare dei dati, questi vengono cifrati più volte con le chiavi di ogni nodo attraversato nel circuito: ogni nodo rimuove uno strato e inoltra al nodo successivo. Questo è un modo per assicurarsi che ciascun nodo coinvolto venga a conoscenza soltanto dei due nodi adiacenti, nessun server conosce il percorso completo. Questa caratteristica permette anche al sistema di essere resistente a intercettazioni o compromissioni di nodi, in quanto non ci sono punti che racchiudono la totalità dell'informazione. Una volta che un circuito è stato stabilito, si possono scambiare diversi tipi di dati con la garanzia di assoluto anonimato.

1.1 Protocollo Tor

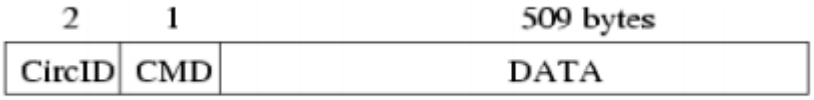
1.1.1 Autenticazione

Prima di iniziare qualsiasi operazione il client OP si connette al directory server, un server che memorizza tutti le informazioni sui nodi onion della rete, e scarica la lista dei nodi OR disponibili con le rispettive chiavi di identificazione. Queste chiavi di identificazione altro non sono che le "chiavi pubbliche" di in un sistema di cifratura asimmetrico, infatti come vedremo più avanti TOR si avvale della crittografia a chiave pubblica in un passo del suo protocollo.

La connessione TLS tra OP-OR oppure OR-OR non è permanente, una delle due parti coinvolte nella comunicazione può interrompere la connessione se non transitano dati per un certo periodo di tempo (chiamato *KeepAlivePeriod*), impostato di default a 5 minuti. Per ragioni di efficienza Tor utilizza lo stesso circuito per connessioni che avvengono nell'arco di dieci minuti. Le richieste successive vengono instradate su un nuovo circuito, per evitare possibili collegamenti con le prime.

1.1.2 Celle e comandi

Una volta stabilita una connessione tra le due entità (OP-OR oppure OR-OR) esse incominciano a scambiarsi dati in celle di lunghezza fissa, che vengono inviate in maniera sequenziale. In questo modo si impedisce ogni attacco che si basi sulla correlazione tra lunghezza e natura dei dati.
Ogni cella ha una taglia di 512 bytes e il suo formato è sempre di questo tipo:



Il **CircId** e il **Command** (CMD) formano il cosidetto header della cella.
Il primo è l'informazione che specifica a quale circuito la cella appartiene, visto che possono essere aperti più circuiti sulla stessa connessione.
Il secondo descrive l'operazione da fare con i dati contenuti nel payload.
Il **Payload** contiene i dati che devono essere trasmessi all'altro nodo, il suo contenuto può variare a seconda del Command.
Classificando in base al *command*, le celle si distinguono in **celle di controllo** e **celle di relay** .

Le celle di controllo trasportano dati tra nodi adiacenti e vengono usate per gestire e controllare il circuito. I possibili comandi sono:

- PADDING, viene utilizzata per implementare il *keepalive* della connessione: viene inviata una cella di questo tipo ogni volta che tra OP e OR non transitano dati da oltre 5 minuti (che è il tempo di default del sistema). Il payload di questa cella è inutilizzato;
- CREATE, serve per inizializzare il processo di creazione del cicuito.
- CREATED, è la cella di notifica che conferma l'avvenuta creazione del circuito
- DESTROY, è usata per eliminare il circuito.

Le celle di relay possono essere inviate solo quando il circuito è "consolidato", cioè il client possiede una chiave condivisa (*session key*)con ogni nodo del circuito creato. La loro funzione è quella di mandare "ordini" e far fare determinate azioni ai nodi del circuito. Il loro payload contiene un header aggiuntivo (*relay header*) strutturato in:

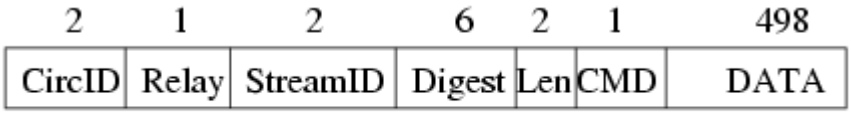
- un comando di relay (*Relay Command*);
- uno *StreamId*, per identificare il giusto canale di comunicazione;
- un *Digest*, che è l'informazione su quale nodo debba processare il payload.
- la lunghezza del relay payload (*Length*);

I possibili comandi che si possono inserire in una cella di relay sono:

- RELAY BEGIN, viene utilizzato per aprire un canale sul circuito;
- RELAY END, per chiudere un canale aperto;
- RELAY DATA, indica che il payload contiene informazioni da trasmettere sul canale aperto;
- RELAY CONNECTED, per notificare all'OP che il comando RELAY BEGIN è stato eseguito con successo;
- RELAY EXTEND, serve a ordinare di aggiungere un certo nodo al circuito;

- RELAY EXTENDED, conferma l'avvenuta estensione del circuito;
- RELAY TEARDOWN, inviato quando si vuole chiudere un canale interrotto improvvisamente;
- RELAY TRUNCATE, serve a interrompere solo parte del circuito, ma tenendolo aperto;
- RELAY TRUNCATED, la sua notifica di corretta interruzione;
- RELAY SENDME, utilizzato per il controllo della congestione.

