



**Università degli Studi di Camerino**

**SCUOLA DI SCIENZE E TECNOLOGIE**

**Corso di Laurea in Informatica (Classe L-31)**

**Servizi Onion  
Dalla teoria all'implementazione**

Laureando  
**Leonardo Migliorelli**

**Matricola 113920**

Relatore  
**Fausto Marcantoni**



# Indice

<b>1</b>	<b>Introduzione</b>	<b>5</b>
1.1	Obiettivi . . . . .	5
1.2	Struttura della Tesi . . . . .	5
<b>2</b>	<b>Onion Routing</b>	<b>7</b>
2.1	Onion come fix alle vulnerabilità della rete internet . . . . .	9
2.2	Applicazioni di utilizzo . . . . .	9
2.3	Onion come Mix Network . . . . .	10
<b>3</b>	<b>Tor § Onion v2</b>	<b>13</b>
3.1	Obiettivi . . . . .	14
3.2	Network Design . . . . .	16
3.2.1	Controllo di Congestione . . . . .	17
3.2.2	Tor Relay . . . . .	18
3.3	Directory Servers . . . . .	19
3.4	Tor Browser . . . . .	20
3.5	Dimostrazione da Wireshark . . . . .	21
<b>4</b>	<b>Implementazione</b>	<b>27</b>
4.1	Servizi Onion . . . . .	27
4.2	Onion V3 . . . . .	28
4.3	Studio delle tecnologie . . . . .	29
4.4	Creazione del servizio . . . . .	30
4.4.1	Gestire la comunicazione tramite i socket unix . . . . .	32
4.4.2	Creazione di un url personalizzato . . . . .	34
4.4.3	Far conoscere il sito . . . . .	36
<b>5</b>	<b>Sistemi di pagamento nella rete onion</b>	<b>39</b>
5.1	Bitcoin . . . . .	39
5.2	Creazione di un wallet BTC . . . . .	39



# 1. Introduzione

Le moderne tecnologie di rete consentono una rapida comunicazione da ogni parte del mondo, avvicinando culture altrimenti distanti migliaia di chilometri. Due dei più grandi temi del nostro secolo sono la privacy e l'anonimato, parlando di reti internet ogni connessione tra client e server passa per una moltitudine di router che conoscono esattamente l'indirizzo (e quindi l'identità) del mittente e del destinatario. Con un po' di conoscenze non è complicato scoprire questi dati e sfruttarli a proprio vantaggio, le stesse big company spesso usano l'indirizzo IP con cui ci si connette al loro sito per tracciare l'utente e fornirgli articoli e pubblicità mirata o vendere i medesimi dati a terzi. La Rete Onion è stata creata per risolvere esattamente questo problema.

## 1.1 Obiettivi

La tesi ha come principale obiettivo la dimostrazione di come è possibile generare un servizio/web server che sfrutta la rete Tor per rendere le connessioni anonime e rendere anonimo lo stesso server, creare un hostname Tor personalizzato, far sì che il servizio sia raggiungibile attraverso un motore di ricerca onion e fornire un sistema di pagamento integrato per i prodotti del servizio.

## 1.2 Struttura della Tesi

Nel primo capitolo viene introdotto l'onion routing, i possibili utilizzi e le mix networks. Il secondo capitolo introduce la rete Tor con le relative migliorie apportate all'onion routing e gli obiettivi prefissati per l'implementazione della rete. Il terzo capitolo introduce ai servizi onion, alla rete V3 che ha apportato alcune modifiche importanti per la sicurezza, precondizioni necessarie per l'implementazione che verrà mostrata dopo uno studio delle tecnologie disponibili nel quarto capitolo. La tesi termina con la descrizione dei sistemi di pagamento e come sono stati implementati nel servizio onion.



## 2. Onion Routing



Figura 2.1: Vita di un pacchetto onion

La **rete onion** è una rete distribuita composta dall’insieme di **onion router** che agiscono come nodi di rete e collaborano per portare un pacchetto dalla sorgente alla destinazione. Il tutto avviene senza che nessun nodo possa conoscere contemporaneamente l’host sorgente e l’host di destinazione, grazie alla criptografia a strati del pacchetto, per cui ogni strato viene criptato con una chiave differente e può essere decriptato solo dal nodo con la stessa chiave simmetrica, scoprendo così le informazioni sul prossimo nodo. Il cosiddetto exit node<sup>1</sup> può infine decriptare la parte finale del messaggio e scoprirne il corpo. La risposta del pacchetto segue lo stesso criterio sfruttando nodi, algoritmi e chiavi differenti.

Questo meccanismo è derivato dallo studio di David Chaum riguardo alle mix networks (Sezione 2.3). Questo concede all’utilizzatore di rimanere completamente anonimo, cosa che non può avvenire con la semplice criptografia SSL che agisce esclusivamente sul corpo del messaggio.

---

<sup>1</sup>Il nodo finale del circuito che conosce l’indirizzo reale della destinazione

La prima operazione che deve effettuare il client<sup>2</sup> per utilizzare la rete onion è la generazione di un nuovo circuito specificando il percorso di nodi che ogni pacchetto dovrà seguire, iniziando dal primo nodo vengono scambiate informazioni crittografiche come l'algoritmo e le chiavi da usare, successivamente si usano queste informazioni per generare un pacchetto indirizzato al secondo nodo per ottenere le relative informazioni crittografiche e così via fino a che non si è generato tutto il circuito (Fig. 2.2), grazie a questo meccanismo neanche durante la generazione del circuito è possibile risalire al client.

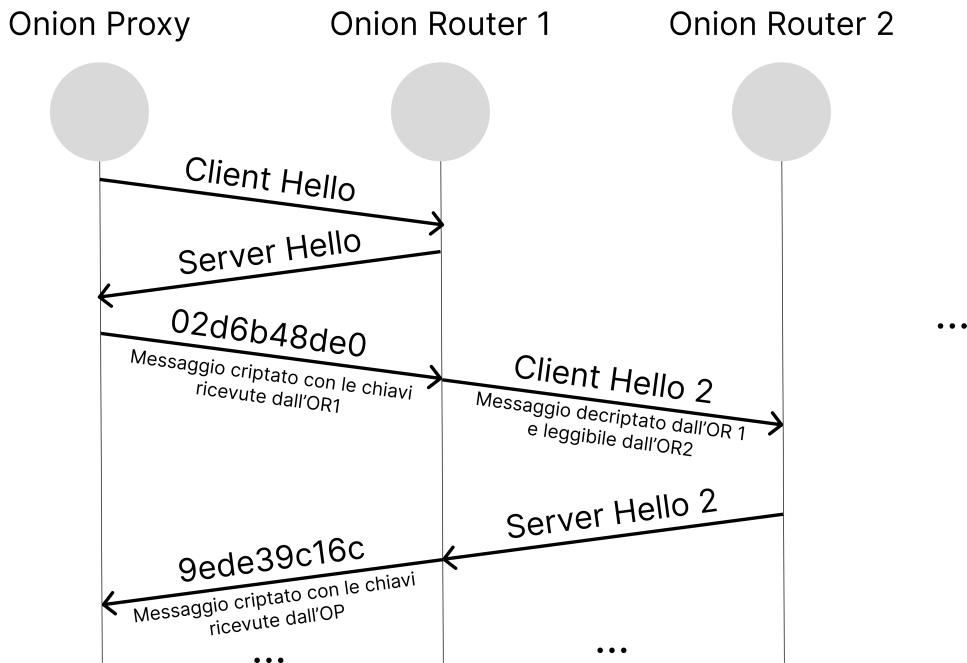


Figura 2.2: Creazione del circuito onion

Dato che la criptografia asimmetrica è troppo costosa per essere utilizzata durante lo scambio dati, essa viene usata solo durante la generazione del circuito e la condivisione delle chiavi simmetriche<sup>3</sup>, successivamente gli strati vengono criptati e decriptati con la stessa chiave simmetrica. Questo consente alla rete onion di avere una bassa latenza che è incrementata solo dal numero di onion router nel percorso e non dalla tecnologia che non è distante da quella usata in HTTPS [GD99].

<sup>2</sup>Anche conosciuto come onion proxy

<sup>3</sup>In un meccanismo simile a quello dell'handshake di HTTPS/TLS

## 2.1 Onion come fix alle vulnerabilità della rete internet

Nella rete internet ci sono due principali vulnerabilità che gravano sull'anonimato e sulla privacy:

- **Traffic analysis**, qualsiasi utente ha la capacità di analizzare il traffico, dato che i protocolli sono standardizzati e gli indirizzi non sono criptati è facile risalire alla sorgente e destinatario del pacchetto<sup>4</sup>.
- **Eavesdropping**, consiste nell'intercettazione dei pacchetti per leggerne il contenuto<sup>5</sup>.

La rete Onion è nata proprio per risolvere queste problematiche, implementando le giuste tecnologie per proteggere gli utenti dall'analisi del traffico e dalle intercettazioni. [MGRG]

## 2.2 Applicazioni di utilizzo

L'onion routing può essere usato con una moltitudine di protocolli e applicazioni, tra i più comuni troviamo HTTP(S), FTP, SSH, SMTP e DNS. L'utilizzo di molti dei protocolli più comuni avviene tramite gli **Onion Proxies**, i quali sono suddivisi in tre proxy layer logici:

- Un proxy che genera e gestisce le connessioni, per operare ha necessità di conoscere la topologia e i percorsi verso altri nodi, informazioni che vengono distribuite in modo sicuro all'interno della rete a ogni nuovo nodo che si connette.
- Un proxy chiamato “**Application Specific Proxy**”, per ogni connessione la relativa applicazione<sup>6</sup> invia al proxy il pacchetto che normalmente invierebbe al server di destinazione, il proxy si occupa di convertire lo stream di dati in un formato accettato dalla rete onion.
- Un proxy opzionale chiamato “**Application Specific Privacy Filter**” che significa lo stream di dati rimuovendo informazioni dal contenuto che potrebbero identificare la sorgente.

I proxy possono essere configurati in molteplici modi, tra i quali vi è la possibilità di eseguire il proxy in un server remoto e sfruttare la rete Tor senza dover installare il software in ogni dispositivo, che quindi non ne deve gestire la computazione. [GD99]

---

<sup>4</sup>Si prenda il considerazione il caso di comunicazioni tra aziende i cui accordi sono confidenziali, una semplice analisi del traffico potrebbe rendere noto a chiunque l'esistenza di accordi

<sup>5</sup>Onion risolve anche questa problematica come effetto collaterale soluzione apportata all'analisi del traffico, infatti insieme all'indirizzo viene criptato anche il contenuto

<sup>6</sup>Un mail client come un Web Browser

## 2.3 Onion come Mix Network

La rete Tor è una delle molteplici reti basate sullo studio di David Chaum sulle Mix network, il suo studio si concentrò sulla rete basata sulla crittografia asimmetrica e sui sistemi di risposta senza conoscere l'identità del destinatario. Tutta la rete doveva funzionare senza necessità di un ente centrale a gestire le connessioni.

Abbiamo una chiave pubblica K e una chiave privata P, da cui derivano due funzioni:

- $K(x)$  la funzione che cripta  $x$  con la chiave pubblica, può solo essere decriptato con la chiave P
- $P(x)$  la funzione che cripta  $x$  con la chiave privata, può solo essere decriptato con la chiave K

Da questo è facile dedurre che  $P(K(x)) = K(P(x)) = x$

Con questa tecnica però un malintenzionato potrebbe determinare il contenuto di un messaggio  $x$  creandone copie identiche  $y_n$  fino a che  $K(y_n) = K(x)$  verificando l'uguaglianza  $y_n = x$  e scoprendo il corpo del messaggio. Per risolvere questo problema si inserisce una stringa casuale R nella funzione crittografica del messaggio ottenendo quindi risultati sempre diversi tramite le funzioni  $K(x, R)$  e  $P(x, R)$ , questa tecnica è chiamata **sealing**. Questa tecnica incrementa di molto la complessità, in quanto per determinare il contenuto del messaggio è inoltre necessario indovinare la stringa R.

