POLITECNICO DI TORINO

Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea Magistrale

Infrastruttura di un sistema riservato e anonimo di trasporto con rotte dinamiche e aleatorie

Relatore

Prof. Guido Marchetto

Candidato

Cosimo Iacomini

ANNO ACCADEMICO 2019-2020

Ringraziamenti

"When I was a young boy, I had a dream. A dream without ambitions,
a dream fueled by insecurity, but I knew within myself,
that I was prepared to make the sacrifices, whatever the cost.
No matter how many mistakes I maked in my life, I kept on pushing
but I did one thing: I always took knowledge from my mistakes,
because without mistakes, you'll never discover perfection.
It doesn't matter how you'll get there, as long as you get there.
If you all have a dream? Then you have duty and responsability
to yourself to make it come true.
We can't make it come true, it's down to you."

Marco Pierre White

Prima di tutto, desidero esprimere i miei ringraziamenti al prof. Dr. Luís Mengual Galán, tutor di questa tesi magistrale, per la dedicazione, il tempo e l'aiuto che mi ha offerto durante la realizzazione. Ringrazio il prof. Guido Marchetto per la disponibilità e il tempo che mi ha dedicato in questo progetto.

Ringrazio il Politecnico di Torino per la opportunità che mi ha dato di terminare i miei studi attraverso il programma Erasmus. Ringrazio la Universidad Politécnica de Madrid per avermi accolto e insegnato un'altra forma di apprendere.

Ringrazio specialmente la mia famiglia per il suo sforzo nell'offrirmi tutti i mezzi e risorse necessarie durante questi 5 anni. Inoltre, la ringrazio per aver creduto in me e avermi motivato affinché non mi arrendessi nei momenti più duri e continuassi a lottare.

Ai miei amici, che hanno reso indimenticabile questo anno a Madrid: Antonio, Alessio, Aurora, Danielino, Diletta, Francesca, Giorgia, Nunzio, Roberta y Valerio. I momenti che abbiamo condiviso non li dimenticherò mai.

Ringrazio specialmente il mio amico Maurizio, per avermi accompagnato durante tutto l'anno e per essere stato una guida in questa esperienza. Madrid non sarebbe stata la stessa senza di lui.

Le mie ultime parole sono di ringraziamento verso i miei amici da una vita di Matera: Angelo, Antonio Bon, Antonio Log e Gianluca. Ora per ragioni diverse viviamo in città lontane, però loro sono la prova che dimostra che l'amicizia può sopravvivere con la distanza.

Sommario

La tesi magistrale che si presenta in questo documento consiste nello sviluppo di un'infrastruttura di un sistema riservato e anonimo di trasporto con rotte dinamiche e aleatorie. Ciò vuol dire che permette agli utenti di trasmettere dati ad un altro estremo garantendo l'anonimato, la privacy delle connessioni e l'efficienza nell'invio e nella ricezione dei dati. In altre parole, l'entità destinataria ottiene un messaggio riservato e autenticato, che però non include nessun metadato associato come per esempio l'indirizzo IP, la posizione, ecc.

Si fa uso di una rete intermedia composta da diversi router che inviano e ricevono messaggi di un protocollo creato appositamente per questo progetto. Inoltre, c'è un'entità chiamata Sistema Gestore delle Rotte (SGR), che svolge le attività più importanti quali inviare il first hop all'utente che vuole inviare dati anonimamente e calcolare le rotte ottime affinché i messaggi viaggino per le route più corte e soprattutto differenti. Si dà anche la possibilità di gestire i dati con connessioni: autenticate per mezzo di certificati digitali; affidabili con protocolli orientati a connessioni che controllano l'interscambio di pacchetti di dati; sicure poiché l'informazione è cifrata con algoritmi crittografici; rapide avendo meccanismi di sincronizzazione e instradamento ottimizzati.

I meccanismi di sicurezza che si incorporano nell'applicazione si ispirano a OpenSSL, che è un pacchetto di librerie relazionate con la crittografia. Questi servizi aiutano a implementare il protocollo di sicurezza SSL/TLS. Offrono anche la possibilità di creare certificati digitali firmati da un'Autorità di Certificazione, i quali servono come piattaforma di cifratura per le connessioni tra i dispositivi. Con tutto questo, si ottiene la riservatezza dei dati da trasferire, l'identificazione dei partecipanti per impedire il furto di identità e la sicurezza di tutti gli implicati contro gli attacchi di terze parti.

L'infrastruttura è stata realizzata nel linguaggio di programmazione Java. Il miglior vantaggio di questo linguaggio è l'essere multi-piattaforma, inoltre, la gestione di thread e socket è molto facile e include una libreria per utilizzare OpenSSL.

Abstract

The Master's Final Project presented in this document consists of the development of an infrastructure of a confidential and anonymous transport system with dynamic and random routes. In other words, users are allowed to transmit data to another end, guaranteeing anonymity, privacy of connections and efficiency in data encryption and reception. The recipient entity obtains a confidential and authenticated message but without including any associated metadata such as IP address, location, etc.

Using an intermediate network, made up of different routers, it sends and receives messages from a protocol created especially for this job. In addition, there is another entity called Routes Management System (RMS) that performs the most important activities, such as sending the first hop to the user who wants to send data anonymously and calculate the optimal routes, so that every time messages take the shortest route and in a different way. It also gives the possibility of managing the data with connections: authenticated by means of digital certificates; reliable with connection-oriented protocols that monitor the exchange of data packs; safe, since the information is encrypted with cryptographic algorithms; and fast with optimized synchronization and routing mechanisms.

The security mechanisms that are incorporated into the application are inspired by OpenSSL, which is a package of libraries related to cryptography. These services help to implement the SSL/TLS security protocol. It also offers the possibility of creating digital certificates signed by a Certificate Authority, which serve as an encryption platform for connections among devices. With all this, we achieve the confidentiality of the data to be transferred, the identification of the participants to prevent impersonations and the security of all those involved in attacks by third parties.

The infrastructure has been made in the Java programming language. The best advantage of this language is that it is multiplatform, besides the handling of threads and sockets, it is very easy to manage and it includes a library to use OpenSSL.

Indice

Indice delle figure

Indice delle tabelle

Capitolo 1

Introduzione e Obiettivi

1.1 Introduzione

Intrinsecamente ai vantaggi che porta una rete di comunicazione sorgono problemi, relazionati con la sicurezza e la privacy, ai quali si deve trovare una soluzione. La ragione per la quale emerge attualmente l'interesse nel costruire sistemi di comunicazioni la cui priorità sia preservare la sicurezza dell'informazione personale e salvaguardare l'identità degli interlocutori è soddisfare la legislazione vigente. Come si può leggere in essa, ogni individuo deve essere informato e dare il suo consenso a qualunque operazione che implica il trattamento di dati di carattere personale che si trovano salvati in un supporto informatico. Nel caso di Internet è specialmente incriminante, posto che tutti quei dati possono essere acquisiti, conservati e utilizzati da terze parti per attività non autorizzate o illecite.
A causa della natura tecnica dei protocolli di comunicazione esistenti oggigiorno, quando ci connettiamo a una rete, sia per uso privato che pubblico, oltre ai dati utili, trasmettiamo informazioni di controllo e gestione che ci identificano. Semplicemente con l'accesso alla rete, in modo trasparente all'utente, indichiamo il computer dal quale si realizza la connessione attraverso un indirizzo IP assegnato, informiamo riguardo il sistema operativo che possediamo, e in più, forniamo le nostre preferenze di navigazione e le applicazioni di cui facciamo uso.