La struttura della cella relay è in conclusione fatta in questo modo:



1.1.3 Crittografia a cipolla

La crittografia alla base del meccanismo TOR è un ibrido tra quella simmetrica e quella a chiave pubblica. Vediamo ora come si combinano tra loro. Per aggiungere un nuovo nodo al circuito, il client esegue un handshake Diffie-Hellmann con il nodo OR selezionato, occultando questo scambio con una cifratura a chiave pubblica. La chiave pubblica del nodo OR (detta *onion key*) viene recuperata dal directory server, che è una specie di grande rubrica con tutte le chiavi pubbliche dei nodi della rete TOR. Per garantire l'integrità della chiave, nella seconda parte dell'handshake (la risposta del nodo OR interessato) viene inserito anche un hash della chiave ricavata, in modo che con un semplice controllo l'utente sia effettivamente sicuro di possedere la stessa chiave dell'OR aggiunto. Si osservi che l'utente non usa una sua chiave pubblica, e quindi questo meccanismo raggiunge un livello di conoscenza unilaterale: il client OP sa con che server sta effettuando lo scambio DH, ma l'OR interessato non conosce chi sta aprendo o estendendo il circuito!
Una volta che il client possiede una chiave condivisa con tutti i nodi del circuito, può procedere a inviare dati (tramite le celle di relay) sfruttando la maggior efficienza della cifratura simmetrica. Fondamentalmente l'OP procede alla cifratura del payload di una cella relay con le session keys di ogni nodo che l'informazione deve superare, da quella del più lontano a quella del

più vicino. In pratica dal nodo destinatario al primo nodo della rete costruita. I nodi, al passare della cella, rimuoveranno il loro "strato" di cifratura e così il messaggio arriverà in chiaro dall'altra parte del circuito. Da questa caratteristica "a strati" deriva appunto il nome "ONION ROUTING".
Gli schemi di cifratura usati da TOR sono:

- RSA con chiave da 1024 bit ed esponente fisso 65537 per la cifratura a chiave pubblica;
- AES a 128 bit in modalità Counter per la crittografia simmetrica;
- per Diffie Hellmann si usa come esponente $g = 2$ mentre come modulo un numero primo di 1024 bit;
- come funzione hash si usa SHA3-256.

1.2 Esempio di gestione di un circuito TOR

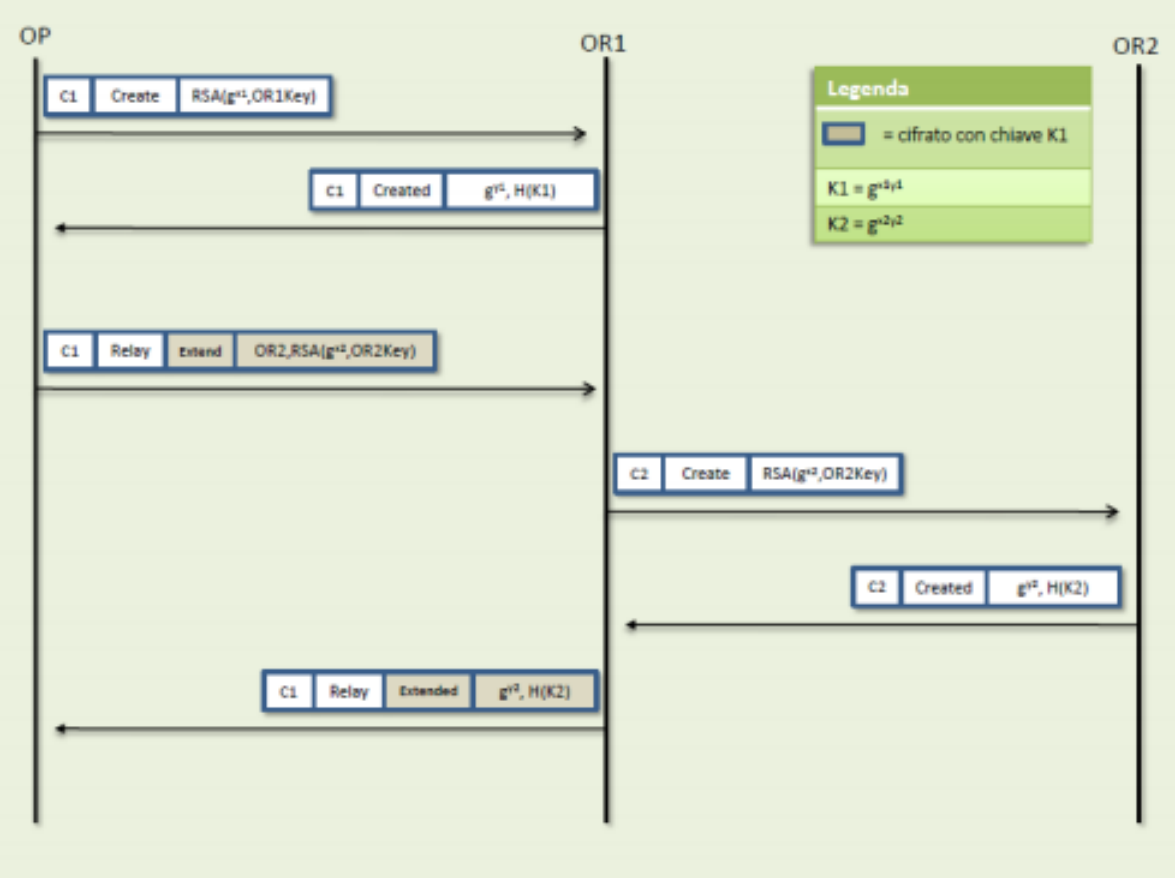
1.2.1 Creare un circuito

Per prima cosa, l'OP manda una cella CREATE al primo nodo da lui scelto (chiamiamolo OR1). Il CircId (C1) sarà un intero arbitrario di 2 byte scelto dal nodo mittente (nel nostro caso OP). Il payload della cella spedita conterrà la prima parte dell'handshake Diffie Hellmann g^{x1} cifrato con l'onion key (che ricordiamo è la chiave pubblica) di OR1;
Il nodo OR1, dopo aver decifrato la cella con la propria chiave privata, risponde con una cella CREATED. Il payload di quest'ultima conterrà la seconda parte dell'handshake DH g^{y1} insieme a un hash della chiave negoziata $K1 = g^{x1y1}$, che indicheremo con H(K1).
Questo accorgimento funziona da checksum, ossia permette a OP di verificare l'integrità di K1. Infatti il client dopo aver ricavato la chiave K1, ne calcolerà l'hash e confrontando con il dato ricevuto H(K1), verificherà di essere in possesso della stessa chiave K1. In caso non coincidano, la connessione viene interrotta.