In una rete questo sistema viene implementato in maniera ridondante

$K1(R1, Kb(R0, M), B)$ <sup>7</sup>.

Quando il nodo 1 riceve il pacchetto lo decripta con la sua chiave privata scartando R1 e inoltrando il nuovo pacchetto<sup>8</sup> a B. Questo processo può essere esteso ad N nodi (mix) incrementando la sicurezza della rete.

Il destinatario di un pacchetto deve avere la possibilità di rispondere sfruttando un indirizzo non tracciabile.

Il dispositivo A utilizza il proprio indirizzo reale per creare un indirizzo non tracciabile, generando due chiavi pubbliche  $Ka$  e  $R1$ <sup>9</sup>, critta il proprio indirizzo A assieme alla chiave R1 come stringa casuale usando la chiave del mix

$K1(R1, A), Ka$

Quando B deve rispondere al pacchetto usa  $Ka$  per criptare il messaggio e  $K1(R1, A)$  come indirizzo di destinazione, il mix M1<sup>10</sup> che riceve il pacchetto usa la chiave privata P1 per decriptare l'indirizzo di destinazione (A) e usa la stringa casuale R1 come ulteriore chiave per criptare il messaggio.

$K1(R1, A), Ka(R0, M) \Rightarrow A, R1(Ka(R0, M))$

Con questo sistema B può rispondere ad A, non conoscendo il vero indirizzo di A, il mix non conosce il contenuto del messaggio ma conosce l'indirizzo di destinazione A che è l'unico che può decriptare il contenuto del messaggio.

Da considerare che tra M1 e B possono esserci N nodi e il messaggio può quindi essere

<sup>7</sup>il messaggio M viene criptato insieme a una stringa casuale R0 con la chiave pubblica del destinatario Kb, il tutto viene criptato assieme all'indirizzo B e a una stringa casuale R1 con la chiave pubblica K1 del nodo 1

<sup>8</sup> $Kb(R0, M)$

<sup>9</sup>R1 in questo caso viene usato sia come stringa casuale ma in realtà esso è anche una chiave pubblica che verrà utilizzata dopo

<sup>10</sup>Il primo mix nel percorso da A a B, il più vicino al nodo A, viene inoltre considerato un nodo di uscita dalla rete dato che è l'unico che conosce il vero indirizzo di destinazione

criptato più volte nel percorso da B a M.

Nelle Mix Networks abbiamo un ente centrale che possiede una lista di pseudonimi creati dagli utenti. Quando un utente vuole creare il proprio pseudonimo invia la richiesta all'ente che decide se accettarla, solo allora lo aggiunge alla lista.

La richiesta contiene una chiave pubblica che agisce da pseudonimo, viene usata per verificare le firme generate dall'utente tramite la relativa chiave privata. Le richieste all'ente possono essere inviate in maniera anonima, tramite una mail non tracciabile o un altro sistema.

[**Cha81**]



### 3. Tor § Onion v2

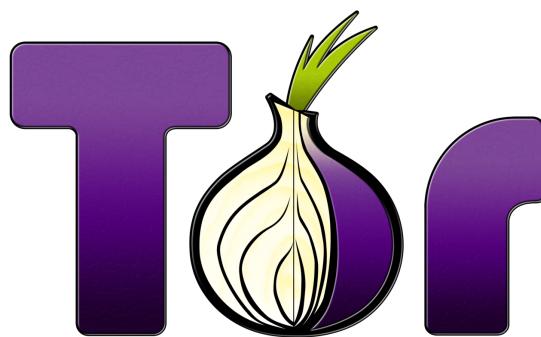


Figura 3.1: Tor logo

Nel 2002 viene presentata la rete Tor, diventata open source 2 anni dopo è l'implementazione più conosciuta di onion routing.

Tra le varie migliore abbiamo:

- **Circuiti Telescopici** Fig. 3.2, mentre prima il processo di creazione del circuito iniziava con la scelta dell'exit node e proseguiva con la definizione degli altri nodi<sup>1</sup>, ora inizia con la scelta del primo nodo, viene instaurata una connessione verso di lui e gli viene richiesto di estendere il circuito ai nodi successivi, in maniera **incrementale**. Questo porta a una migliore **segretezza del canale**, i nodi compromessi non sono più in grado di forzare gli altri onion router a decriptare un traffico precedentemente registrato.
- L'implementazione del **proxy di applicazione** attraverso lo standard SOCKS, consente alla maggior parte del traffico TCP di funzionare senza modifiche. Precedentemente era necessario implementare un proxy per ogni applicazione, e di conseguenza generare un circuito per ogni applicazione, con conseguente duplicazione di chiavi. Il proxy implementato in TOR invece genera il circuito a livello di TCP e più processi applicativi possono utilizzarlo. Per garantire la non tracciabilità di un utente nello stesso stream dati usato da più applicazioni è stato implementato il meccanismo dei *Rotating Circuits* che genera ogni minuto un nuovo circuito se quello precedente non è in uso.
- **Controllo di congestione**, un sistema decentralizzato che sfrutta ack end-to-end per garantire l'anonimato (Sec. 3.2.1).
- **Directory Server**, nodi più fidati di altri che descrivono le informazioni di rete in maniera sicura e affidabile (Sec. 3.3).

<sup>1</sup>Lasciando per ultimo il nodo d'ingresso

- **Politiche di uscita variabili**, ogni nodo possiede delle politiche che specificano le connessioni consentite e rifiutate, fondamentale in una rete distribuita fatta da volontari.
- **Controllo di integrità end-to-end**, viene eseguito un controllo di integrità nel momento in cui il pacchetto esce dalla rete per garantire che i contenuti non sono stati alterati.

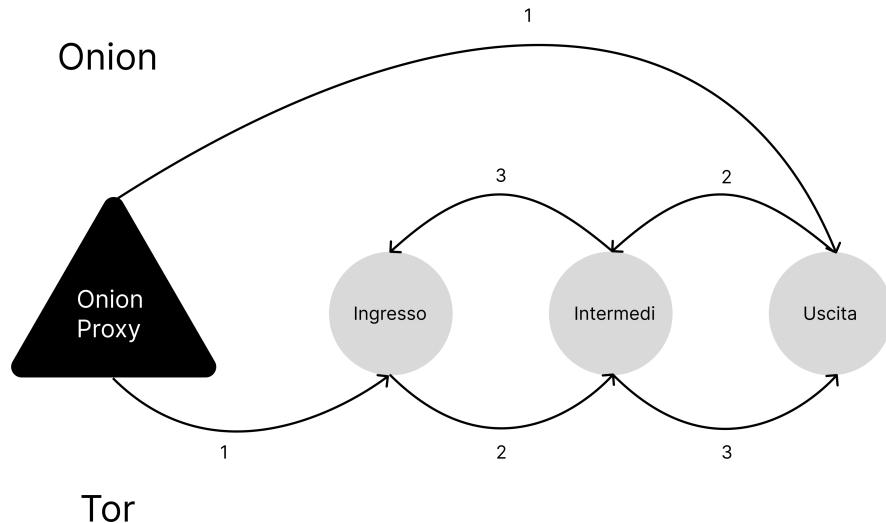


Figura 3.2: Differenza tra la creazione di un circuito onion e Tor

Tor però non è completamente sicura, infatti non filtra informazioni di privacy nel corpo del messaggio come invece avviene in altri sistemi come *Privoxy* o *Anonymizer* e non offre garanzie in caso di attacco end-to-end che concerne sia sorgente che destinazione. [RDos]

### 3.1 Obiettivi

L’obiettivo principale della rete TOR è la creazione di una rete che possa rendere gli attacchi molto più complessi da portare a termine scoraggiando così un possibile hacker, da questo principio cardine sono derivati gli altri obiettivi, tra cui:

- **Usabilità**, la rete Tor a differenza degli altri sistemi che implementano le Mix-Networks (Sec. 2.3) predilige la bassa latenza<sup>2</sup> e l’usabilità, un aspetto fonda-

<sup>2</sup>Il che rende la rete adatta all’utilizzo tramite un web browser

mentale se si vuole garantire l'anonimato<sup>3</sup>. Da questo derivano altri obiettivi come

- Basse latenze, determinate anche dal fatto che più applicazioni possono usare lo stesso circuito TCP senza doverne generare uno nuovo per ogni stream dati, questo consente di ridurre il delay causato dalla criptografia asimmetrica.
  - Essere implementabile con meno configurazioni possibili, per incrementare il numero di servizi che possano attirare utenti.
  - Essere multi piattaforma, per incrementare la base di utenti.
- **Semplicità**, la rete deve essere facile da comprendere, doveva essere usabile nel mondo reale e non doveva essere troppo costosa.

[Cha81]

---

<sup>3</sup>Maggiori sono i nodi più semplicemente si può garantire l'anonimato

## 3.2 Network Design

A differenza di altri sistemi che usano una rete peer-to-peer, Tor è una *overlay network*<sup>4</sup>, questa scelta è derivata dalle possibili vulnerabilità di una rete basata sugli utenti, infatti un attaccante potrebbe compromettere il traffico leggendo o manipolando i dati.

Ogni onion router è rappresentato come un processo software che mantiene due tipi di chiavi

- Chiave d'identità, una chiave di lunga durata usata per firmare i pacchetti garantendo agli altri nodi della rete l'autenticità del messaggio e delle informazioni contenute, tra cui indirizzo, bandwidth, exit policy, ecc..
- Chiave onion, una chiave di breve durata usata per decriptare le richieste all'interno dei circuiti utente.

Un elemento chiave della rete TOR sono le **celle**, pacchetti di dimensione fissa a 512 bytes, come ogni tipo di pacchetto sono divisi in header e payload, l'header contiene l'identificativo del circuito e un comando che indica come gestire il payload. Vi sono due tipi di celle:

- **Control**, pacchetti di controllo gestiti dal primo router che li riceve, per questo non vengono mai inoltrati. Il comando può essere:
  - **Padding** per mantenere viva la connessione.
  - **Create** per creare un nuovo circuito.
  - **Destroy** per eliminare un circuito.
- **Relay**, trasporta stream dati per cui ha necessità di un header aggiuntivo contenente l'identificativo streamID, il checksum per il controllo di integrità, la dimensione del payload e un comando di relay. Il comando può essere:
  - **Begin** per iniziare uno stream.
  - **End** per terminare uno stream.
  - **Teardown** per terminare uno stream in modo forzato, usato per quelli "rotti"
  - **Connected** per rispondere al relay begin dell'OP, informandolo che lo stream è stato creato con successo.
  - **Extend** per estendere un circuito di un router.

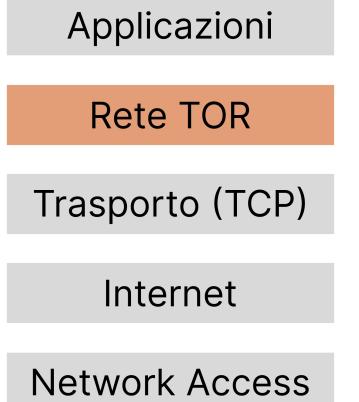


Figura 3.3: TCP/IP stack with TOR network

<sup>4</sup>Una rete virtuale creata sfruttando una rete fisica preesistente

- **Truncate** per eseguire un teardown di una parte del circuito.
- **Sendme** usato nel controllo di congestione per indicare che nuove celle dati possono essere ricevute.
- **Drop**

L'utente per generare il circuito segue un processo di negoziazione incrementale:

1. Come primo passo l'utente, o meglio l'Onion Proxy crea e invia una richiesta *relay* al primo nodo nel percorso scelto<sup>5</sup>.
2. Avviene la condivisione della chiave simmetrica tramite l'handshake di Diffie-Hellman, la creazione dell'ID del circuito e di conseguenza la creazione della connessione con il primo nodo (R1).
3. L'utente invia una richiesta relay extend indicando a R1 l'indirizzo del secondo nodo nel percorso scelto (R2), R1 inizia una connessione con R2 definendo un nuovo id e associando la connessione OP-R1 con la connessione R1-R2, OP e R2 non si conoscono a vicenda e comunicano solo tramite l'intermediario R1. In fine R1 invia all'utente le informazioni del nuovo nodo tra cui la chiave simmetrica per il relativo strato.
4. Questa operazione viene effettuata, richiedendo all'ultimo router corrente di creare una connessione con il suo successivo, finché il circuito non viene completato [Cha81].