Nello sviluppo di applicazioni distribuite è essenziale l'incorporazione di servizi di sicurezza come l'identità e l'autenticazione degli utenti che intervengono, la riservatezza o l'integrità delle comunicazioni. Tuttavia, in alcune occasioni può essere un requisito delle applicazioni distribuite l'occultamento di alcuni dei metadati propri della comunicazione come può essere l'indirizzo IP dell'utente, la sua posizione

fisica, il sistema operativo, i router di transito ecc.
Se vogliamo occultare i metadati associati a una comunicazione una delle alternative che si hanno è utilizzare le reti anonime. Il problema di queste reti commerciali (per esempio Tor, I2P e Freenet) è che non si conoscono gli algoritmi interni di instradamento. Inoltre, bisogna fidarsi dei fornitori di queste reti in riferimento alla custodia delle route delle comunicazioni stabilite.

Nel caso di cui ci si occupa si punta per i cosiddetti sistemi anonimi come pilastro base della nostra soluzione, i cui fondamenti teorici contribuiscono a dare una risposta ai problemi di sicurezza e privacy presenti nelle reti di comunicazione. Avvalendosi dei concetti quali l'anonimato, la riservatezza o l'identità e, sfruttando i meccanismi di sicurezza di recente attuazione, come i certificati digitali, e le tecniche di instradamento con algoritmi dinamici, si cerca di montare una infrastruttura sicura e efficiente.

Ciononostante, le proposte pubblicate fino ad oggi sull'ottimizzazione delle comunicazioni anonime in quanto ai livelli di anonimato e latenza non hanno soddisfatto le necessità dei sistemi anonimi, fatto per il quale ancora non si implementano con gran successo. Così si spiega che è venuto fuori un crescente interesse nel trovare modi che permettano l'invio anonimo di messaggi, minimizzando i tempi di risposta e massimizzando gli elementi di sicurezza.
In questo contesto il progetto proposto consisterà nello sviluppo di una rete anonima formata da vari nodi nella quale i costi per arrivare a un nodo qualsiasi siano aleatori e, pertanto, le route tra un'origine e una destinazione non siano predicibili e nemmeno durature nel tempo. L'infrastruttura si integra con: connessioni TLS tra qualunque nodo della rete anonima e nell'accesso; servizi di riservatezza e non ripudio nelle proprie comunicazioni.
Per permettere l'intercomunicazione di computer che si trovano in zone geografiche lontane e non predeterminate, si disporrà di una rete interna di nodi trasmittenti, il cui flusso di dati sarà gestito efficientemente da un algoritmo di routing che calcolerà le rotte ottime anche la rete non si saturi o congestioni. Allo stesso tempo, esisterà un meccanismo che informi riguardo lo stato della rete e lo aggiorni in tempo reale affinché l'utente abbia accesso istantaneo all'informazione.

Partendo dalla base che non è possibile implementare una struttura perfetta che sia immune a qualunque circostanza anomala o di pericolo, si progettano tecniche e procedimenti mirati all'analisi esaustiva dei sistemi di comunicazione.
Questo si fa prima di integrarli per prevenire, o per lo meno, diminuire l'impatto negativo che avrà un attacco o un'intrusione in una rete di computer da terze parti

in modo che ci permettano di utilizzare le loro risorse senza limitazioni di nessun tipo.
Bisognerà misurare il grado con il quale il sistema realizza tutte le attività assegnate e verificare il suo comportamento fronte a condizioni avverse o di errore. Per saper se il programma è sufficientemente robusto e sicuro affinché entri in funzionamento dentro un ambiente reale, sarà imprescindibile sottometterlo a diverse prove.
Una parte importante del software è il suo ciclo di vita, il quale, è un chiaro indicatore di quanto bene o male è stato sviluppato un progetto, posto che quanto più lungo sia questo, maggiore redditività si otterrà con l'inversione. In termini quantitativi si deve prestare attenzione al costo di aggiornamento e mantenimento, alla soglia di miglioria e alla capacità di ampliamento del sistema.
Un ultimo aspetto da recensire è quello inerente all'esperienza d'uso. Se un cliente non si trova comodo con un'applicazione per circostanze tali che non è intuitiva o il suo progetto è poco attrattivo o non risponde con la celerità raccomandabile; per quanto forte, utile ed efficiente possiamo pensare che sia, sarà condannato al fallimento.

Avendo tutti questi aspetti in conto, l'obiettivo è sviluppare un progetto con il quale ottenere un prodotto finale che riunisca tutti i requisiti richiesti. Avremo un programma che somministrerà un insieme di servizi per poter stabilire una rete di comunicazione multi-punto, che si caratterizzerà per i protocolli incorporati per dotare il sistema di affidabilità e sicurezza.

1.2 Obiettivi del progetto

L'obiettivo fondamentale di questo progetto è la realizzazione di una infrastruttura di un sistema riservato e anonimo di trasporto con rotte dinamiche e aleatorie. All'interno di questo obiettivo generale si includono i seguenti obiettivi specifici:

- Introduzione ai servizi di sicurezza e reti anonime.
- Creazione di una rete anonima e sviluppo dell'algoritmo di gestione delle rotte dinamiche e aleatorie.
- Creazione dell'infrastruttura di rete per la connessione remota delle entità cliente, destinataria e nodi della rete anonima.
- Implementazione dei servizi di sicurezza dentro la rete anonima, nell'accesso ad essa e nei messaggi inviati per un'entità cliente.
- Verifica del sistema, test di rendimento.

Per raggiungere questi obiettivi menzionati, l'infrastruttura che si vuole sviluppare sarà composta dalle seguenti 4 entità:

Cliente : l'entità cliente originaria de la comunicazione.

Destinazione : l'entità destinataria ricevente della comunicazione.

SGR : il Sistema Gestore delle Rotte (SGR) incaricato della generazione aleatoria e dinamica delle rotte tra un origine e una destinazione.

Rete : i nodi della rete anonima che comunicheranno con il SGR per stabilire dinamicamente un cammino tra origine e destinazione.

I passi da seguire per inviare un messaggio anonimo sono i seguenti:

1. Internamente i nodi della rete anonima trasmettono il loro stato al SGR.
2. L'entità cliente si connetterà al SGR e otterrà un ticket digitale che le permetterà di conoscere il nodo di accesso alla rete anonima.
3. Il SGR stabilisce in maniera aleatoria i costi e le rotte e invia le tabelle di routing ai nodi che devono utilizzarle per un periodo di tempo.

4. Finalmente, l'entità cliente invia i suoi dati all'entità destinataria entrando nella rete anonima dal nodo ottenuto dal SGR.

5. L'entità destinazione riceve un messaggio riservato e autenticato che non include nessun metadato associato all'indirizzo IP, alla posizione ecc.

Dal cliente alla destinazione i messaggi viaggiano per la rete realizzando connessioni TLS tra qualunque nodo della rete anonima e nell'accesso ad essa. I messaggi inviati tra il SGR e il cliente e tra il SGR e i nodi sono cifrati usando UDP.