1.2.2 Estendere il circuito

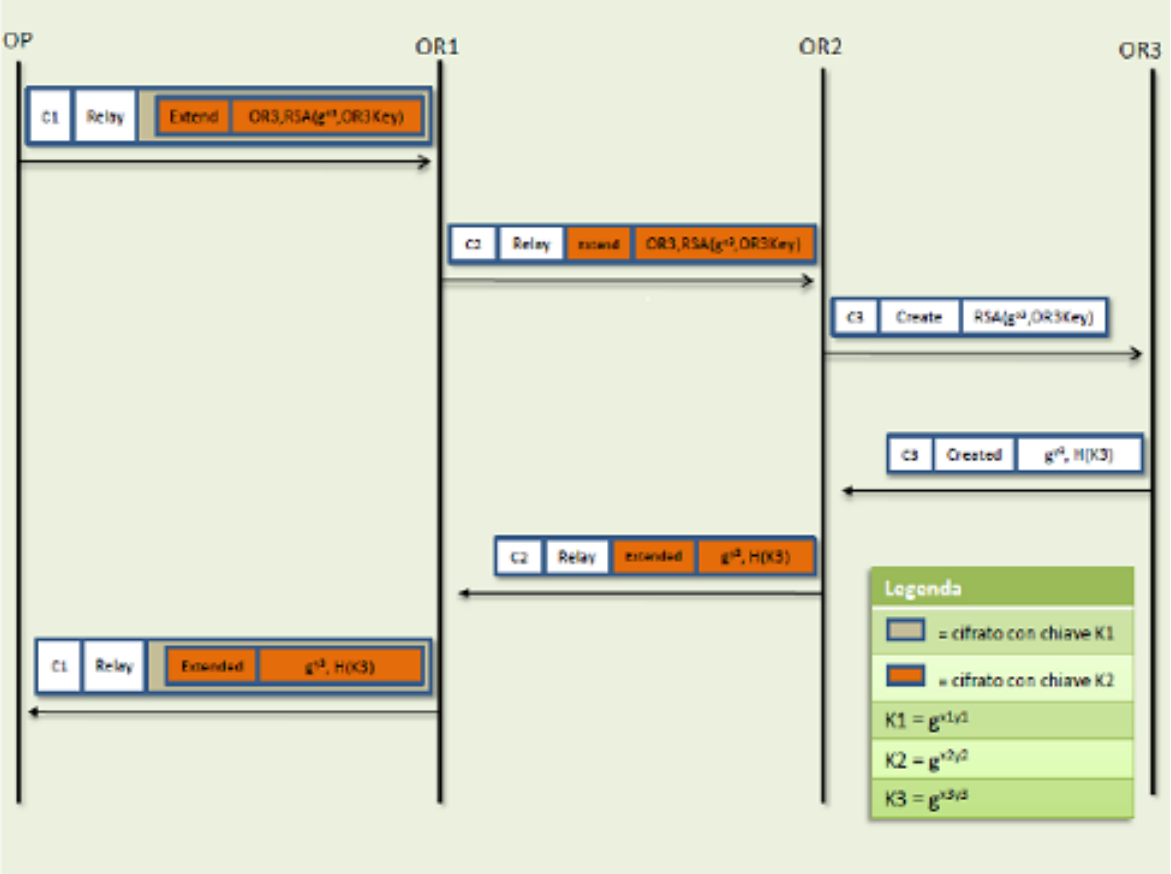
Per estendere il circuito, l'OP manda una cella RELAY EXTEND a OR1, cifrando con K1 il payload della cella totale, cioè il Relay header e il Relay payload. L'informazione contenuta è l'indirizzo del successivo nodo OR2 e la prima parte di un nuovo scambio DH (g^{x2}). La parte relativa a DH è cifrata

con la chiave pubblica di OR2.
OR1 decifra con K1, vede che deve estendere il circuito a OR2, quindi sceglie un nuovo CIRCID (C2) per la propria connessione con lui e infine copia la prima parte dell' handshake DH in una cella CREATE che poi manda a OR2.
OR2 decifra con la propria chiave privata, risponde a OR1 con una cella CREATED contenente g^{y2} (seconda parte dello scambio DH con OP), insieme al solito hash della chiave negoziata K2.
Dopo aver ricevuto questa risposta, OR1 inserisce il tutto in una cella RELAY EXTENDED e la passa indietro ad OP cifrandola con K1.
OP decifra, ricava la chiave condivisa K2, ne verifica l'integrità: in caso di successo il circuito ora è esteso fino a OR2.



Il client ora è in possesso di K1 e K2 condivise rispettivamente con OR1 e OR2, supponiamo voglia fare un ulteriore aggiunta al percorso.

Il procedimento è analogo a quello esposto in precedenza, con la differenza che questa volta, la cella RELAY spedita dall'OP viene cifrata iterativamente, prima con la chiave K2 e poi con la chiave K1.
La cella viene inoltrata verso OR1, il quale ne decifra il primo strato mediante la propria chiave di sessione (K1) e controlla se è lui a dover processare la cella calcolando il Digest presente nel Relay header. Verificato di non essere lui il nodo designato, invia la cella (spogliata del primo strato) a OR2 che, dopo aver decifrato lo strato successivo mediante K2 e controllato il Digest, invia una cella CREATE verso OR3.
A questo punto OR3 manderà la sua risposta CREATED a OR2 con seconda parte di DH e hash della chiave K3 ricavata.
Infine la notifica arriverà a OP con il meccanismo precedente, ma al contrario, estendendo ulteriormente il circuito.