Ogni applicazione, in base alle proprie necessità, invia le richieste di stream TCP all'Onion Proxy, che sceglie il circuito più recente (oppure genera un nuovo circuito) e sceglie un exit node, solitamente l'ultimo router. A questo punto l'OP invia una cella *relay begin* con un identificativo casuale all'exit node, la risposta dell'exit node conferma l'esistenza del nuovo stream TCP e l'OP può accettare i dati delle applicazioni TCP e inoltrarli nella rete Onion.

### 3.2.1 Controllo di Congestione

In una rete Onion non è possibile sostituire un router nel circuito, quando non è più in grado di gestire le richieste, senza rendere illeggibile il traffico, per questo è stato implementato un meccanismo di controllo di congestione che limita i circuiti o gli stream dati per evitare di saturare la rete. Sono stati sviluppati 2 meccanismi:

- **Controllo di Circuito**, ogni OP mantiene le informazioni di due finestre per ogni OR, mentre ogni OR mantiene le informazioni delle finestre dell'OP di ogni circuito di cui fa parte. Le 2 finestre iniziano da un valore di 1000 e decrementano a ogni cella, esse sono:
  - **Packaging window**, tiene traccia del numero di celle che un OR può inviare verso l'OP. Quando un OR riceve abbastanza celle dati (100) invia un relay sendme con *streamID* = 0 al relativo OP, il quale, alla ricezione, incrementa la finestra di packaging (di 100) per quell'OR. L'Onion Proxy

---

<sup>5</sup>Tor possiede una lista di relay pre-caricata da cui sceglie i propri nodi per il circuito, inoltre i relay vengono aggiornati grazie a dei server centralizzati chiamati Directory Authority

agisce in maniera simile, quando riceve abbastanza celle dati invia un relay sendme all'OR il quale incrementa la finestra per il relativo OP. Se la finestra di packaging raggiunge 0 il nodo (che sia un'Onion Router o un'Onion Proxy) smette di leggere e inviare celle attendendo il sendme.

- **Delivery window**, tiene traccia del numero di celle dati che un OR è in grado di inviare fuori dalla rete.
- **Controllo di Stream**, simile al controllo di circuito ma il meccanismo è applicato a ogni stream TCP separatamente. Ogni stream inizia con packaging = 500 con incrementi di 50 a ogni relay sendme. Il relay sendme, a differenza del controllo a livello di circuito, viene inviato quando il numero di bytes che devono essere inviati è inferiore al threshold di 10 celle.

[RDos]

### 3.2.2 Tor Relay

La rete Tor si basa su nodi gestiti da volontari che forniscono la propria potenza di calcolo per consentire alla rete Tor di funzionare, l'aumento di nodi porta la rete a essere più veloce, stabile e sicura per gli utenti. Questi nodi sono anche chiamati *Tor Relay*, ognuno possiede una exit policy che ne stabilisce il tipo e le caratteristiche, si suddividono quindi in:

- **Non-exit Relay**, sono i nodi interni alla rete, vengono definiti in base alla exit-policy, essi si dividono in:
  - **Guard Relay**, sono i primi nodi di ogni circuito, devono rispettare specifiche di velocità e stabilità non inferiori a 2MByte/s.
  - **Middle Relay**, agiscono come tramite tra i nodi *Guard* ed *Exit*, possono diventare Guard solo se rispettano le relative specifiche.
- **Exit Relay**, sono gli ultimi nodi di ogni circuito, ricevono traffico dalla rete Tor e lo inoltrano in una rete normale. Questo porta coloro che fanno hosting di questi relay a essere i più esposti a rischi legali<sup>6</sup>, per questo e altri motivi è sconsigliato eseguire un Tor Relay dalla propria abitazione e viene piuttosto suggerito di affidarsi ad altre istituzioni.
- **Bridge Relay**, sono nodi non pubblici<sup>7</sup> i cui IP non possono quindi essere bloccati da governi o ISP in quanto privati. Inoltre generalmente non ricevono denunce legali di abuso della rete internet come invece accade per gli Exit Relay.

[Torf] [Torh]

Dal sito Tor [Torg] è possibile vedere le statistiche riguardanti i relay, la loro bandwidth, il tipo, gli indirizzi IP e il tempo di esecuzione. Dallo stesso sito è inoltre possibile filtrare per paese, vedendo il numero e caratteristiche dei Relay in Italia.

---

<sup>6</sup>Se un utente usa la rete onion per scaricare contenuti con copyright, l'IP da cui è stato scaricato il file sarà quello del Relay e quindi le colpe ricadranno sul proprietario del server

<sup>7</sup>Non vengono inseriti nelle directory pubbliche Tor

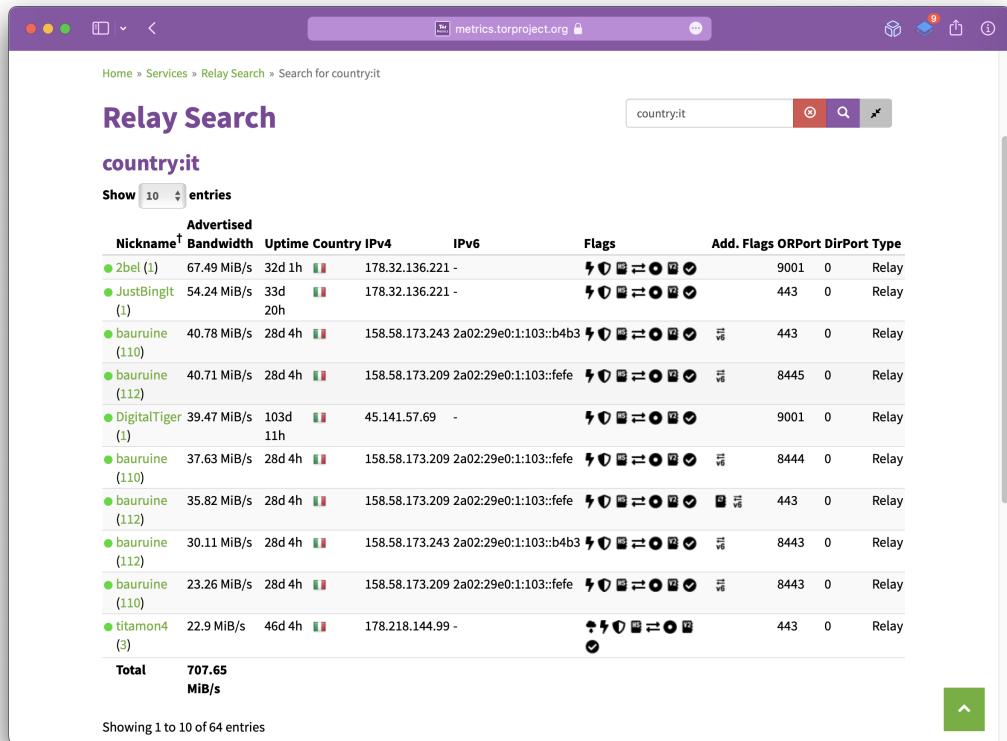


Figura 3.4: Statistiche dei Tor Relay in Italia

### 3.3 Directory Servers

I directory server sono un piccolo sottogruppo di onion router utilizzati per tracciare i cambiamenti nella topologia di rete. In particolare un directory server agisce come un server HTTP accessibile dai client per ottenere lo stato della rete e la lista dei router aggiornata periodicamente dagli stessi OR.

Quando il directory server riceve un aggiornamento da un OR prima di tutto controlla la chiave d'identità del router, garantendo che un attaccante non possa fingersi un OR manomettendo la rete [Cha81]. Il software di accesso alla rete onion è precaricato con le informazioni sui directory server e le relative chiavi, le informazioni di rete vengono aggiornate periodicamente dall'OP.

I directory server sono anche fondamentali per la connessione agli onion services, infatti ogni servizio creato genera un *Onion Service Descriptor* contenente una lista degli introduction points e la chiave pubblica, il pacchetto viene criptato con la chiave privata e inviato al directory server che quindi agisce come fosse un DNS server per gli onion services. Quando l'utente tenta la connessione a un sito onion richiede e riceve dal directory server il relativo *Descriptor* per l'indirizzo onion, usa la chiave pubblica, derivata dalla stringa dell'indirizzo onion a cui vuole connettersi, per decriptare il pacchetto.

Nel caso il Directory Server fosse compromesso e contenesse un Descriptor malevolo<sup>8</sup> generato per reindirizzare il traffico, l'indirizzo onion che possiede il client non sarebbe

<sup>8</sup>Che però non potrà essere stato criptato la stessa chiave privata

in grado<sup>9</sup> di decriptare le informazioni, dato che sono state criptate con una differente chiave privata. [Tord]. Non c'è neanche la necessità di determinare se le informazioni sono corrette, perché a patto che la chiave privata non sia resa pubblica, nessuno potrà manomettere il Descriptor senza renderlo illeggibile e/o "*indecifrabile*".

### 3.4 Tor Browser

Una delle principali vulnerabilità della rete Tor è la possibilità che un servizio web crei un codice JavaScript<sup>10</sup> malevolo in grado di ottenere informazioni sull'indirizzo dell'utente deanonymizzandolo. Gli sviluppatori di Tor Browser hanno inserito appositamente un sistema di sicurezza che blocca gli script, questo però di contro porta molti siti a non funzionare correttamente.

Essendo la rete Tor fortemente basata sulla criptografia la navigazione di un sito onion tramite Tor non fa comparire alcun avviso di sicurezza in caso di mancanza di HTTPS, dato che il traffico è già criptato. Da tenere inoltre in considerazione che un certificato proveniente da una Certificate Authority potrebbe generare problemi di anonimizzazione del servizio a causa della Certificate Transparency [Goo].

---

<sup>9</sup>tramite la chiave pubblica al suo interno nascosta

<sup>10</sup>che viene eseguito sul browser dell'utente

### 3.5 Dimostrazione da Wireshark

Ci sono diversi elementi che possono indicare l'esistenza di un circuito onion, innanzitutto esso viene creato sfruttando il protocollo TLS<sup>11</sup> tramite TCP, inserendo quindi TLS come filtro in Wireshark possiamo vedere i pacchetti con cui viene generato il circuito.

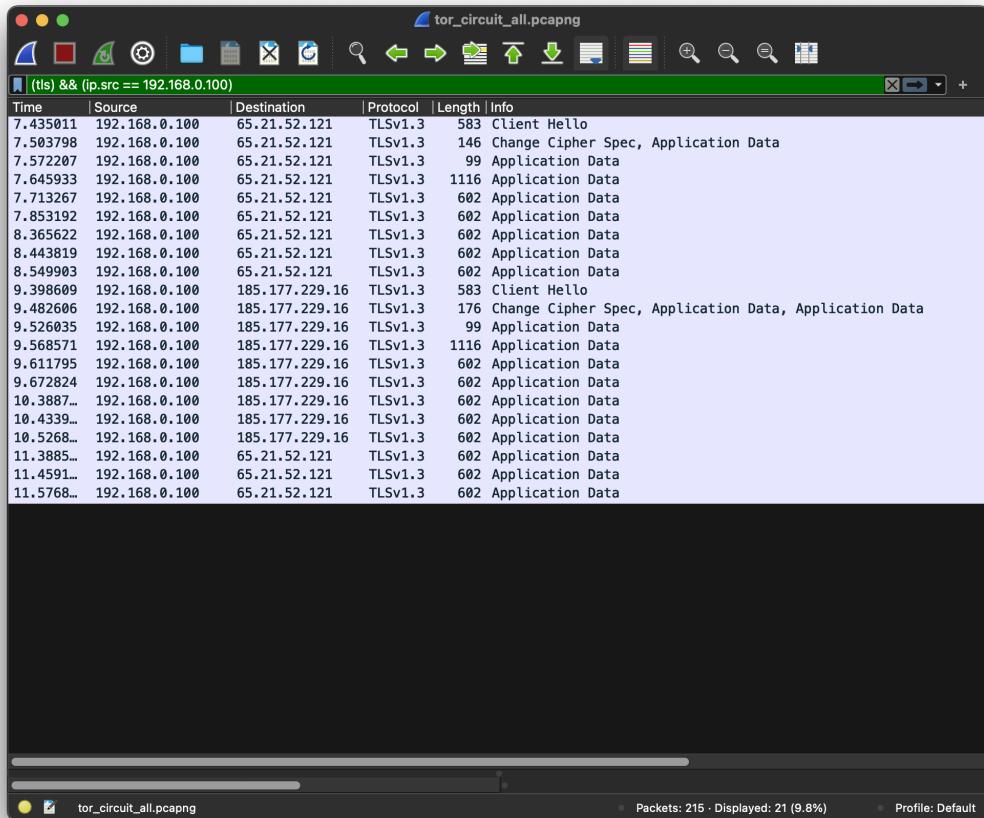


Figura 3.5: Wireshark circuit

Da qui vediamo che i due IP principali a cui il dispositivo si connette sono 65.21.52.121 e 185.177.229.16. Vediamo anche che Tor scambia con questi due IP lo stesso numero e tipo di pacchetti con la stessa quantità di byte, per cui vi è una correlazione. Eseguendo il filtro per IP possiamo quindi vedere tutti i pacchetti che i due dispositivi si scambiano.