Per lo sviluppo di quello che è stato commentato si utilizzeranno le librerie di Java JCE (Java Criptography Extension) e JSSE (Java Secure Sockets Extension), e la piattaforma OpenSSL per la creazione di certificati di ogni entità dell'infrastruttura.

1.3 Struttura del documento

Questa tesi magistrale si compone del presente documento e dell'infrastruttura di un sistema riservato e anonimo di trasporto con rotte dinamiche e aleatorie sviluppata, il cui codice sorgente sarà allegato in un capitolo successivo per la sua spiegazione. Questo documento è organizzato in diversi capitoli il cui contenuto e scopo si descrive brevemente in seguito:

Capitolo 1 - Introduzione e obiettivi : Include la giustificazione della necessita di sviluppare il lavoro invece di acquisire o applicare direttamente ciò che esiste in tutta la gamma di categorie di processi, prodotti e servizi della società o istituzione che utilizza il progetto e i suoi obiettivi.

Capitolo 2 - Stato dell'arte : Include un'analisi dell'ambito di business in cui è inquadrato il progetto, le sue abitudini e le esigenze di prodotti o servizi tecnologici.

Capitolo 3 - Sviluppo : Spiega la metodologia seguita per lo sviluppo del sistema e le particolarità della soluzione adottata.

Capitolo 4 - Risultati : Vengono analizzati il funzionamento e le funzionalità del sistema sviluppato giustificando che la tecnologia risultante del progetto soddisfa i desideri e le necessità del cliente (reale, potenziale, utile).

Capitolo 5 - Conclusioni : Si sollevano conclusioni sui risultati ottenuti.

Capitolo 6 - Linee future : Si analizzano i limiti delle tecnologie utilizzate e si sollevano possibili migliorie del sistema sviluppato.

1.4 Normativa applicabile

La presente tesi magistrale è stata sviluppata nella Escuela Técnica Superior de Ingenieros Informáticos de la Universidad Politécnica de Madrid, a seguito della partecipazione al programma Erasmus+.
Dato che nel campo dell'informatica esistono diversi tipi di progetti, la tesi in questione è uno dei progetti di investigazione che esplorano o creano un algoritmo, un teorema o un metodo innovativo in risposta allo stato dell'arte. Si tratta di una tesi magistrale con Modalità B della sopracitata università, soggetta alla normativa applicabile da essa, così come alla normativa specifica del titolo citato. Per tesi magistrale con Modalità B spetta un carico di lavoro di 30 cfu che sono stati divisi nei seguenti punti:

1. Comprensione dei principi basici di sicurezza: servizi, meccanismi, protocolli e infrastruttura di reti anonime.
2. Comprensione e analisi di ambienti di sicurezza e delle librerie JCE di Java e di OpenSSL.
3. Progetto e sviluppo del sistema anonimo: infrastruttura di comunicazione tra i nodi.
4. Implementazione dell'algoritmo di gestione delle rotte dinamiche e aleatorie.
5. Sviluppo dell'infrastruttura di comunicazione tra i nodi del sistema anonimo e il SGR.
6. Sviluppo dell'infrastruttura di comunicazione tra il sistema cliente e destinatario utilizzando la rete anonima.
7. Incorporazione dei servizi di sicurezza nella rete anonima e nell'accesso.
8. Incorporazione dei servizi di riservatezza e non ripudio nei messaggi del cliente.
9. Sviluppo dell'interfaccia del cliente, del SGR e dell'entità destinataria.
10. Valutazione e test.
11. Documentazione.

Capitolo 2

Stato dell'arte

Questo capitolo mostra le basi teoriche della comunicazione anonima come concetti di sicurezza della rete, tecniche di anonimato inventate e sistemi di anonimato creati finora.
Oggi le persone usano Internet per svolgere la maggior parte delle attività relative al computer, come l'invio di messaggi, l'invio di e-mail, la ricerca di ciò a cui sono interessati e lo scambio di dati con agenzie pubbliche e/o private [1]. Queste ultime cercano di aumentare l'interazione elettronica al fine di controllare le informazioni su un utente. Alcuni esempi sono il consumo quotidiano, lo stile di vita, le opinioni, l'interazione con altre persone, le preferenze.

D'altro canto, sono state proposte tecnologie allo scopo di massimizzare la privacy (Privacy Enhancing Technologies) di un utente, di un gruppo di persone o di un'organizzazione. Questo tipo di tecnologia può aiutare le organizzazioni a rispettare i principi di protezione della privacy stabiliti nei diritti umani [2], offrendo agli utenti un maggiore potere di controllo delle loro informazioni e possono decidere come e quando devono essere usate da terzi.
Oggi esistono applicazioni come e-mail e browser Web anonimi che garantiscono la comunicazione senza il rilevamento dell'identità. I sistemi di gestione delle identità potenzialmente consentono alle persone di accedere a servizi e risorse senza dover fornire informazioni al riguardo. Pertanto, gli utenti dovrebbero avere fiducia in queste organizzazioni che devono verificare l'identità degli utenti e generare certificati elettronici che non contengono informazioni sull'identità e che consentono l'accesso ai servizi offerti da terzi.

Un altro tipo di tecnologia che aumenta la privacy sono quelli che estendono la protezione come:

- I sistemi di accesso biometrico crittografato che consentono l'uso delle impronte digitali.
- Accesso sicuro ai dati personali degli utenti online.
- Programmi che consentono ai browser di rilevare automaticamente le politiche sulla privacy dei siti Web e confrontarle con le preferenze espresse dagli utenti.
- Sistemi di allerta e avvertenza collegati alle stesse informazioni e che ne impediscono l'utilizzo in caso di non conformità alle politiche sulla privacy.

La terminologia proposta in [3] è la più usata ed è menzionata in [4, 5, 6, 7, 8, 9, 10].

2.1 Terminologia della comunicazione anonima

Le parole più importanti di questa parte di tesi sono: anonimato, non relatività, non osservabilità, pseudo-anonimato e gestione dell'identità.
Lo scenario tipico nel quale si utilizzano queste parole è caratterizzato da un emittente e un ricevente che utilizzano una rete di comunicazione per trasmettere un messaggio.

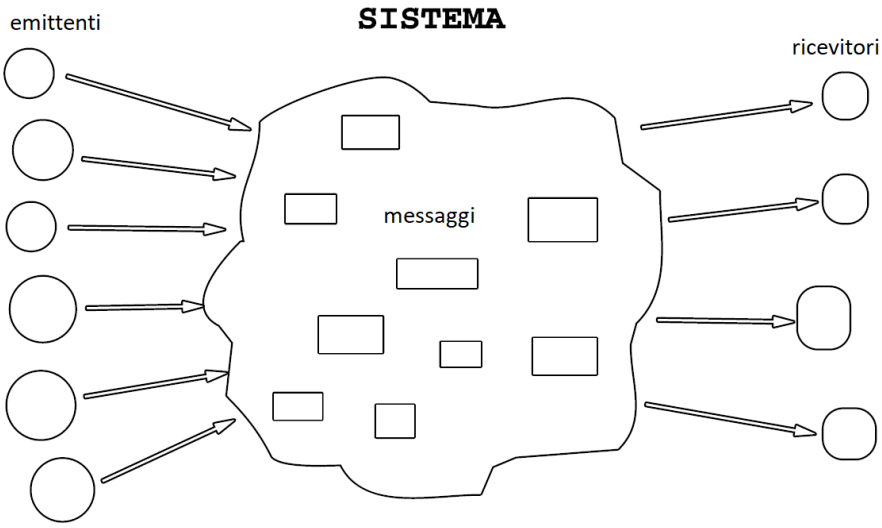

Figura 2.1: Configurazione del Sistema Generale

Successivamente si presentano casi di studio dalla prospettiva dell'attaccante che può vigilare le comunicazioni, cambiare il contenuto e studiarne i modelli. L'attaccante può essere interno o esterno al sistema.