1.2.3 Gestione delle celle RELAY da parte del circuito

Una volta che l'OP ha creato il circuito, quindi condivide una chiave con ogni OR del circuito, può mandare una generica cella RELAY in questo modo:

- l'OP seleziona l'OR destinatario (es: OR2) e calcola il digest del campo Data: questa informazione verrà letta come ESEGUI solo dal nodo destinatario, mentre per gli altri nodi intermediari equivarrà a un INOLTRA;
- l'OP cifra iterativamente il Relay header e payload con la chiave simmetrica di ogni nodo del circuito, dall'OR destinatario (ossia il più lontano) a quello più vicino;
- ogni OR del circuito che riceve una cella RELAY, controlla il CircId, decifra il Relay header e il payload con la chiave di sessione negoziata per quel circuito e verifica il digest del campo Data: se il digest è valido significa che l'OR è l'esatto destinatario della cella e ne processa il contenuto, altrimenti inoltra la cella lungo il circuito verso il prossimo OR;

L'OP tratta le celle RELAY in arrivo in maniera simile:

- l'OR mittente calcola il Digest del campo Data e cifra l'header e il payload con la chiave che condivide con l'OP;
- Al passaggio della cella tutti gli OR sul circuito cifrano a loro volta l'header e il payload della cella con la propria chiave mandando il messaggio così formato indietro;
- l'OP quando riceve la cella RELAY, iterativamente decifra ciascuno strato con la chiave relativa ad ogni OR del circuito, dal più vicino al più lontano. Ad ogni passaggio di decifratura, l'OP verifica il digest e, se esso è corretto, è sicuro che la cella provenga dall'OR corrispondente all'ultima chiave utilizzata per la decifratura. Questo permette un effettivo controllo sull'integrità dei dati trasmessi.

Capitolo 2

Hidden services

Il sistema Tor è usato principalmente per offrire anonimato ai client, ma permette questo livello di segretezza anche ad interi server: è possibile ospitare un server nella rete Tor.
L'idea è quella di nascondere un servizio all'interno di un nodo per tenere la localizzazione del server sconosciuta. Questa pratica può essere di particolare utilità quando si vuole sfuggire a certe misure di censura, ma risulta conveniente anche dal punto di vista della sicurezza: conoscere la posizione di un determinato server, infatti, facilita gli attacchi a quel servizio. Un attaccante potrebbe violare quel server per sabotarlo o per raccogliere dati sugli utenti che ne fruiscono (un attacco DOS). In conclusione la rete Tor può fungere anche da labirinto, nel quale celare servizi: i cosidetti *Hidden Services*.

2.1 Funzionamento

2.1.1 Premessa

Nella costruzione di un hidden service entrano in gioco due nuove entità:

- gli *introduction point* sono alcuni OR (di default sono 3, ma possono arrivare fino a un massimo di 20) scelti dal provider, opportunamente pubblicizzati, che fungono da punti d'accesso al servizio.
- il *rendezvous point* è un determinato OR scelto dal client che funge da intermediario nella comunicazione tra il server e il client.

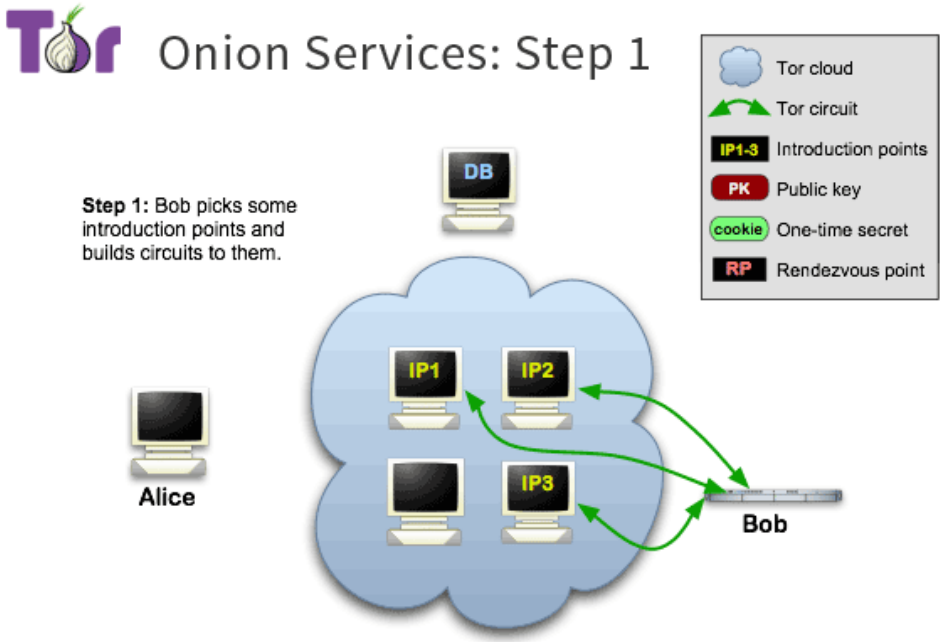
Per rendere maggiormente utilizzabile il servizio, è necessario che il provider possa in un qualche modo "pubblicizzarlo" nella rete e che un potenziale utente sia in grado di conoscerne le informazioni essenziali per poterne far uso. Tutte le informazioni sugli hidden services sono contenute nei Directory

server, i database della rete Tor (gli stessi che contengono la rubrica delle chiavi pubbliche di tutti gli onion router della rete Tor). Per ottenere le informazioni su determinati servizi è necessario uno strumento per la ricerca dei dati: in questo compito Tor si avvale di un sistema di lookup chiave-valore. Senza entrare nei dettagli, un sistema di lookup è una modalità di ricerca delle risorse; nel nostro caso un lookup system a chiave-valore "memorizza" le risorse con delle parole chiave tramite i rispettivi hash. Un client che scopre il servizio, può ottenere tutte le informazioni interrogando il lookup system. Nel caso voglia rimanere anonimo, può effettuare questo passaggio tramite una connessione Tor. Vediamo ora come funziona generalmente il meccanismo di distribuzione del servizio.