<sup>11</sup>In particolare la nuova versione di Tor usa TLSv1.3

### *Dimostrazione da Wireshark*

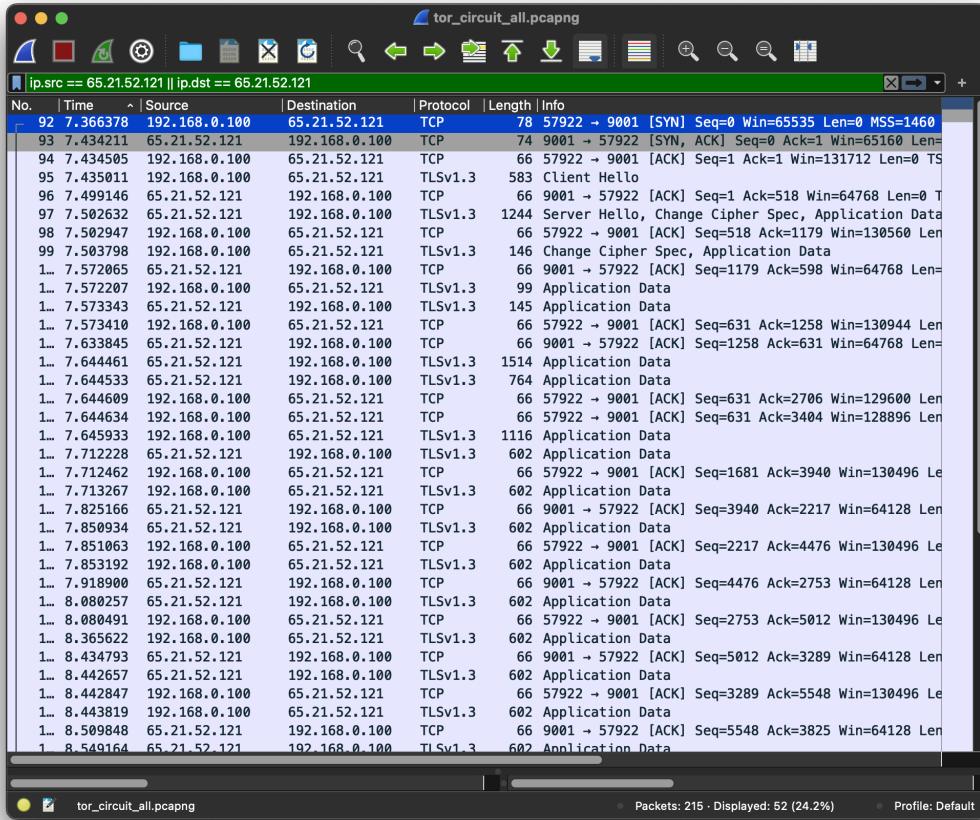


Figura 3.6: Wireshark IP filtering

Vediamo quindi che inizialmente viene eseguito un TCP handshake, successivamente inizia lo scambio di chiavi tramite TLSv1.3, possiamo inoltre notare che la porta con cui comuniciamo con il server 65.21.52.121 è la 9001<sup>12</sup>. A ogni messaggio TLS corrisponde una risposta ACK TCP che possiamo ignorare.

<sup>12</sup>Tor Project suggerisce di non usare la porta 9001 per i bridge relay perché è facilmente associabile alla rete Onion [Torb]

Analizziamo il primo pacchetto TLS che viene inviato al server, è denominato **Client Hello** e possiede vari campi:

- **Cipher Suites**, una lista di protocolli di criptografia simmetrica supportati dal client che invia la richiesta [IETF]. In questo caso i protocolli supportati sono 18, tra cui abbiamo AES 128 GCM SHA256 e AES 256 GCM SHA384.
- Compression Methods, implementato in TLSv1.3 solo per retro compatibilità dei server, che possono anche ricevere e gestire richieste TLSv1.2. A differenza dei client TLSv1.3 che devono impostare questo parametro a 0 (null), altrimenti riceveranno un illegal\_parameter message.
- **Extensions**, usato sia dalle applicazioni che sfruttano TLS, come onion, che da alcune funzionalità di TLSv1.3, implementate come estensioni per mantenere la retro compatibilità. In tal modo i server legacy possono semplicemente ignorare le estensioni non riconosciute.

[IETg]

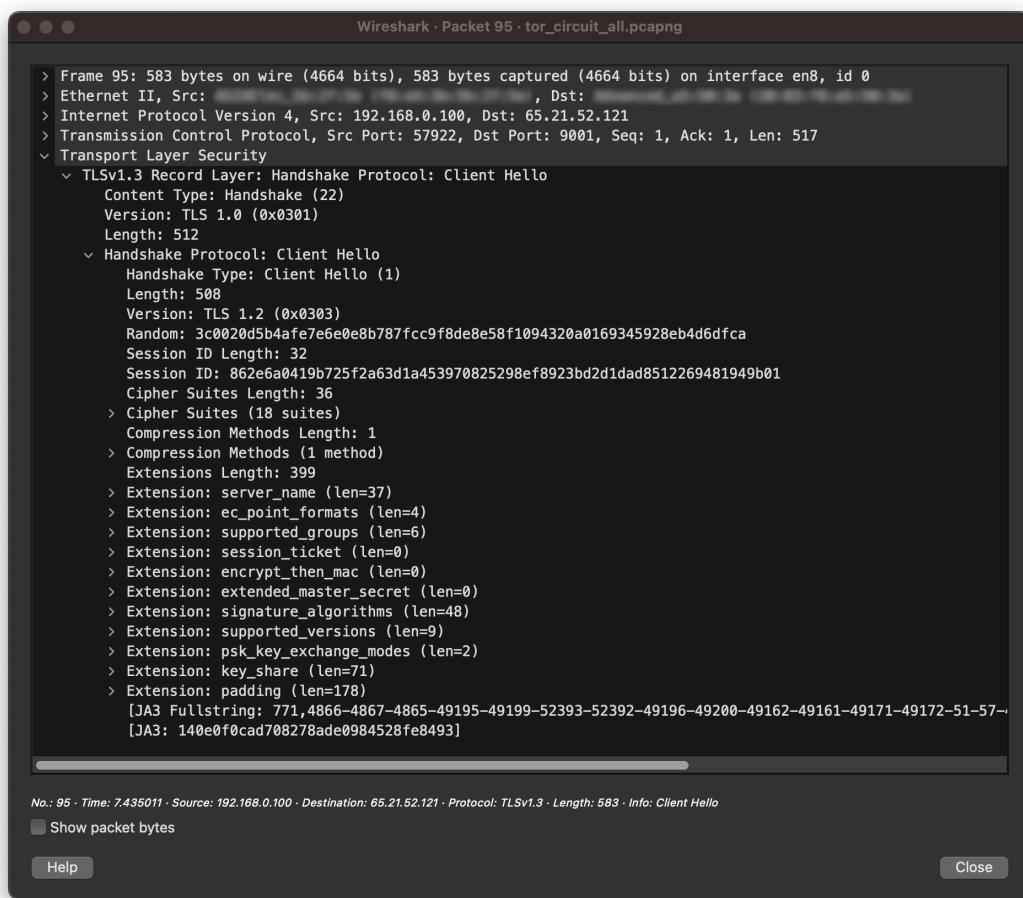


Figura 3.7: Client Hello

Come vediamo onion sfrutta diverse estensioni

- `server_name`, utilizzato per facilitare le connessioni sicure, indica il nome del server [IETa].
- `ec_point_formats`, sta per Elliptic Curve Point Formats e indica i formati di compressione (Point Formats) supportati dal client [IETd].
- `supported_groups`, precedentemente chiamato Supported Elliptic Curves Extensions, indica gli Elliptic Curves utilizzati ordinati in base alla preferenza del client [IETd].
- `session_ticket`, non usato.
- `encrypt_then_mac`, non usato.
- `extended_master_secret`, non viene più usato ma deve comunque essere indicato per retro compatibilità [IETb].
- `signature_algorithms`, utilizzato per indicare gli algoritmi di firma supportati dal client [IETe].
- `supported_versions`, utilizzato dal client per indicare le versioni di TLS supportate e dal server per indicare la versione utilizzata [IETe].
- `psk_key_exchange_modes`, utilizzato esclusivamente dal client per indicare le modalità di condivisione della passkey [IETe].
- `key_share`, utilizzato per inviare la chiave pubblica con cui criptare il traffico del server verso questo client, indicando anche il tipo di chiave [IETe].
- `padding`, utilizzato come una sequenza di 0 sufficienti per far raggiungere al pacchetto almeno 512 bytes di lunghezza. Il campo può essere omesso per i pacchetti di almeno 512 bytes. Con pacchetti di lunghezza tra 509 e 511 bytes questo campo porta il pacchetto a superare i 512 bytes, in quanto non può essere inferiore ai 4 bytes [IETc].

Successivamente il server risponde con un **Server Hello**, indicando un unico cipher suite e le estensioni `supported_versions` e `key_share`.