2.1.1 Anonimato

Un soggetto è anonimo quando non può essere identificato dentro un insieme di soggetti, denominato *insieme anonimo*. Non essere identificato significa che quel soggetto non può essere caratterizzato in maniera univoca o particolare dentro quell'insieme. Dato che c'è un insieme di soggetti che potrebbero essere la potenziale causa dell'azione, un soggetto agisce anonimamente quando, dal punto di vista dell'avversario, la sua azione non può relazionarsi con la sua identità. L'anonimato deve permettere a un soggetto di utilizzare una risorsa o un servizio senza rivelare la sua identità, quindi l'anonimato non ha come scopo proteggere l'identità di un

usuario, come erroneamente si pensa, ma ciò che pretende è evitare che altri utenti o soggetti possano determinare l'identità di un utente quando fa un'operazione.

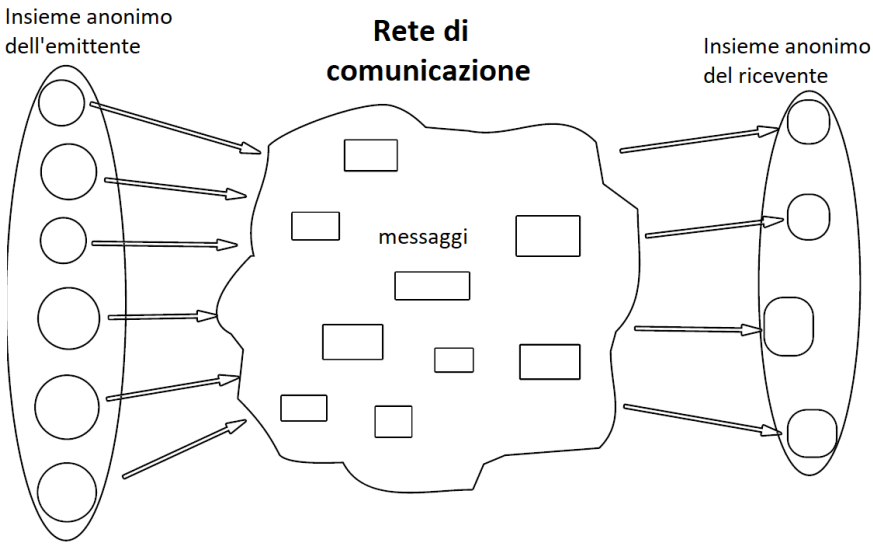

Figura 2.2: Insiemi Anonimi

L'anonimato potrebbe essere classificato in base alle entità coinvolte o in base alla loro posizione. Un'altra classificazione dell'anonimato riguarda il contesto in cui viene applicato, ad esempio *invio e ricezione di e-mail* o *interazione con un database*. In ogni classificazione ci saranno diversi attributi che caratterizzano l'insieme anonimo, e quindi l'anonimato del soggetto.
Poiché l'anonimato dipende dal contesto, definito dai suoi attributi, le variazioni nel contesto potrebbero cambiare i livelli di anonimato. Se si intende distinguere tra *livelli* di anonimato, è necessario essere in grado di quantificarlo per poter distinguere tra i diversi sistemi anonimi.

In breve, un utente può eseguire un'azione anonima solo se esiste un insieme di soggetti che potenzialmente hanno gli stessi attributi. Ad esempio, un mittente di un messaggio può essere anonimo solo se costituisce parte di una serie di potenziali mittenti (con attributi simili), che è la sua insieme anonimo, e può essere un sottoinsieme di tutti i soggetti a livello globale che possono inviare un messaggio in un momento specifico. Lo stesso vale per i destinatari dei messaggi. Questo può essere visto nella figura. Gli insiemi anonimi per mittenti e destinatari di messaggi possono essere lo stesso set, possono essere disgiunti e possono essere sovrapposti.

Il set anonimo è correlato all'attaccante, ovvero il set anonimo viene delimitato in base al livello di conoscenza dell'attaccante. Si vuole quindi, attraverso l'anonimato, che l'attaccante abbia le stesse informazioni prima e dopo un suo attacco.

2.1.2 Non relatività

La non relatività è un concetto pratico solo se le proprietà di anonimato, pseudo-anonimato e non osservabilità dei sistemi anonimi sono state precedentemente definite e le entità da correlare sono caratterizzate. La non relatività implica che non è possibile determinare che un determinato utente sia stato la causa di una determinata azione nel sistema da parte di altri utenti.

Ad esempio, l'invio e la ricezione di messaggi sono gli elementi di interesse (IDI) [3], quindi se può possibile definire l'anonimato come non relatività di un IDI con qualsiasi identificatore di un soggetto. L'anonimato di un IDI è la non relazione con qualsiasi soggetto e l'anonimato di un soggetto è la sua non relazione con qualsiasi IDI. Le proprietà che rendono un mittente non correlabile a nessun messaggio e che un messaggio non può essere correlato a un mittente sono considerate *anonimato del mittente*. Un destinatario che non può essere correlato a nessun messaggio e un messaggio che non può essere correlato a nessun destinatario è considerato *anonimato del destinatario*. In questo modo, la *relazione di anonimato* è definita come l'impossibilità di determinare chi comunica con chi, cioè, mittente e destinatario non sono correlati.

La non relatività è una condizione sufficiente per l'anonimato, ma non è una condizione necessaria. Si può dimostrare che in alcuni casi si può ottenere un alto livello di anonimato fallendo la proprietà della non relatività.

2.1.3 Non osservabilità

La non osservabilità è definita come la protezione degli elementi di interesse in se stessi, ovvero un IDI deve essere indistinguibile tra un insieme di IDI dello stesso tipo. Il tipo definisce le caratteristiche di interesse in ciascun caso.

Analogamente al caso dell'anonimato, esistono anche insiemi di soggetti non osservabili rispetto a questa proprietà: il *set di emittenti non osservabili* ha la proprietà che ogni emittente non può essere distinto dal resto, il *set di ricevitori non osservabili* ha la proprietà di non essere in grado di distinguere, dal punto di vista dell'attaccante, un ricevitore dal resto dei ricevitori di quel set. La *relazione di non*

osservabilità in questo modo significa che non è possibile distinguere tra un mittente e un destinatario che scambiano un messaggio, vale a dire, se un particolare messaggio viene considerato, allora *l'insieme della relazione di non osservabilità* è costituito da tutte le possibili coppie mittente-destinatario, tra le quali non è stato possibile distinguere o determinare l'esistenza di una relazione di trasmissione.
Ad esempio, la non osservabilità potrebbe essere definita come la proprietà che un utente può utilizzare una risorsa o un servizio senza che terze parti abbiano la possibilità di determinare che la risorsa o il servizio venga utilizzato.