2.1.2 Costruzione di un Hidden Service

Primo passo

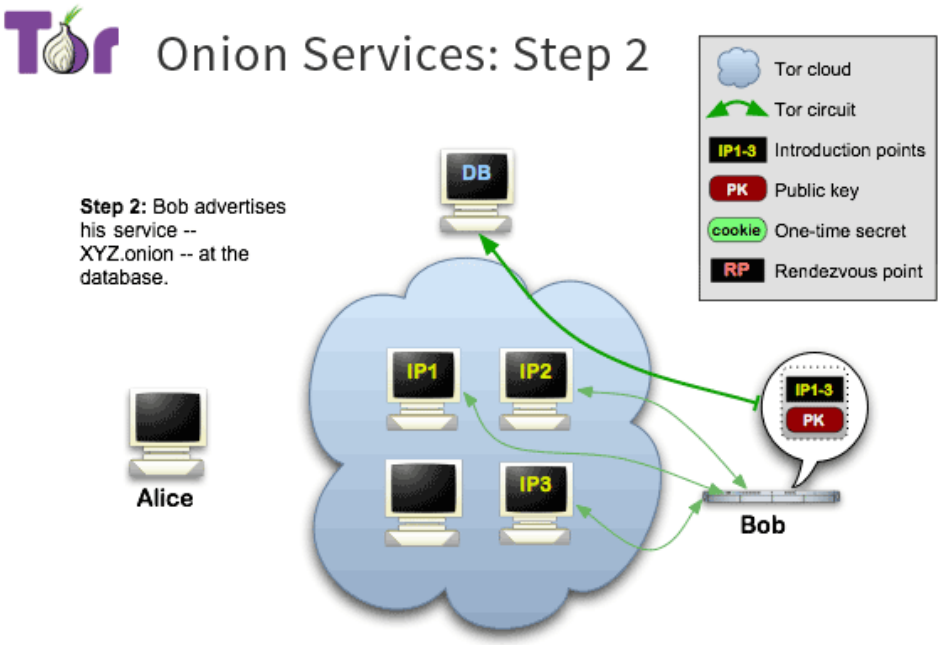
Supponiamo Bob voglia creare un *hidden service*. Per prima cosa, genera una coppia di chiavi a lungo termine: una pubblica e una privata, che saranno utilizzate per identificare il servizio.
Poi sceglie alcuni nodi e stabilisce una connessione Tor con ognuno di essi: questi saranno gli introduction point del servizio. Bob comunica loro la chiave pubblica e chiede di aspettare le richieste dei client.



Secondo Passo

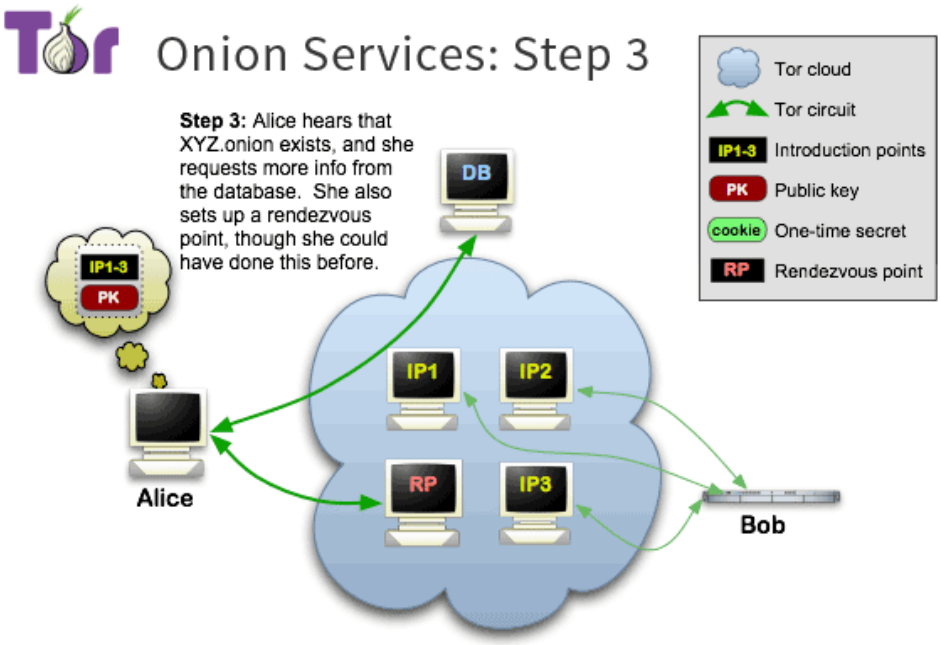
L'Onion Proxy compila il cosidetto *Hidden Service Descriptor*, cioè una dichiarazione, firmata con chiave privata, contenente la chiave pubblica e un sommario di tutti gli introduction point. Invia il descriptor ai directory server (usando sempre un circuito completo Tor per celare questo collegamento), questi provvederanno a indicizzare il tutto con la chiave pubblica per il lookup system. Il descriptor verrà trovato dai client che richiederanno XYZ.onion, dove "XYZ" è un nome di 16 caratteri derivato in maniera unica dalla chiave pubblica. Ora il servizio è attivo.

La creazione automatica del nome del servizio ha lo scopo di garantire a tutti (client, introduction point e directory server) la certezza di star lavorando col giusto onion service, anche se questo può risultare scomodo e poco pratico.