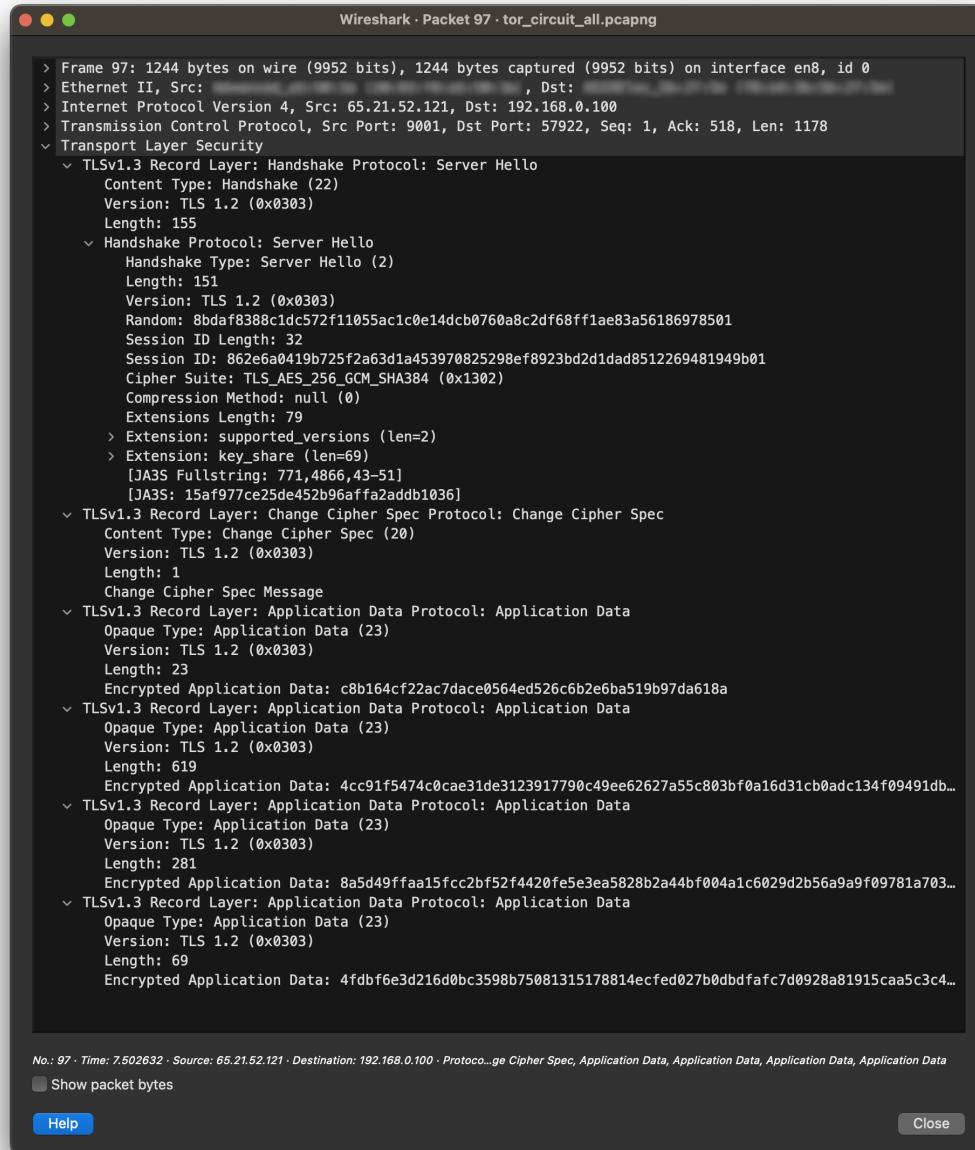


Figura 3.8: Server Hello e Application Data

Da qui la connessione viaggia per il circuito onion e tutti i dati sono criptati con le chiavi che possiede solo il client Tor, per cui non è possibile analizzare il traffico, possiamo solo intuire quali pacchetti sono destinati alla rete onion.



# 4. Implementazione

## 4.1 Servizi Onion

Nelle comuni reti internet i servizi vengono utilizzati facendo richieste basate sugli indirizzi IP pubblici, la sorgente e destinazione di un pacchetto è quindi conosciuta da tutti i dispositivi che lo ricevono. Onion invece garantisce l'anonimato non solo agli utenti ma anche ai servizi, nascondendone l'indirizzo IP. Questo viene fatto tramite uno pseudonimo di lungo termine, identico in tutti i circuiti e stabile anche al un fallimento di un router.

I principali obiettivi sono:

- Gli attaccanti non devono riuscire a manipolare la rete sostituendosi un servizio esistente. Questo livello di affidabilità deriva dalla criptografia asimmetrica che ci garantisce che il servizio a cui cerchiamo di connetterci sia autentico, solo lui possiede la copia privata della chiave pubblica con cui tentiamo la connessione.
- Sicurezza dagli attacchi DoS, viene fatto tramite l'uso di più punti d'ingresso alla rete.

La creazione di un servizio onion passa per diversi punti prima di poter essere raggiungibile:

1. Genera una coppia di chiave pubblica e privata per identificarsi.
2. Definisce alcuni onion router come punti di ingresso nella rete, da cui riceverà le richieste dei clienti e invia loro la chiave pubblica.
3. Crea un circuito con ogni punto di ingresso.
4. Invia ai Directory Server onion le informazioni sui punti d'ingresso e l'hash della chiave pubblica che sarà usata come hostname (Sec. 3.3).

Quando un utente tenta la connessione:

1. Inizialmente esegue il lookup del dominio tramite i Directory Server.
2. Successivamente sceglie un OR come tramite per il servizio e genera un circuito verso di esso, sfruttando uno specifico cookie per identificare il servizio.
3. Viene generato uno stream verso uno dei punti d'ingresso del servizio con tutte le informazioni riguardo se stesso, il nodo tramite e il cookie. Tutto viene criptato con la chiave pubblica del servizio.
4. Inizia l'handshake di Diffie-Hellman tra OP e servizio.

5. Il servizio onion a sua volta genera un circuito verso il nodo tramite per poter rispondere all'OP con la seconda parte dell'handshake.
6. Il nodo tramite collega i due circuiti generando un circuito dati bidirezionale.

Sia il programma dell'utente che il web server non vengono modificati e non sono nemmeno a corrente che il traffico viaggia per una rete onion [Cha81].

## 4.2 Onion V3

La terza generazione di onion nasce per risolvere alcuni problemi di sicurezza. In particolare rispetto alla seconda generazione:

- Sono stati aggiornati i sistemi di crittografia, da SHA1/DH/RSA1024 a SHA3/ed25519/curve25519.
- Sono stati migliorati i directory server e il directory protocol.
- È stato cambiato il sistema di hostname. Precedentemente venivano usati i primi 80 bit dell'hash (SHA1) della chiave pubblica per creare l'hostname<sup>1</sup>, nella terza generazione vengono codificati in base32.
  - La chiave pubblica, un totale di 32 byte in ed25519.
  - Il checksum di 2 byte.
  - Un byte di versione, di default '\x03'.

In totale il nuovo hostname possiede 56 caratteri<sup>2</sup>.

Inoltre l'indirizzo v3 contenendo l'intera chiave pubblica, una volta decodificato<sup>2</sup> in base32<sup>3</sup> ci ritorna tutte le informazioni necessarie per connettersi al servizio.

[Tor13]

---

<sup>1</sup>un esempio **yyhws9optuwiwsns.onion**

<sup>2</sup>esempio l'indirizzo *pg6mmjyjmcrsslvykfunntlaru7p5svn6y2ymmjubnubxndf4pscryd.onion*, se proviamo a decodificarlo otteniamo la stringa 79bcc625184b05194975c28b66b66b0469f7f6556fb1ac3189a79b40dda32f1f214703, come notiamo termina con 03, il byte di versione

<sup>3</sup>Definito nell'RFC 3548 e 4648

### 4.3 Studio delle tecnologie

Per creare un servizio sulla rete onion si possono usare una moltitudine di tecnologie differenti, ogni singolo aspetto ha necessità di effettuare delle scelte.

- Deploy del Server, la prima scelta ricade sulla creazione del server e in particolare scegliere dove eseguire il deploy. Potremmo fare il deploy in locale, in una qualsiasi macchina Intel/AMD ma ci sono molteplici motivi per usare un servizio cloud, tra cui la sicurezza, la facilità d'implementazione<sup>4</sup>. Ci sono 3 principali competitor e molti altri secondari che per la maggior parte sfruttano i server delle 3 aziende principali:
  - Google Cloud, il servizio cloud di Google, è il terzo più grande servizio cloud.
  - Microsoft Azure, il servizio cloud di Microsoft, è il secondo più grande servizio cloud.
  - **Amazon Web Service (AWS)**, il più importante servizio cloud, è gestito e reso disponibile da Amazon e possiede molti servizi tra cui scegliere. Per questa implementazione useremo il servizio **EC2** messo a disposizione da AWS<sup>5</sup> in una macchina *t3.micro*.
- Sistema Operativo, la seconda scelta ricade sul SO della nostra macchina. I principali *"concorrenti"* sono Windows Server e **Linux**, anche se la scelta migliore ricade su Linux:
  - Leggerezza, affidabilità e sicurezza derivate da un sistema GNU/Linux.
  - Maggiore controllo sul sistema operativo e personalizzazione di ogni aspetto, è anche possibile creare un sistema operativo Linux su misura per le proprie esigenze.
  - Maggiore utilizzo in ambienti server.
  - Nessun costo aggiuntivo derivato dall'acquisto di una licenza d'uso come Windows server.

Useremo **Debian** nella versione 11.

- Web Server, abbiamo due web server principali:
  - Apache httpd, il più vecchio dei due, rilasciato nel 1995 con licenza Apache 2.0.
  - **Nginx**, un web server più moderno che fornisce diversi altri strumenti oltre al web server, tra cui il proxy e load balancer. È stato rilasciato nel 2004, è gratuito e come httpd è open source, ha ufficialmente superato Apache nel luglio 2021 [Web]. Viene anche utilizzato in container Docker, ma noi lo useremo in una classica macchina virtuale.

---

<sup>4</sup>Per esempio il login da remoto in una macchina locale richiederebbe un IP statico e comunque non potremmo accendere e spegnere la macchina senza implementare sistemi esterni o nel migliore dei casi un Wake on Lan

<sup>5</sup>Gli altri servizi cloud messi a disposizione da Google o Microsoft forniscono strumenti simili, anche se la configurazione può essere differente

## 4.4 Creazione del servizio

Una volta creato l'account AWS ed eseguito l'accesso possiamo creare una macchina virtuale andando nella sezione EC2 → Instances → Launch Instances [Aws], possiamo creare la macchina in qualsiasi regione, in questo caso la andremo a creare a Milano *eu-south-1*. Durante la configurazione possiamo selezionare le opzioni che abbiamo precedentemente analizzato:

1. Il nome.
2. Il sistema operativo da usare, selezioniamo Debian 11 e in particolare l'architettura x86.
3. L'istanza scelta è una semplice t3.micro con 2 vCPU e 1 GiB di memoria, questo a dimostrazione che per eseguire un servizio onion non è necessario un server potente.
4. Una coppia di due chiavi pubblica e privata per la connessione SSH, è anche possibile riusare una coppia precedentemente creata e quindi usare le stesse chiavi per più server.
5. Successivamente è fondamentale aggiungere alla macchina i giusti security group, di default AWS non consente nessun tipo di traffico in entrata o in uscita dalla macchina.
  - Consentire la connessione SSH alla porta 22.

Type	Protocol	Port range	Source
SSH	TCP	22	0.0.0.0/0

Figura 4.1: security Group 1, SSH in entrata

- Consentire il traffico TCP in uscita.

IP version	Type	Protocol	Port range	Destination
IPv4	All TCP	TCP	0 - 65535	0.0.0.0/0

Figura 4.2: security Group 2, TCP in uscita

Una volta creata la macchina avremo la possibilità di scaricare la chiave privata con formato \*.pem che useremo per connetterci tramite SSH alla shell della macchina.

Listing 4.1: Connessione SSH

```
1 ssh -i key.pem admin@*.compute.amazonaws.com
```

La primissima operazione sarà l'aggiornamento dei repository e del sistema.

Listing 4.2: Aggiornamento del sistema

```
1 sudo apt update && sudo apt full-upgrade -y
```

Poi installiamo il web server nginx che abbiamo analizzato prima.

Listing 4.3: Installazione Nginx

```
1 sudo apt install nginx
```

Una volta completata l'installazione il web server viene automaticamente avviato e messo in ascolto sulla porta 80, in caso non fosse attivo:

Listing 4.4: Avvio di Nginx

```
1 sudo systemctl start nginx
```

Per installare Tor è innanzitutto necessario installare **apt-transport-https** per utilizzare i repository tramite HTTPS.

```
1 sudo apt install apt-transport-https
```

Dal comando `lsb_release -c` possiamo vedere la distribuzione corrente e inserire il repository corretto nella configurazione, con Debian 11 siamo su di una distribuzione bullseye, creiamo un file chiamato `tor.list`<sup>6</sup> nella cartella `/etc/apt/sources.list.d` dove andremmo ad aggiungere i repository TOR tramite le seguenti righe:

Listing 4.5: Tor Repository

```
1 deb [signed-by=/usr/share/keyrings/tor-archive-keyring.gpg] https://deb.torproject.org/torproject.org bullseye main
2 deb-src [signed-by=/usr/share/keyrings/tor-archive-keyring.gpg] https://deb.torproject.org/torproject.org bullseye main
```

Assicuriamoci che GPG sia installato nel sistema prima di aggiungere le chiavi.