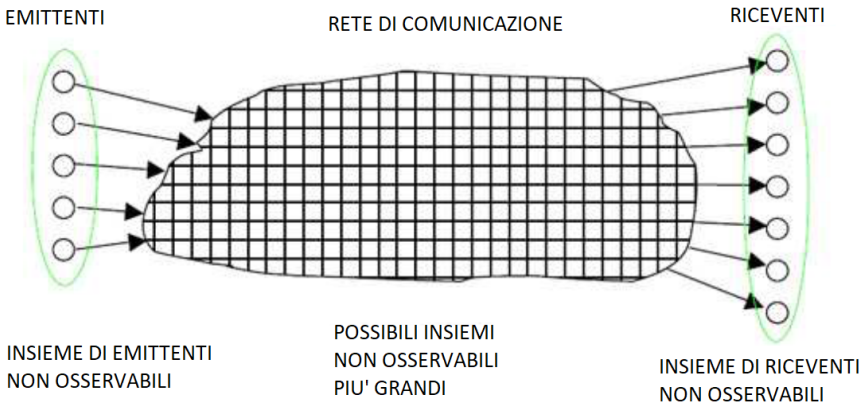

Figura 2.3: Insiemi non osservabili

2.1.4 Pseudo-anonimato

Gli pseudonimi sono identificatori, ovvero nomi o altre stringhe di bit, di soggetti [3]. Un esempio specifico sono gli identificatori di mittenti e destinatari di messaggi. In alcuni casi è conveniente raggruppare i nomi reali in un set diverso dai nomi falsi e in altri casi un nome reale può essere considerato come uno pseudonimo che risulta da una selezione iniziale per identificare un soggetto. Uno pseudonimo può essere utilizzato per una serie di argomenti o per fare riferimento a un singolo argomento. Un soggetto può avere diversi pseudonimi e non può essere trasferito ad altri utenti se non in alcuni casi. Lo pseudonimo può essere considerato come un tipo di attributo soggetto, che può essere controllato dal progettista del sistema o dall'utente stesso. Lo pseudonimo del mittente è definito come l'uso di pseudonimi dal mittente e lo pseudonimo del destinatario è definito come l'uso di pseudonimi del destinatario [3].

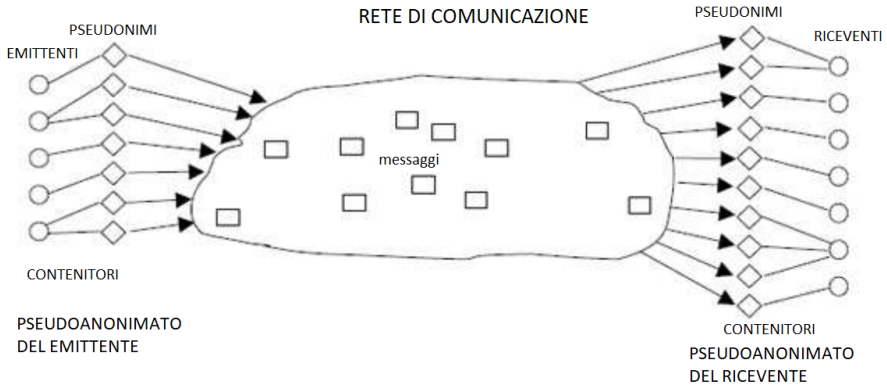

Figura 2.4: Pseudoanonimato

In breve, uno pseudonimo è un nome falso utilizzato come identificatore e lo pseudo-anonimato è il processo in cui gli pseudonimi vengono utilizzati come identificatori.

È possibile consentire un determinato livello di reputazione, utilizzando lo stesso pseudonimo. Quando vengono utilizzati sistemi anonimi, è possibile per un utente abusare e intraprendere alcune azioni non legali, per questo ci sono configurazioni di sistema anonime che consentono solo in alcuni casi di rivelare l'identità dell'utente. Solo determinate autorità certificate possono rivelare questa identità.

Poiché uno pseudonimo digitale è una stringa di bit correlata in questo lavoro a un singolo soggetto, possono essere utilizzati per autenticare i messaggi. Ciò significa che la responsabilità può essere gestita attraverso pseudonimi, ma in pratica potrebbero non essere sufficienti, per questo, sono accompagnati da firme digitali o le autorità di certificazione digitali sono utilizzate per convalidare gli pseudonimi. Esistono alcune definizioni particolari di pseudo-anonimato:

- *Pseudonimo pubblico*: la relazione tra uno pseudonimo e il suo soggetto è di dominio pubblico.
- *Pseudonimo non pubblico*: la relazione tra uno pseudonimo e il suo soggetto è nota a determinate parti ma non è di dominio pubblico.
- *Pseudonimo inizialmente non correlato*: la relazione iniziale tra uno pseudonimo e il suo soggetto è, almeno inizialmente, sconosciuta a chiunque.

Vari tipi di pseudonimi possono essere considerati in relazione alla relatività: pseudonimo personale, pseudonimo per ruolo, pseudonimo per relazione, pseudonimo per

relazione e per ruolo e pseudonimo per transazione. Lo pseudonimo personale è il più correlabile mentre lo pseudonimo per transazione è il meno correlabile.

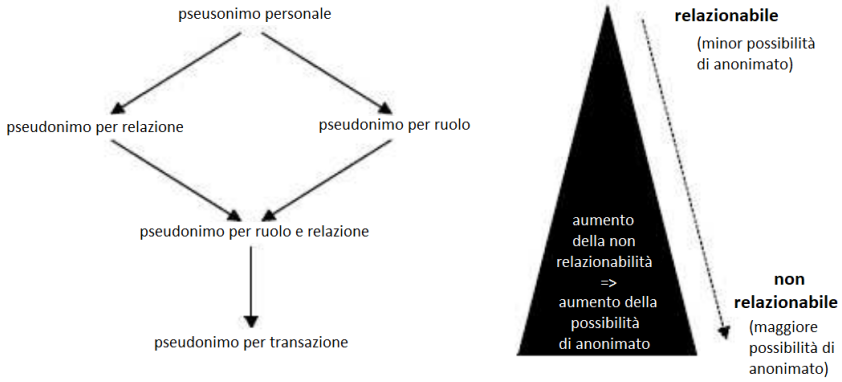

Figura 2.5: Anonimato rispetto allo Pseudoanonimato

L'anonimato è più forte quanto meno il soggetto è collegato a uno pseudonimo, ovvero meno dati personali del soggetto possono essere correlati allo pseudonimo e mentre gli pseudonimi vengono utilizzati con minore frequenza. In caso contrario diminuisce.

Un altro caso in cui è possibile utilizzare pseudonimi digitali è testare le firme digitali attraverso un meccanismo a chiave pubblica. I soggetti possono dimostrare di essere i proprietari dello pseudonimo costruendo una firma digitale creata con la loro chiave privata. Detto questo, gli pseudonimi digitali sono le stesse chiavi pubbliche generate dagli utenti stessi. Si chiama certificato di identità, lo pseudonimo creato dal vero nome di un soggetto. Se un certificato digitale contiene più informazioni (attributi) e fa riferimento a un certificato di chiave pubblica specifico, viene chiamato certificato di attributo.