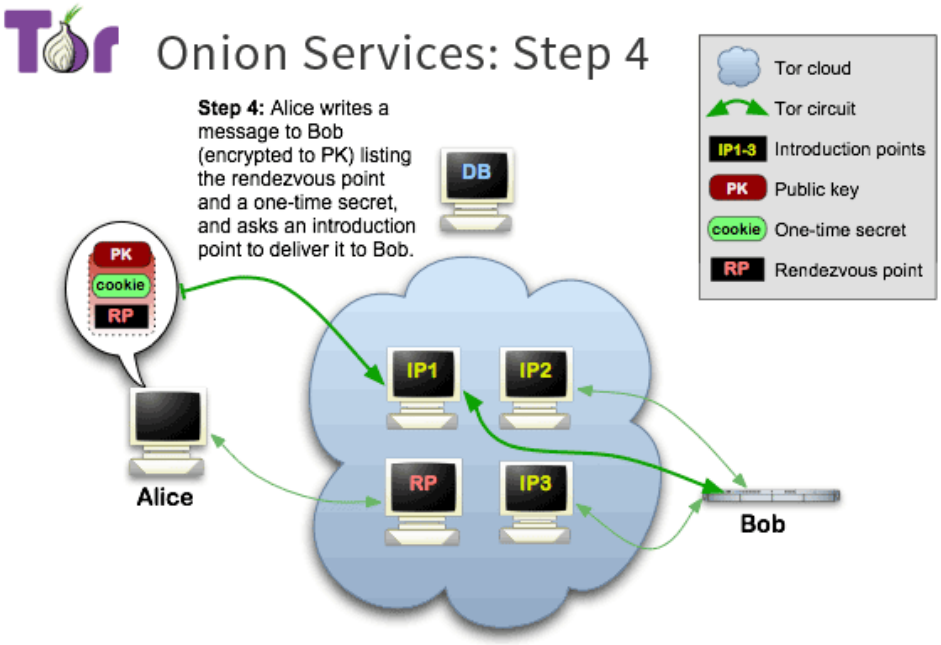
Terzo passo

Un client, Alice, viene a conoscenza del servizio e vuole contattarlo. Allora interroga il lookup system e scarica il descriptor del servizio desiderato. Lo user così è in possesso della chiave pubblica e degli introduction points. Intanto, Alice costruisce un circuito Tor con un altro nodo, comunicandogli un one-time secret (detto anche rendezvous cookie: un numero arbitrario di 20 byte) e chiedendogli di fare da *rendevous point*.



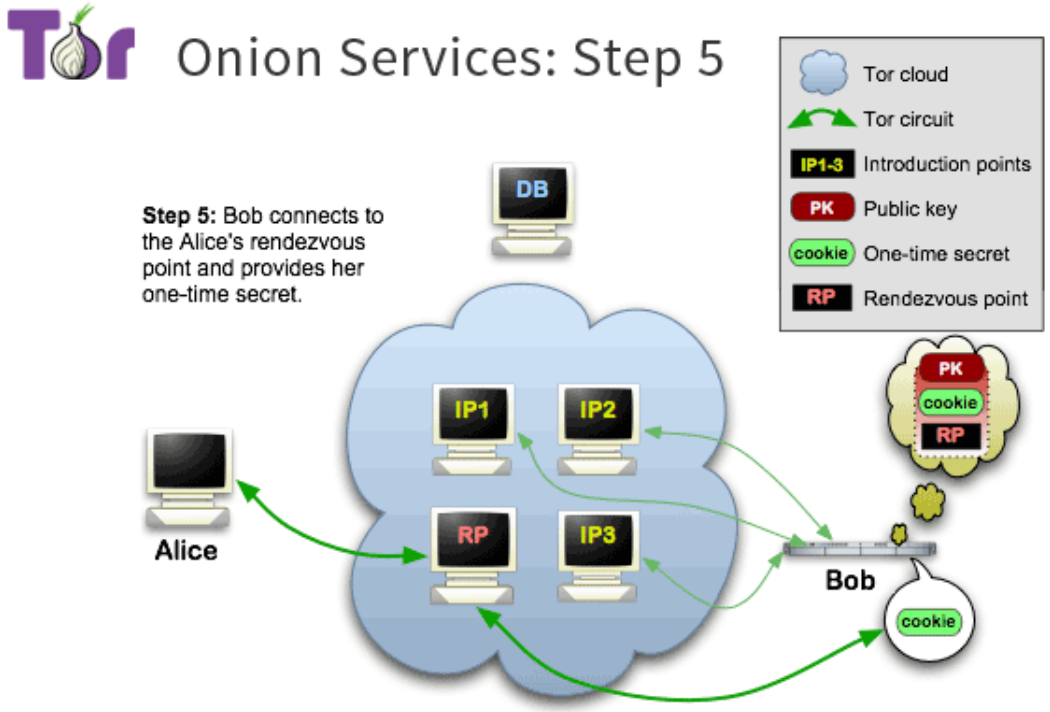
Quarto passo

Una volta che il client è connesso con il rendezvous point e conosce il descriptor, scrive un *introduce message* a uno degli introduction point. L'introduce message contiene l'indirizzo del rendezvous point e il one-time secret, tutto cifrato con la chiave pubblica del servizio. L'introduction point non fa altro che girare il messaggio al server vero e proprio.
Si osservi che essendo la comunicazione anonima grazie all'onion routing, nessuno è in grado di correlare il mittente all'introduce message.