Listing 4.6: Installazione GPG

```
1 sudo apt install gpg
```

Aggiungiamo le chiavi GPG della repository appena creata.

Listing 4.7: Aggiunta chiavi gpg dal repository Tor

```
1 wget -qO- https://deb.torproject.org/torproject.org/A3C4F0F979CAA22CDBA8F512EE8CBC9E886DDD89.asc | gpg --dearmor | tee /usr/share/keyrings/tor-archive-keyring.gpg >/dev/null
```

Aggiorniamo gli index e installiamo finalmente Tor.

Listing 4.8: Installazione Tor

```
1 sudo apt update && sudo apt install tor -y
```

[**Tora**]

Generiamo il file `torrc` contenente tutte le opzioni commentate e le relative descrizioni, ci servirà per impostare tutti i parametri di configurazione.

Listing 4.9: Generazione file torrc

```
1 sudo tor -f /etc/tor/torrc
```

---

<sup>6</sup>Il nome è configurabile

Apriamo il file `/etc/tor/torrc` e rimuoviamo il commento alle righe `HiddenServicePort 80 127.0.0.1:80` e `HiddenServiceDir /var/lib/tor/hidden_service`, il primo indica che il traffico arrivato dalla porta virtuale (80) viene reindirizzato alla porta 80 del localhost, ovvero la porta in cui è in ascolto il web server nginx, il secondo ci serve per indicare la directory in cui salvare l'hostname del sito e le chiavi crittografiche, o eventualmente la directory in cui sono già presenti.

Riavviamo Tor per aggiornare la configurazione, assicurandoci così che il file `torrc` non contenga errori e che il sistema sia funzionante.

```
1 sudo systemctl restart tor
```

#### 4.4.1 Gestire la comunicazione tramite i socket unix

A questo punto Tor ha creato gli introduction points e ha generato un circuito con ognuno di essi, ha inoltre generato le chiavi e il proprio hostname, queste informazioni sono state aggiunte nella cartella che abbiamo inserito nell'`HiddenServicePort`, in particolare il file chiamato `hostname` contiene l'indirizzo Tor del nostro servizio [**Tore**]. Questo sistema però non è completamente sicuro, infatti Tor è connesso con il web server tramite la porta 80, essendo entrambi i processi sulla stessa macchina il sistema ottimale è utilizzare un socket unix tra i due processi. Questo impedisce a un utente **non** Tor di accedere direttamente al web server senza dover configurare un firewall che comunque aggiunge complessità nella rete.

Apriamo il file `/etc/nginx/sites-enabled/default` e nella sezione `server` aggiungiamo il socket in ascolto, così che la comunicazione possa passare anche per il socket unix.

Listing 4.10: Aggiunta/creazione socket unix in nginx

```
1 listen unix:/var/run/website.sock;
```

Possiamo inoltre rimuovere la connessione dalla porta 80 commentando le seguenti righe.

Listing 4.11: Rimozione connessione dalla porta 80 in nginx

```
1 #listen 80 default_server;
2 #listen [::]:80 default_server;
```

Dopo aver riavviato nginx ed esserci assicurati che non vi siano errori apriamo il file `torrc` e inseriamo lo stesso socket.

Listing 4.12: Aggiunta socket unix a torrc

```
1 HiddenServicePort 80 unix:/var/run/website.sock
```

Infine riavviamo Tor.

```
1 sudo systemctl restart tor
```

Se tutto è andato a buon fine il web server non dovrebbe essere in grado di rispondere alle richieste HTTP dalla porta 80, ma solo a quelle provenienti dalla rete Tor tramite l'hostname `.onion` che abbiamo ricevuto precedentemente.

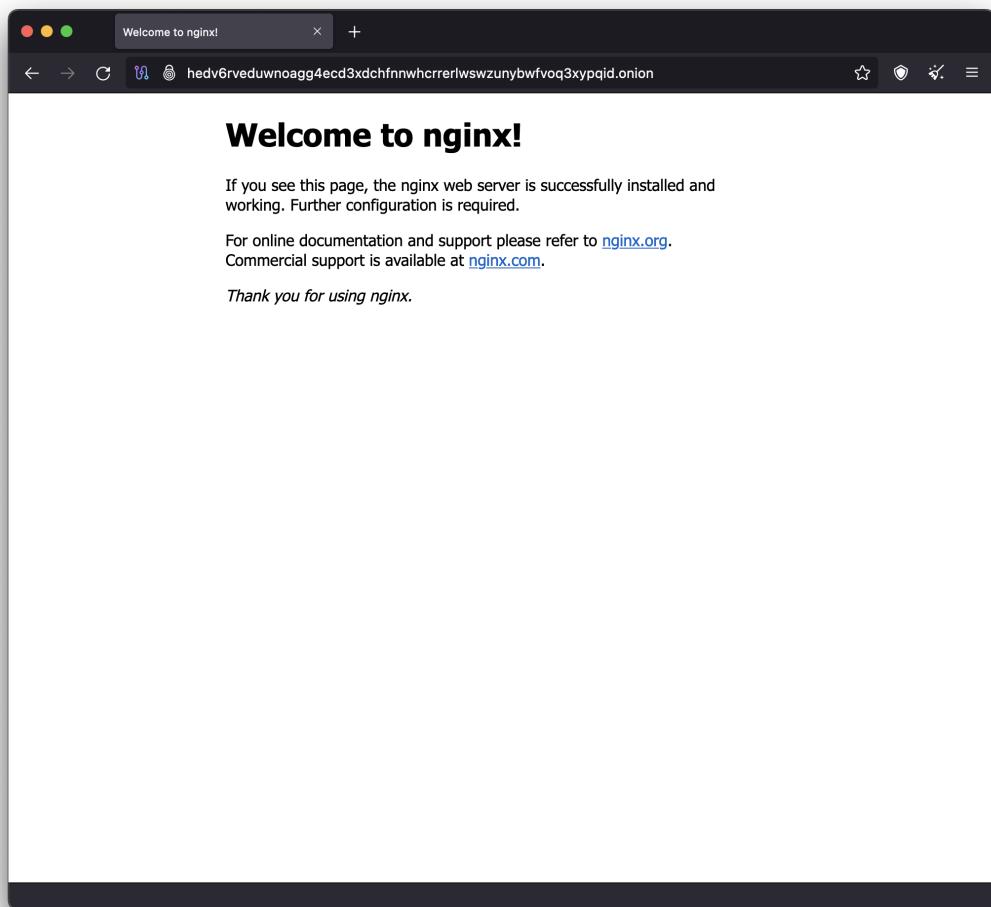
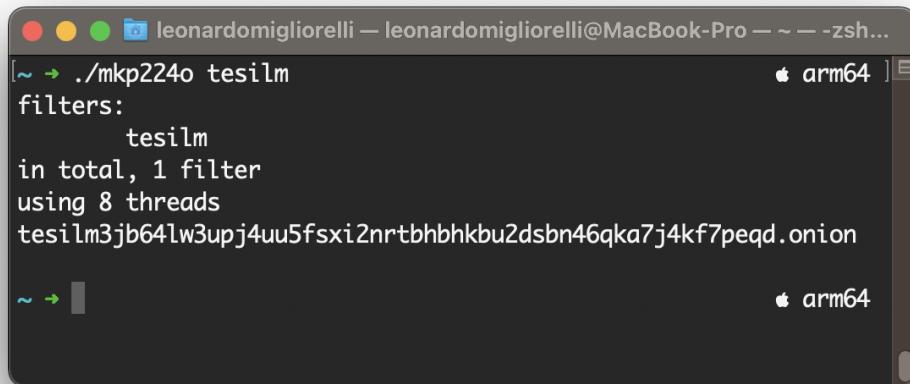


Figura 4.3: Connessione al web server nginx

#### 4.4.2 Creazione di un url personalizzato

Ora che il servizio è attivo e disponibile possiamo iniziare a configurarlo a nostro piacimento, la prima cosa che potremmo voler fare è cambiare l'hostname impostandone uno personalizzato, almeno in parte, un esempio è il sito email di Proton **proton-mailrmez3lotccipshtkleegetolb73fuirgj7r4o4vfu7ozyd.onion**.

Si sfrutta la tecnica del brute force per generare coppie di chiavi casuali da cui ottenere indirizzi *.onion* fino a che non ne troviamo uno che inizia con la stringa che abbiamo scelto, potrebbe impiegare diverso tempo ed è consigliabile usare un computer più potente della nostra macchina virtuale. Vi sono diversi strumenti per fare quest'operazione sia v2 che v3, per questa implementazione useremo il tool mfp224o [cat] che appunto ci permette di fare quest'operazione con gli indirizzi v3, possiamo installarlo su Linux o usarlo su Windows da riga di comando semplicemente indicando il nome con cui l'indirizzo dovrà iniziare.



```
leonardomigliorelli — leonardomigliorelli@MacBook-Pro — ~ — -zsh...
[~ → ./mfp224o tesilm
filters:
    tesilm
in total, 1 filter
using 8 threads
tesilm3jb64lw3upj4uu5fsxi2nrtbhbhkbu2dsbn46qka7j4kf7peqd.onion
~ → ]
```

Figura 4.4: Comando mfp224o e output

7

Da notare che più caratteri si scelgono nel matching più il software impiegherà nella generazione di un dominio. Otteniamo il nuovo indirizzo del servizio<sup>8</sup> con le relative chiavi.

Possiamo copiarle sul server con SCP usando l'opzione -i per definire il file d'identità, come facciamo con SSH.

```
1 cd tesilm3jb64lw3upj4uu5fsxi2nrtbhbhkbu2dsbn46qka7j4kf7peqd.onion
2 scp -i key.pem * admin@*.compute.amazonaws.com:~/var/lib/tor/
hidden_service
```

Avviamo la connessione al server e riavviamo Tor per impostare le modifiche.

<sup>7</sup>Il file eseguibile chiamato *mfp224o* viene avviato direttamente dalla cartella del codice sorgente, per cui apponiamo *./*

<sup>8</sup>tesilm3jb64lw3upj4uu5fsxi2nrtbhbhkbu2dsbn46qka7j4kf7peqd.onion

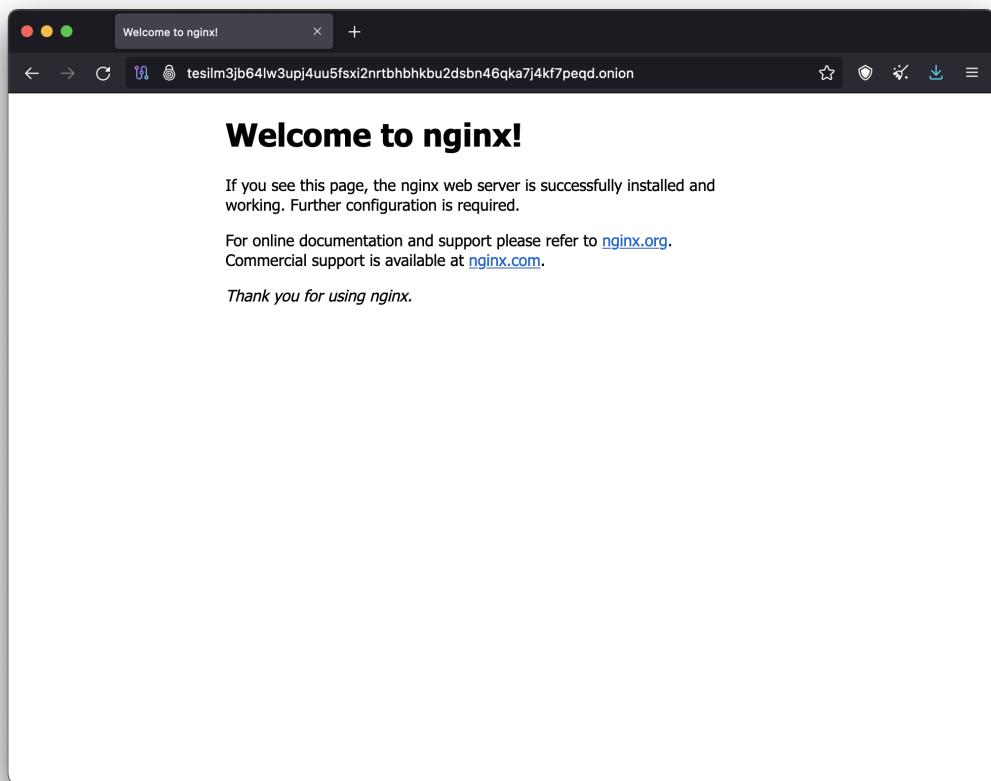


Figura 4.5: Connessione con il nuovo URL

#### 4.4.3 Far conoscere il sito

Un'altra caratteristica importante per ogni sito è la capacità di essere trovato dall'utente finale, solitamente tramite un motore di ricerca, ci sono diversi motori che eseguono l'indexing di indirizzi onion. Tra questi c'è notEvil<sup>9</sup> funziona come un classico motore di ricerca, a differenza di altri motori come Torch che invece esegue la ricerca più in base al contenuto che all'url<sup>10</sup>.