2.1.5 Gestione dell'identità

È realistico supporre che un avversario possa ottenere informazioni sul mittente o sul destinatario del messaggio dal contenuto o dal contesto (luogo, ora, ...) del messaggio. Usando queste informazioni, l'attaccante può mettere in relazione i messaggi e gli pseudonimi utilizzati. Si devi anche presumere che le applicazioni create da terze parti e non solo esseri umani, utilizzino pseudonimi per inviare e ricevere messaggi. L'identità è in gran parte legata agli esseri umani, ma può essere collegata a qualsiasi soggetto, umano e non, come un computer.

Come il set anonimo, si può definire un *set identificabile* di possibili soggetti. L'identificabilità è lo stato di essere identificabile all'interno di una serie di soggetti. L'identificabilità è inversa all'anonimato, ovvero maggiore è il livello o il grado di identificabilità, minore è il grado o il livello di anonimato e viceversa. In base a quanto sopra, un'identità è un insieme di attributi appartenenti a un individuo che gli consente di distinguersi dal resto degli individui che fanno parte di un determinato insieme. Per questo motivo non esiste un'identità unica e universale, ma possono essercene diverse per lo stesso individuo, a seconda dell'insieme e del contesto a cui si fa riferimento. Anche i valori degli attributi stessi possono cambiare nel tempo.

Vengono presentati alcuni termini relativi all'identità:

- Ruolo: è un comportamento o una serie di azioni previste in un determinato contesto individuale.

- Identità parziale: l'identità di ogni persona è composta da molte identità parziali, che rappresentano la persona in un contesto o ruolo specifico. In questo modo, un'identità parziale è un sottoinsieme di attributi dell'insieme che costituisce l'identità completa, formata dall'unione di tutti gli attributi di questa persona. Gli attributi possono cambiare nel tempo, come detto prima. Uno pseudonimo può essere considerato come un'identità parziale. Un'identità parziale non può caratterizzare in modo univoco un soggetto all'interno di un insieme specifico.

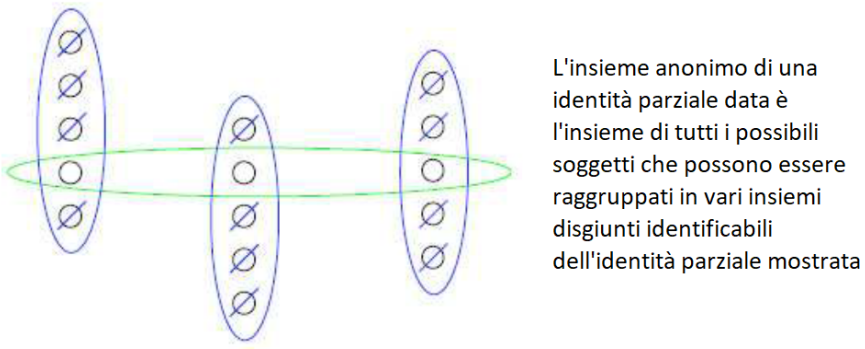

Figura 2.6: Insieme Anonimo vs. Identità parziali

- Identità digitale: indica gli attributi di una persona. Ad esempio, un identificatore di identità digitale può essere un indirizzo e-mail.

- Identità virtuale: a volte è sinonimo di identità digitale ed essendo non reale viene utilizzata in ambienti multi-utente e multi-player.

Vengono presentati alcuni termini relativi all'identità:

- Gestione delle identità: ovvero la gestione delle identità parziali di un soggetto. La quantificazione del livello di reputazione è possibile quando un utente riutilizza identità parziali.

- Gestione dell'identità per migliorare la privacy: quando non fornisce più credibilità tra entità parziali rispetto a fornire dati omettendo le informazioni degli pseudonimi.

- Gestione delle identità per migliorare la privacy nella progettazione delle applicazioni: un'applicazione è progettata per consentire alla gestione delle identità di migliorare la privacy se nessun modello di invio/ricezione di messaggi o attributi dati alle entità implica maggiore affidabilità di quanto è strettamente necessario ai fini dell'applicazione.

- Sistema di gestione delle identità: se può a distinguere tra un sistema e un'applicazione per la gestione delle identità. Il primo è un'infrastruttura, il secondo è un insieme di strumenti coordinati tra loro per gestire individualmente le loro comunicazioni, configurate sul lato server o sul lato utente.

- Sistema per la gestione dell'identità nel miglioramento della privacy: si consente all'utente di avere un più alto grado di controllo per controllare l'affidabilità dei propri dati personali. L'identità è riconosciuta per cercare di ridurre al minimo la quantità di dati utilizzati. Questo sistema aiuta gli utenti a scegliere i loro pseudonimi che rappresentano le loro identità parziali.

2.2 Tecniche di comunicazione anonima

I concetti fondamentali di anonimato sono stati introdotti nella sezione precedente. Successivamente alcuni di essi verranno ripetuti. L'anonimato di un soggetto è definito come lo stato di non essere identificabile all'interno di un insieme di soggetti, chiamato *set anonimo*. Come menzionato in [11], l'emettitore può essere anonimo solo all'interno di un insieme di potenziali emettitori, che corrisponderebbe al *insieme anonimo dell'emettitore*, che a sua volta può essere un sottoinsieme di tutti i soggetti di in tutto il mondo che potrebbe inviare messaggi in determinati momenti. Questo tipo di anonimato si chiama *anonimato del mittente*. Lo stesso vale per il destinatario, che può essere anonimo. solo all'interno di un insieme di possibili ricevitori, chiamato *set anonimo del destinatario*, e questo tipo di anonimato è chiamato *anonimato del destinatario*. Inoltre, c'è la *anonimato della relazione*, che è la proprietà di non riuscire a mettere in relazione chi comunica con chi. La *non relatività* significa che all'interno del sistema le diverse entità, qui denominate *oggetti di interesse* o IDI (messaggi, mittenti, destinatari, ecc.) non sono correlate alle informazioni che sono state conservate in precedenza dall'avversario per eseguire un attacco (informazioni a priori). In altre parole, l'anonimato del mittente/destinatario può essere definito come le proprietà che un determinato messaggio non è correlato a nessun mittente/destinatario e che qualsiasi messaggio non è correlato a un particolare mittente/destinatario. L'anonimato si rafforza tanto più grande è il suo insieme anonimo, e più uniforme è la distribuzione dell'esecuzione delle azioni da parte dei soggetti all'interno del set, ovvero il livello di anonimato non dipende solo dalle dimensioni del set, ma anche della probabilità che un determinato soggetto possa generare una determinata azione.

I soggetti non possono avere lo stesso livello di anonimato contro tutti i tipi di possibili attacchi generati da partecipanti interni o esterni. Si presume che, dal punto di vista dell'attaccante, il livello di anonimato possa solo diminuire, ovvero si presume che l'attaccante non dimentichi le informazioni che ha e che è riuscito a raccogliere durante la sua osservazione e influenza sulla comunicazione nel sistema . Esistono diverse tecniche di anonimato che possono essere definite attraverso criteri che:

- *Scopo della protezione*: indica il tipo di anonimato che viene fornito (mittente, destinatario o relazione).