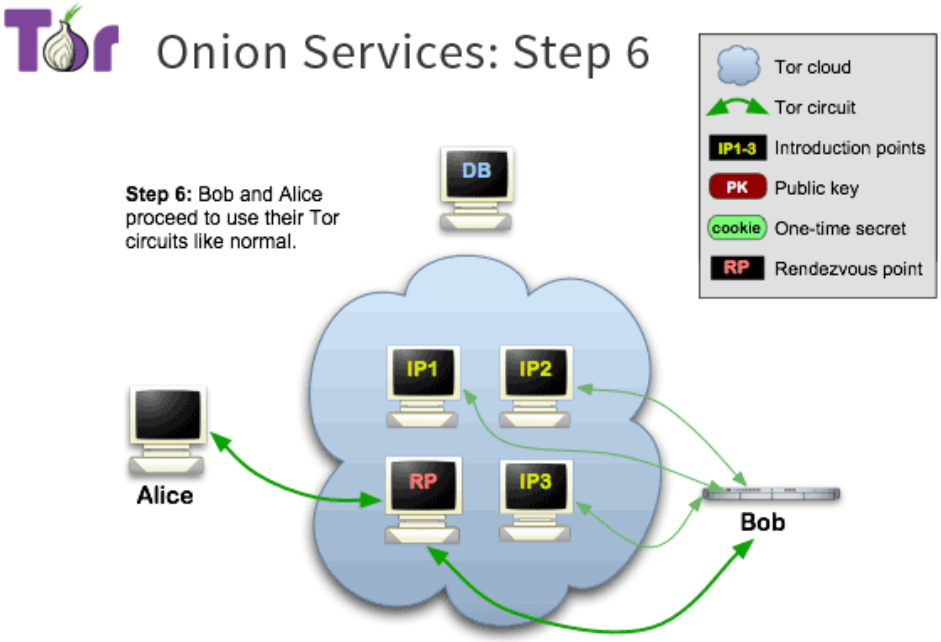
Quinto passo

Bob decifra l'introduce message del client scoprendo l'indirizzo il cookie e il rendezvous point. Poi crea un circuito verso quest'ultimo e gli invia il one-time secret in un messaggio chiamato *rendezvous message*.
Si noti che il circuito tra l'introduction point e il provider comunque non viene distrutto, permettendo così al secondo di continuare ad accettare connessioni tramite il primo.



Sesto passo

Infine il rendezvous point conferma al client l'avvenuta connessione con il server. Ora Alice e Bob possono comunicare attraverso il rendezvous point che semplicemente inoltra i messaggi da un capo all'altro usando i circuiti precedentemente stabiliti.



Capitolo 3

Attacchi alla rete Tor

Passiamo ora a vedere una carrellata dei possibili attacchi al sistema di Onion Routing.
Possiamo raggruppare tutti i possibili attacchi in due categorie: quelli **passivi** e quelli **attivi**. La sostanziale differenza fra i due tipi di approcci è che in quelli passivi, l'avversario non altera i dati in transito, mentre in quelli attivi l'avversario cerca di inserirsi modificando informazioni, messaggi oppure impersonando server.

3.1 Attacchi passivi

Osservazione del pattern del traffico

Osservare le connessioni di un utente non rivelerà la destinazione o il contenuto dei dati, ma può evidenziare uno schema (pattern) del traffico in entrata e in uscita, infatti particolari tipi di file hanno una propria traccia caratteristica (voluminoso e lento, intervallato e breve...).

Osservazione dei contenuti dell'user

Se i dati vengono cifrati quando entrano nel circuito Tor, quelli in uscita non sempre lo sono, per cui un attaccante che osserva i dati in uscita da un exit node può leggerli in chiaro.

Correlazione temporale

Un avversario che osserva i movimenti di dati ed è in grado di leggere sia nel punto di partenza, sia nel nodo di uscita può misurare i tempi

di invio e ricezione per ricavare maggiori informazioni sul contenuto della comunicazione.

Correlazione sul volume dei dati

L'idea è la stessa dell'attacco precedente, ma invece che analizzare i tempi, l'attaccante studia i volumi di dati. In caso di coincidenze, può essere in grado di correlare due nodi.

Fingerprinting

Questo tipo di attacco è simile ai precedenti: l'attaccante conosce i pattern temporali e volumetrici di tanti tipi di dati e costruisce una sorta di database. A questo punto osserva il traffico di un nodo client e cerca delle corrispondenze nel database, a quel punto può identificare tipi di dati e quindi capire l'attività del client.

3.2 Attacchi attivi

Compromissione delle chiavi

Un attaccante che riesca ad impossessarsi della chiave di sessione TLS può vedere le celle di controllo e le celle di relay cifrate su quella connessione. Se scopre invece una chiave di sessione del circuito sarà in grado di rimuovere uno strato di cifratura.
La possibilità di compromettere un OR gli permetterebbe di comportarsi da tale, e nel peggiore dei casi potrebbe addirittura risalire lungo il circuito fino a compromettere tutti i nodi del circuito. Questo attacco comunque risulta difficile grazie alla rotazione frequente delle chiavi e a una breve durata del circuito, infatti una volta che il circuito viene distrutto, diventa impossibile collegare sorgente con destinazione.

Smear attacks

Un attaccante potrebbe usare la rete Tor per commettere azioni non approvate allo scopo di abbassare il grado di reputazione della rete e costringere gli operatori a chiuderla.

Dos non-observed node

Un osservatore che ha capacità limitate e quindi può controllare solo una parte del traffico, potrebbe attaccare i nodi non osservati con l'obiettivo di farli chiudere o abbassare la loro reputazione, in questo modo gli utenti saranno persuasi a non usarli.

Tagging

É l'azione di alterare una cella inserendo dei tag nel flusso delle comunicazioni. Se l'utente accede a certi siti con celle taggate, è più facile scoprire il suo cammino.

Compromissione dei directory server

Le autorità di directory approvano di volta in volta un consenso sullo stato della rete, in base alla loro visione di essa. Se qualche server non funzionasse, questa operazione verrebbe fatta da quelli rimanenti, quindi se vengono compromessi abbastanza server, può venire pubblicato uno stato non valido o non funzionante.
A causa di questo sistema di approvazione a maggioranza assoluta, nel caso un attaccante riuscisse a controllare più di metà DS, egli sarebbe in grado di introdurre a piacere nodi nella rete.