In NotEvil è possibile contattare gli amministratori del motore per richiedere l'aggiunta di un sito. Dalla pagina ufficiale è infatti presente il pulsante di contatto, dopo un breve controllo ci permette di inviare una richiesta anonima così che il sito possa essere analizzato e aggiunto.

Un altro sistema per far conoscere il sito agli utenti finali è l'utilizzo di siti specializzati come OnionDir<sup>11</sup> che permette di aggiungere il proprio sito in maniera completamente anonima.

In questo caso basta cliccare sul tasto Add Link nella toolbar in alto.

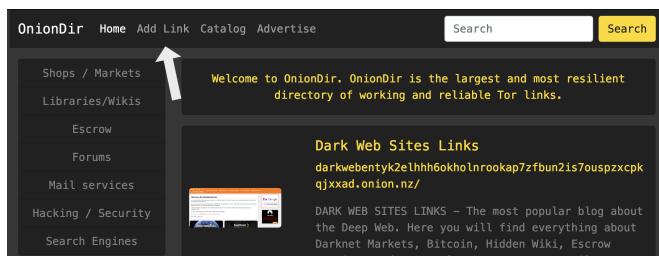


Figura 4.6: OnionDir, Pagina iniziale

A questo punto possiamo inserire l'URL e una piccola descrizione, come NotEvil il sito sarà scansionato per evitare truffe o contenuti proibiti, come indicato nelle poche righe di descrizione dei termini del servizio.

Figura 4.7: OnionDir, form aggiunta sito

<sup>9</sup><http://notevilmtxf25uw7tskqj6njlppebymlrerfv5hc4tuq7c7hilbyiqd.onion>

<sup>10</sup>Un test di ricerca di 'Proton mail' in entrambi mostra solo notEvil restituirci il sito corretto

<sup>11</sup><https://oniondiricuc4x2y5qbucg4jyp2ael5rxy7aahy5f4fbars2jkkf7vad.onion.nz>

Un altro sistema per far conoscere il sito è tramite il cosiddetto *Onion-Location Header* che permette di indicare al browser (e quindi all'utente) che il sito è disponibile anche tramite la rete onion, in questo modo quando la pagina viene visitata il browser può mostrare un pulsante per eseguire un redirect al sito onion.

Il miglior esempio è il sito ufficiale di [Tor Project](#) che mostra un pulsante per aprire il sito onion quando viene visitato tramite la rete Tor<sup>12</sup>.

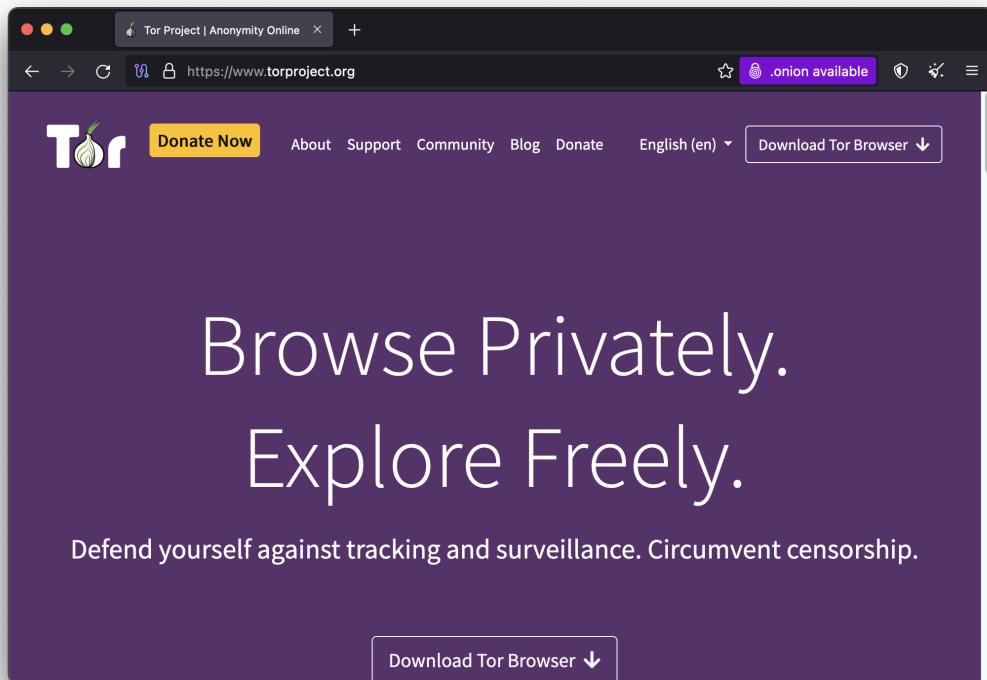


Figura 4.8: Tor Browser mostra il pulsante *.onion available* quando si visita il sito tramite Tor

Per aggiungere questo header è sufficiente aggiungere la seguente riga al file di configurazione del server.

```
1 add_header Onion-Location http://
    tesilm3jb64lw3upj4uu5fsxi2nrtbhbhkbu2dsbn46qka7j4kf7peqd.
    onion$request_uri;
```

<sup>13</sup>. Oppure possiamo aggiungere direttamente il tag `<meta>` nella sezione header della pagina HTML, nel caso di configurazioni particolari.

```
1 <meta http-equiv="onion-location" content="http://
    tesilm3jb64lw3upj4uu5fsxi2nrtbhbhkbu2dsbn46qka7j4kf7peqd.onion
    " />
```

<sup>12</sup>Da tenere in considerazione che questo metodo richiede un sito normale per funzionare

<sup>13</sup>Il tag `$request_uri` indica la sotto-pagina corrente, e quindi la pagina in cui andremo a fare il redirect, questa operazione non può essere fatta inserendo direttamente il tag all'interno del file HTML, in quanto non è possibile leggere l'url corrente della pagina

L'unica differenza con il metodo tramite configurazione è che il tag è statico e non varia in base alla pagina, per cui se vogliamo eseguire un redirect nello stesso file dobbiamo configurare separatamente ogni tag.

L'implementazione di questa seconda tecnica è stata mostrata nel mio sito [tesi.miglio.dev](https://tesi.miglio.dev) che mostra il pulsante per aprire il sito onion quando viene visitato tramite Tor.

[Torc]

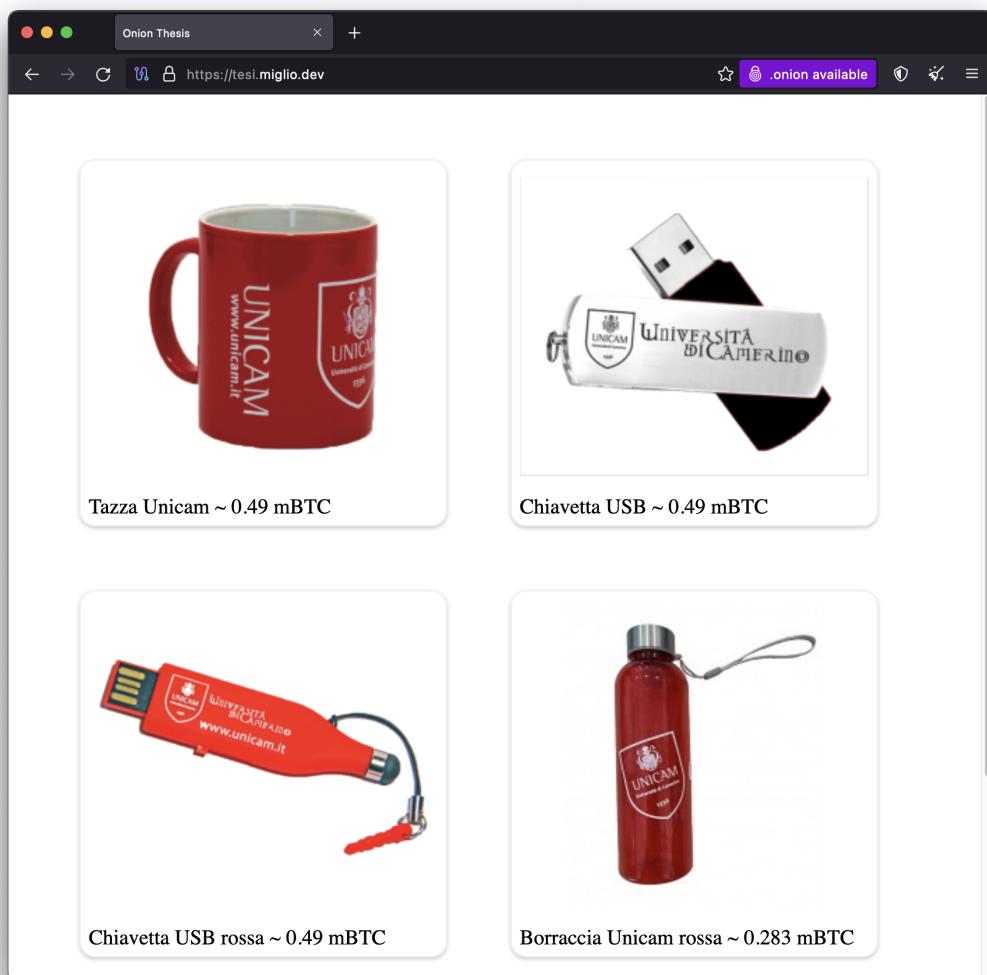


Figura 4.9: Tor Browser mostra il pulsante *.onion available* quando si visita il nostro sito tramite Tor

# 5. Sistemi di pagamento nella rete onion

La rete onion, come abbiamo visto, è fortemente basata sull'anonimato e sulla privacy. A differenza dei sistemi di pagamento tradizionali, che si basano su un sistema centralizzato e sulla forte identificazione<sup>1</sup> degli utenti.

## 5.1 Bitcoin

BTC è la cripto valuta più utilizzata nella rete onion, alcuni report [Blo] affermano che nei primi 4 mesi del 2020 il traffico di BTC nel dark web era di 795 milioni di dollari<sup>2</sup>, un aumento del 65% rispetto allo stesso periodo del 2019.

Creata nel 2007 da Satoshi Nakamoto<sup>3</sup> è la prima cripto valuta decentralizzata, fu pensata come una moneta virtuale assente da un ente centrale che ne controllasse la circolazione e la creazione.

Il sistema di transazioni BTC è basato sulla *blockchain*, un registro pubblico strutturato come una catena di blocchi, in cui ogni blocco rappresenta una transazione. Ogni transazione contiene la chiave pubblica del proprietario, l'hash del blocco precedente, la chiave privata e la firma digitale del precedente proprietario. Questo impedisce di modificare la catena, in quanto questo cambierebbe l'hash e invaliderebbe i blocchi successivi.

Una domanda comune è: *Come può un sistema basato su transazioni pubbliche essere usato in una rete basata sulla privacy?*, infatti chiunque può vedere le transazioni associate a un wallet semplicemente conoscendone l'indirizzo, non è però possibile risalire al proprietario perché nessun dato personale è associato all'indirizzo. Chi usa il proprio indirizzo nella rete TOR deve infatti stare attento a non associarlo alla sua persona, in quanto questo potrebbe essere usato per risalire alla sua identità. [Nak]

## 5.2 Creazione di un wallet BTC

Per accettare pagamenti nel nostro servizio onion abbiamo bisogno di possedere un indirizzo BTC. Ci sono wallet che possiamo usare per gestire i pagamenti, per questo esempio useremo Electrum<sup>4</sup>, un wallet open source e gratuito.