- *Livello di sicurezza*: si deve indicare il livello di sicurezza che si vuole raggiungere.
- *Modello di attaccante*: gli attaccanti contro i quali il sistema deve essere protetto (esterno, partecipanti, fornitori di servizi).
- *Modello di fiducia*: i soggetti di cui l'utente può fidarsi, ad esempio fornitori di servizi, partecipanti esterni, altri utenti.

2.2.1 Proxy semplici

Come ogni proxy, le richieste che devono essere fatte a un servizio Web non vengono fatte direttamente al server, ma piuttosto è un server proxy che effettua la richiesta da un punto intermedio. Ossia, le richieste non vengono inviate direttamente dall'utente al server Web, ma il client si connette a un altro server, chiamato proxy, e questo genera la richiesta Web al server. Il server Web vede solo l'indirizzo IP del server proxy e non quello dell'utente e il server proxy può filtrare le informazioni di richiesta che potrebbero essere utilizzate per identificare l'utente.

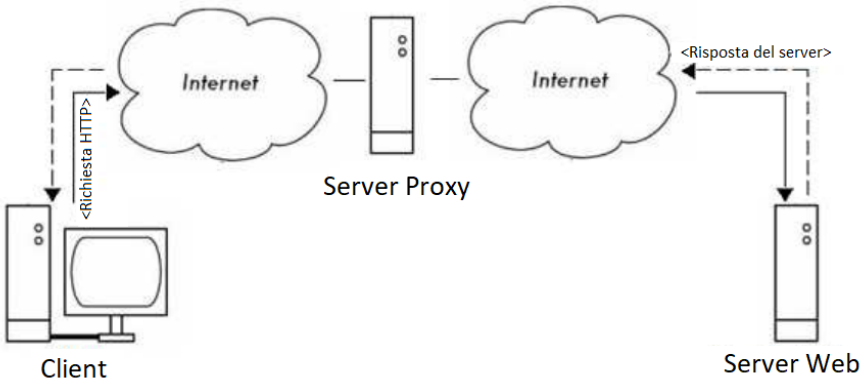

Figura 2.7: Proxy

Le due possibilità di connessione a un proxy sono tramite:

1. *Sito Web*: l'uso di servizi anonimi è consentito tramite un sito Web, pertanto l'utente non deve installare programmi. Consiste nel fornire l'indirizzo del sito Web in cui l'utente desidera connettersi al proxy e quindi quando arriva la risposta, il proxy lo consegna all'utente. Un dettaglio importante è che se il proxy necessita di collegamenti aggiuntivi, li cerchi e li trasformi in modo che vengano utilizzati anche come comunicazione iniziale. In questo modo il

client è autorizzato a rimanere anonimo nel caso in cui vengano monitorati i collegamenti aggiuntivi.

2. *Proxy locali*: contrariamente al precedente, l'utente deve installare un programma, registrarsi e impostare il browser per connettersi ai siti web. Per nascondere la propria identità, l'utente utilizza il programma che contiene un elenco di proxy aperti, ovvero proxy lasciati aperti per uso pubblico, ne sceglie uno a caso e lo utilizza allo stesso modo dei siti Web proxy. Il programma proxy locale è progettato per cercare e scoprire automaticamente nuovi proxy aperti.

Non ci sono restrizioni sull'uso di entrambi in combinazione, inoltre è conveniente.

Si possono usare più proxy, ciascuno con i suoi vantaggi e svantaggi. Possono essere utilizzati per formare una catena al fine di ottenere un livello migliore di filtro per le richieste Web (HTTP). Un aspetto molto importante ora è la fiducia in ciascuno di essi.
Vi sono due obiettivi di protezione: l'anonimato del mittente rispetto al destinatario e l'anonimato della relazione rispetto a tutto il resto. Non esiste protezione contro la collisione di più provider proxy danneggiati e non esiste protezione contro gli attaccanti che possono abbinare i messaggi in entrata e in uscita attraverso l'attacco di tipo analisi temporale.

2.2.2 Crowd

Questo sistema ha lo scopo di fornire elevate prestazioni e anonimato dell'emittente ed è costituito da una raccolta dinamica di utenti chiamata folla o crowd per nascondere le attività di un utente. Prima di raggiungere il server Web, una richiesta passa attraverso un numero casuale di partecipanti della folla e ogni membro quando riceve il messaggio sceglie se inviarlo direttamente al server di destinazione o ad un altro membro del sistema [12]. In questo modo né il server né nessun altro membro del sistema sa chi è l'iniziatore della richiesta e si chiama *plausible deniability*.

Se l'utente vuole far parte della rete, deve installare un programma aggiuntivo, chiamato jondo, registrarsi con un server centrale, chiamato blender, e creare uno pseudonimo e una password per l'autenticazione personale. Il jondo è una specie di proxy locale, quindi il browser deve essere configurato per usarlo. All'avvio del sistema, un jondo contatta prima il blender per richiedere l'accesso alla rete Crowds. Per accedere, il server invia un elenco dei membri della rete (jondo) e le loro chiavi di

crittografia simmetriche. Inoltre, il server informa il resto dei membri della rete sul nuovo membro. Queste chiavi sono necessarie perché tutte le informazioni (richieste e risposte) sono crittografate da un membro a un altro nella rete. Qualsiasi richiesta Web da un browser viene inviata a un jondo locale, che a sua volta invia la richiesta in modo casuale a un membro della rete.

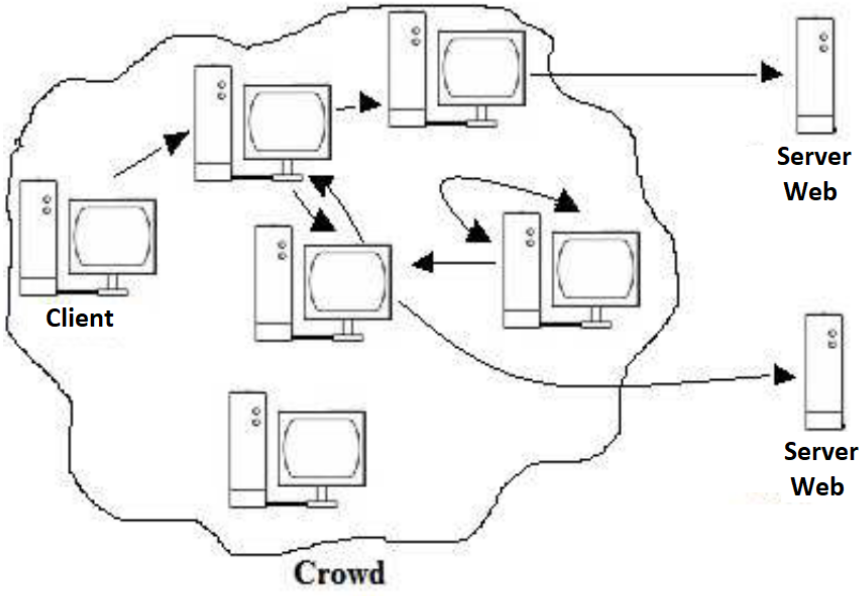

Figura 2.8: Crowd

Quando un messaggio viene inviato o ricevuto da un membro della rete, un identificativo di traccia (ID) viene archiviato in un database locale. Questo ID consente di inviare risposte nella direzione opposta dal server al client richiedente. Ogni jondo memorizza le coppie ID del percorso corrispondenti all'ingresso e all'uscita di una richiesta. Se un jondo riceve un messaggio da un altro jondo, verifica se l'ID della traccia ricevuto è già memorizzato nella tabella, in tal caso, il jondo inoltra il messaggio al jondo successivo, a seconda del secondo ID della traccia nella tabella. Se l'ID della traccia non è presente nella tabella, viene selezionata una nuova destinazione (jondo o il server finale), il messaggio viene inviato e una nuova coppia viene memorizzata nella tabella.