Splitting

L'attaccante cerca di convincere alcuni operatori sulla fiducia da dare o meno verso un altro server,creando divisioni e confusione nel mettersi d'accordo. Ottiene un cosiddetto split del consenso generato dalle autorità e così facendo è in grado di isolare alcuni utenti sulla base del consenso ricevuto.

Tricking

Consiste semplicemente nel riuscire a far approvare un nodo non funzionante oppure togliere la fiducia a uno corretto, portando maggiore inaffidabilità nella rete.

Attacchi agli hidden service

La strategia più comune per attaccare un hidden service è quella del *flooding*, ossia inondare di richieste gli introduction point sovraccaricando la

connessione e impedendo al provider di fornire il suo servizio.
Il controllo degli introduction point invece permette all'attaccante di indirizzare le richieste dove vuole, ma visto che il servizio ha sempre più di un nodo di ingresso, se l'attaccante vuole distruggere un intero hidden service dovrà compromettere tutti i suoi introduction point.

Capitolo 4

Conclusioni

Garantire anonimato in rete è senza dubbio un obiettivo ambizioso, difficile, ma di grande attualità. Tor ha raggiunto negli anni un buon livello di servizio, ma abbiamo visto come possa essere vulnerabile ugualmente a certi attacchi.
Il problema principale di Tor è che essendo gestito da volontari, che costruiscono i nodi della rete e mettono a disposizione la loro banda per far passare il traffico, non dispone di abbastanza "potenza". Infatti il numero di nodi non è così elevato da poter gestire una grossa domanda. Chiaramente più persone conosceranno Tor, più persone lo useranno e la base di volontari pian, piano si allargherà, ingrandendo la rete e aumentando il suo livello di protezione di identità. Infatti in questo sistema, maggiore usabilità significa maggior sicurezza. Quindi la futura direzione di Tor è sicuramente cercare di attirare più utenti per potersi espandere in grandezza e migliorarsi in qualità del servizio.
Tor però è stato reso famoso anche per il suo uso a scopi criminali, infatti è tendenzialmente uno degli strumenti più utilizzati per entrare nel *dark web*, quella parte nascosta di internet (ossia costituita da siti non indicizzati dai motori di ricerca), dove si trovano i cosiddetti *black market* (ossia siti per acquistare illegalmente armi, droghe...). La lotta alle associazioni criminali e terroristiche, quindi passa anche nel riuscire ad attaccare Tor e questo ovviamente non fa bene alla piattaforma. Immaginiamo che un governo riesca a tracciare le attività di un criminale in rete, a quel punto cosa gli impedirebbe di spiare anche i cittadini comuni? Quindi quando si parla di progresso della privacy della rete, bisogna guardare i due lati della medaglia.
In ogni caso, penso che investire nella ricerca in questo settore sia necessario perché in questo nuovo millennio, tutta la nostra persona si ritrova online: foto, informazioni, conti bancari, acquisti, attività, lavoro...
É importante, secondo me, riuscire a riappropriarci della nostra identità.

Appendice A

Il protocollo TLS

Il protocollo TLS (acronimo di *Transport Layer Security*) è uno standard di "presentazione" tra un client e un server che garantisce maggiore sicurezza nella connessione effettuata. Nello specifico, il protocollo fornisce:

- *Autenticazione*, cioè la certezza di conoscere l'identità del server cui ci sta connettendo.
- *Integrità dei dati*, che significa protezione da potenziali modifiche (accidentali o meno) delle informazioni passate nel canale.
- *Confidenzialità*, ossia protezione dei contenuti da eventuali terze parti che intercettino i messaggi.

Osservazione: l'autenticazione, normalmente, è unilaterale: è solo il server ad autenticarsi presso il client. Questo significa che l'utente conosce l'identità del server, ma non viceversa: l'utente resta anonimo agli occhi del servizio. É comunque possibile ottenere bilateralità, bisognerà che anche il client si doti di un proprio certificato digitale.

Principi di funzionamento

Possiamo sommariamente vedere il protocollo in 3 fasi:

1. Negoziazione fra le parti dell'algoritmo da utilizzare
2. Scambio delle chiavi e autenticazione
3. Cifratura simmetrica e autenticazione del messaggio

Nella prima fase, client e server si accordano su quali algoritmi crittografici usare nella comunicazione. In particolare devono scegliere un protocollo di cifratura simmetrica (DES, Triple DES o AES i più utilizzati), uno standard per lo scambio delle chiavi (di solito a chiave pubblica come RSA o Diffie-Hellmann) e l'algoritmo di autenticazione (ossia il MAC).

La fase di autenticazione del server consiste semplicemente nel far verificare al client la validità della firma digitale del certificato dello stesso server, questa deve essere riconosciuta presso una *Certificate Authority* (CA).

A questo punto, dopo essersi scambiati la chiave K, può iniziare lo scambio dei dati usando la cifratura simmetrica. L'integrità dei dati è garantita usando un costrutto HMAC (keyed-hash message authentication code): una combinazione del messaggio originale e della chiave segreta che viene data in input a una funzione hash (MD5 o SHA3-256). HMAC funziona esattamente come tutti i protocolli di autenticazione: è un tag che si aggiunge al messaggio e che il destinatario calcola e controlla al momento della ricezione.

Bibliografia

[1] Roger Dingledine, Nick Mathewson, Paul Syverson,
Tor: The Second-Generation Onion Router, 2004.

[2] Alfredo de Santis, *The Onion Router*, 2010.

[3] Wikipedia, *Tor(software)*,
`https://it.wikipedia.org/wiki/Tor_(software)`

[4] Wikipedia, *Onion routing*,
`https://it.wikipedia.org/wiki/Onion_routing`

[5] The Tor Project, *Tor: Onion Service Protocol*,
`https://www.torproject.org/docs/onion-services.html.en`

[6] Wikipedia, *Transport Layer Security*,
`https://it.wikipedia.org/wiki/Transport_Layer_Security`