---

<sup>1</sup>Le banche sono infatti obbligate a chiedere i documenti dei correntisti al fine di tenere traccia dei loro patrimoni e dei movimenti di denaro

<sup>2</sup>Valore derivato dalla somma del traffico di denaro inviato e ricevuto da enti che operano nella rete TOR

<sup>3</sup>Uno pseudonimo, la vera identità di Satoshi Nakamoto è ancora sconosciuta

<sup>4</sup><https://electrum.org/>

## *Creazione di un wallet BTC*

una volta installato ed eseguito possiamo creare un nuovo wallet, scegliendo un nome, il tipo Standard Wallet, possiamo inoltre scegliere se usare un seed già esistente oppure generarne uno nuovo.

È inoltre importante salvare il seed in un luogo sicuro, in quanto questo è l'unico modo per recuperare il wallet in caso di perdita. Da qui possiamo creare un nuovo indirizzo BTC, cliccando su *Ricevi*, impostando la scadenza su Never e poi cliccando su *Crea Richiesta*, successivamente cliccando su Bitcoin URI possiamo visualizzare il solo indirizzo BTC.

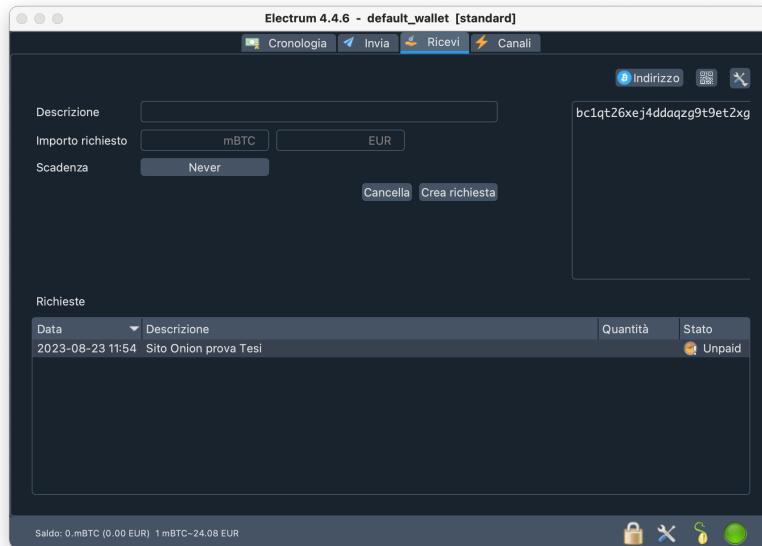


Figura 5.1: Pagina ricevi con indirizzo BTC

A questo punto possiamo copiare l'indirizzo e inserirlo nella pagina di pagamento del nostro servizio onion, oppure possiamo far generare un qrcode e inserirlo nella pagina.

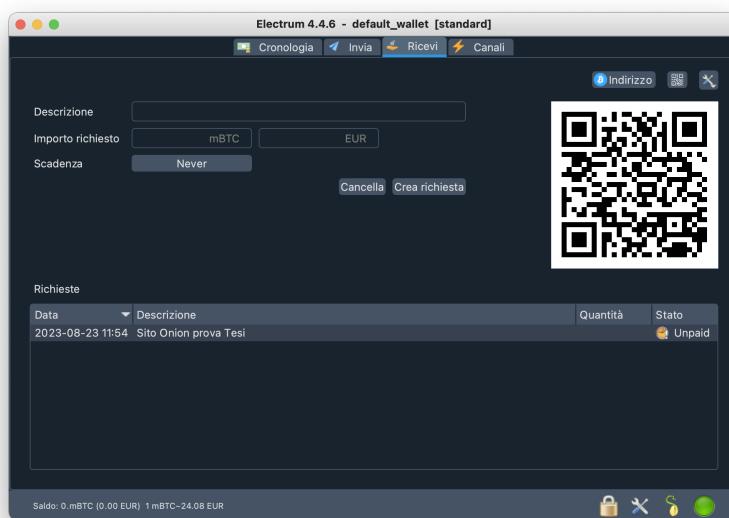


Figura 5.2: Pagina ricevi con qrcode

# Conclusioni

La tesi evidenzia come la rete Onion abbia un ruolo significativo nell'ambito della privacy e dell'anonimato online. Dalle sue primissime versioni fino alla più recente rete Tor, che è stata analizzata in questo lavoro, ha dimostrato di essere un mezzo efficace ed efficiente per proteggere la comunicazione e la navigazione degli utenti in un mondo in cui la privacy viene sempre più trattata come vera e propria moneta. D'altro canto l'uso delle reti Onion ha sollevato problemi legali e sociali riguardo all'identificazione di attività illegali sul web, dimostrando la necessità di definire un equilibrio tra privacy e prevenzione dei crimini.

È stata mostrata anche la facilità con cui è possibile creare un servizio Onion configurando il proxy Tor per reindirizzare il traffico dalla rete Onion verso un web server. Implementando anche elementi interessanti come la creazione di un dominio personalizzato e la gestione dei pagamenti.

L'argomento in assoluto più interessante durante la ricerca, stesura e studio della tesi è stato proprio l'analisi delle tecnologie sviluppate per consentire agli utenti di godere di una maggiore sicurezza online, assieme all'associazione del dominio alla coppia di chiavi asimmetriche necessarie per garantire l'anonimato.

## Sviluppi futuri

Nella scrittura del lavoro di tesi sono emersi 3 possibili sviluppi futuri:

- **Analisi di OnionShare**, è un software che permette di condividere file, inviare messaggi e creare un servizio onion in modo anonimo e sicuro sfruttando la rete TOR. Oltre a come sia possibile implementare un onion proxy all'interno di un'applicazione desktop per fornire un servizio in maniera anonima e sicura.
- **Analisi di alternative** come *I2P* e *Freenet*, come differiscono da TOR e come le diverse tecnologie implementate influenzano la sicurezza e l'anonimato degli utenti.
- Analisi di come la rete TOR venga usata dai cittadini di paesi dove vi è una forte censura e sorveglianza sulle attività svolte online, come delle implicazioni legali che gli utenti TOR potrebbero dover affrontare.



# Bibliografia

- [Aws] AWS Launch Instance. <https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances>:
- [Blo] Crystal Blockchain. *BTC activity on dark web*. <https://crystalblockchain.com/articles/onion-use-and-bitcoin-a-crypto-activity-report-by-crystal-blockchain/>.
- [cat] cathugger. *mkp224o - vanity address generator for ed25519 onion services*. <https://github.com/cathugger/mkp224o/>.
- [Cha81] David Chaum. «Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms». In: *ACM* (1981).
- [GD99] Syverson P. Goldschlag D. Reed M. «Onion Routing for Anonymous and Private Internet Connections». In: (1999).
- [Goo] Google. *Certificate Transparency*. <https://certificate.transparency.dev/>.
- [IETa] IETF. *RFC6066*. <https://www.rfc-editor.org/rfc/rfc6066>.
- [IETb] IETF. *RFC7627*. <https://www.rfc-editor.org/rfc/rfc7627>.
- [IETc] IETF. *RFC7685*. <https://www.rfc-editor.org/rfc/rfc7685>.
- [IETd] IETF. *RFC8422*. <https://www.rfc-editor.org/rfc/rfc8422>.
- [IETe] IETF. *RFC8446*. <https://www.rfc-editor.org/rfc/rfc8446>.
- [IETf] IETF. *RFC8448*. <https://www.rfc-editor.org/rfc/rfc8448>.
- [IETg] IETF. *TLSv1.3*. <https://www.rfc-editor.org/rfc/rfc8446#section-4.1.2>.
- [MGRG] Paul F. Syverson Michael G. Reed e David M. Goldschlag. «Anonymous Connections and Onion Routing». In: () .
- [Nak] Satoshi Nakamoto. «Bitcoin: A Peer-to-Peer Electronic Cash System». In: () .
- [RDos] Paul Syverson Roger Dingledine Nick Mathewson. «Tor: The Second-Generation Onion Router». In: (agosto 2004).
- [Tora] Tor. *enable official repo*. <https://support.torproject.org/apt/tor-deb-repo/>.
- [Torb] Tor. *Onion Bridge Configuration*. <https://community.torproject.org/relay/setup/bridge-ubuntu/>.
- [Torc] Tor. *Onion-Location header*. <https://community.torproject.org/onion-services/advanced/onion-location/>.
- [Tord] Tor. *Onion Services Overview*. <https://community.torproject.org/onion-services/overview/>.

## Bibliografia

---

- [Tore] Tor. *Setup Onion Service*. <https://community.torproject.org/onion-services/setup/>.
- [Torf] Tor. *Tor relay*. <https://community.torproject.org/relay/>.
- [Torg] Tor. *Tor relay metrics*. <https://metrics.torproject.org/rs.html>.
- [Torh] Tor. *Tor relay types*. <https://community.torproject.org/relay/types-of-relays/>.
- [Tor13] Tor. *Rendezvous Specification - Version 3*. 2013. <https://gitweb.torproject.org/torspec.git/spec-v3.txt>.
- [Web] *WebServer History*. [https://w3techs.com/technologies/history\\_overview/web-server/details](https://w3techs.com/technologies/history_overview/web-server/details)

# Listings

4.1	Connessione SSH . . . . .	30
4.2	Aggiornamento del sistema . . . . .	30
4.3	Installazione Nginx . . . . .	31
4.4	Avvio di Nginx . . . . .	31
4.5	Tor Repository . . . . .	31
4.6	Installazione GPG . . . . .	31
4.7	Aggiunta chiavi gpg dal repository Tor . . . . .	31
4.8	Installazione Tor . . . . .	31
4.9	Generazione file torrc . . . . .	31
4.10	Aggiunta/creazione socket unix in nginx . . . . .	32
4.11	Rimozione connessione dalla porta 80 in nginx . . . . .	32
4.12	Aggiunta socket unix a torrc . . . . .	32



# Elenco delle figure

2.1	Vita di un pacchetto onion . . . . .	7
2.2	Creazione del circuito onion . . . . .	8
3.1	Tor logo . . . . .	13
3.2	Differenza tra la creazione di un circuito onion e Tor . . . . .	14
3.3	TCP/IP stack with TOR network . . . . .	16
3.4	Statistiche dei Tor Relay in Italia . . . . .	19
3.5	Wireshark circuit . . . . .	21
3.6	Wireshark IP filtering . . . . .	22
3.7	Client Hello . . . . .	23
3.8	Server Hello e Application Data . . . . .	25
4.1	security Group 1, SSH in entrata . . . . .	30
4.2	security Group 2, TCP in uscita . . . . .	30
4.3	Connessione al web server nginx . . . . .	33
4.4	Comando mkp224o e output . . . . .	34
4.5	Connessione con il nuovo URL . . . . .	35
4.6	OnionDir, Pagina iniziale . . . . .	36
4.7	OnionDir, form aggiunta sito . . . . .	36
4.8	Tor Browser mostra il pulsante <i>.onion available</i> quando si visita il sito tramite Tor . . . . .	37
4.9	Tor Browser mostra il pulsante <i>.onion available</i> quando si visita il nostro sito tramite Tor . . . . .	38
5.1	Pagina ricevi con indirizzo BTC . . . . .	40
5.2	Pagina ricevi con qrcode . . . . .	40



# **Ringraziamenti**

In primo luogo vorrei ringraziare il Professore Fausto Marcantoni, non solo per avermi dato la possibilità di lavorare a questa tesi, ma anche per avermi fatto appassionare alla materia e ai suoi insegnamenti.

Ringrazio i miei genitori, per avermi sempre sostenuto ed insieme ai miei nonni per avermi permesso di studiare e di arrivare fino a questo punto.