Come la catena di proxy, questo sistema protegge l'anonimato del mittente dai destinatari e l'anonimato della relazione contro tutto il resto. Questo sistema non fornisce protezione contro gli attaccanti esterni che stanno monitorando l'intera rete. Inoltre, la crittografia asimmetrica non viene utilizzata e si può raggiungere un livello

incondizionato di sicurezza. A proposito di fiducia, l'utente deve riporla nel blender e nella disponibilità della rete.

2.2.3 Diffusione o Broadcast

Questa tecnica sfrutta il meccanismo utilizzato per ricevere segnali radio o televisivi, ovvero le informazioni vengono inviate a tutti i partecipanti della rete e ciascuno sceglie localmente se è importante per lui. In questo modo l'attaccante passivo non può determinare se l'informazione sta andando a un determinato ricevitore. Se il messaggio viene indirizzato a un partecipante specifico, viene utilizzato un indirizzo implicito, ovvero non esiste alcuna relazione tra questo indirizzo e il destinatario fisico, ma solo per il destinatario che può interpretarlo. Il mittente non può ottenere informazioni specifiche su questo destinatario e quando invia un messaggio inserisce questo indirizzo implicito. Ogni destinatario che riceve il messaggio verifica che questo sia l'indirizzo implicito. Se l'indirizzo può essere visto pubblicamente, si chiama *indirizzo implicito visibile*. Per evitare il problema che possono essere correlati messaggi diversi per lo stesso destinatario, in ciascun messaggio vengono utilizzati indirizzi visibili diversi. Inoltre, il messaggio dovrebbe impedire ad altri destinatari della trasmissione di leggerne il contenuto. Al contrario, se l'indirizzo implicito è crittografato, viene chiamato indirizzo invisibile implicito, ma questo costringe ogni stazione a decifrare tutti i messaggi che riceve per verificare se qualcuno di essi corrisponde ad esso.

In una rete commutata, in cui ogni stazione riceve solo ciò che è stato inviato da un altro partecipante o riceve solo ciò che ha richiesto, è possibile utilizzare un sistema di comunicazione multiplo o multicast. Questo tipo di diffusione parziale significa che ciascuno dei partecipanti non riceve tutti i messaggi ma un sottoinsieme di essi, riducendo così la larghezza di banda necessaria, ma diminuisce anche il livello di anonimato.

Per quanto riguarda i criteri, si analizzano ciascuno di essi. Gli obiettivi di protezione della tecnica di trasmissione sono l'anonimato del destinatario che utilizza gli indirizzi impliciti, la non osservabilità del ricevitore contro avversari esterni se viene utilizzato un tipo di traffico di riempimento chiamato traffico dummy. La non relazionalità di messaggi diversi per lo stesso destinatario si ottiene utilizzando indirizzi impliciti diversi per ciascun messaggio. Per quanto riguarda il modello dell'attaccante, per quanto riguarda l'anonimato e la non correlazione, esiste una protezione contro gli attaccanti passivi esterni e interni (osservatori). Per quanto

riguarda la non osservabilità, esiste una protezione contro gli attaccanti esterni. Per quanto riguarda il modello di fiducia, se vengono utilizzati traffico dummy e indirizzi impliciti, non è necessario fidarsi di un altro partecipante o di qualsiasi fornitore di servizi.

2.2.4 Rete ad anello

Come dice il nome, in questa tecnica le stazioni sono cablate in maniera circolare, quindi è valido solo per configurazioni di reti locali o regionali.

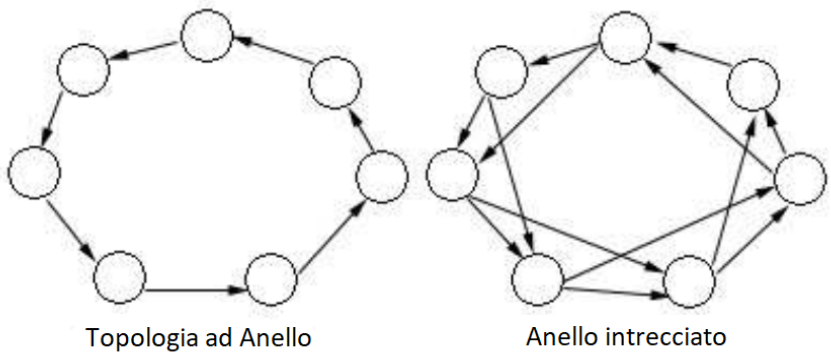

Figura 2.9: Topologia ad Anello

Attraverso l'invio ripetuto ad ogni stazione partecipante, queste sembrano essere la stazione iniziale, quindi l'anonimato viene fornito dal mittente rispetto agli avversari che osservano o controllano le stazioni o collegano le reti, purché non siano direttamente prima o dopo il mittente. Ogni messaggio è indipendente dal messaggio originale attraverso l'uso della rigenerazione del segnale digitale in ciascuna stazione partecipante. Inoltre, il destinatario è anonimo e non osservabile a causa dell'inoltro del messaggio sull'intero anello. Un prerequisito aggiuntivo necessario per garantire l'anonimato dell'emittente è che i permessi di invio siano correttamente configurati e garantiti.

Se due stazioni senza collaborazione reciproca vogliono osservare una stazione situata tra loro, non saranno in grado di osservare qualcosa di significativo perché i messaggi in uscita sono crittografati. Se un utente malintenzionato non può circondare una stazione e controllare i messaggi in entrata e in uscita, potrebbe dedurre solo se una stazione, all'interno di un piccolo gruppo di stazioni direttamente collegate, invia un messaggio, senza sapere quale. Con questa topologia, se si utilizza la

rigenerazione del segnale digitale e una tecnica per più accessi anonimi, si fornisce l'anonimato del mittente e del destinatario rispetto a un utente malintenzionato che controlla alcune stazioni. Al fine di garantire che i messaggi vengano ricevuti dalle stazioni appropriate, è sufficiente che al mittente ritorni invariato il messaggio dopo esser tornato dall'anello.

Nella topologia ad anello le connessioni sono serializzate, quindi affinché la comunicazione abbia successo le stazioni devono funzionare correttamente e i nodi difettosi dovranno essere rimossi dalla rete. Una soluzione è l'anello attorcigliato, ovvero due reti di anelli interconnesse tra loro, ad esempio, la prima collega i nodi adiacenti, la seconda collega coppie di stazioni separate da un'altra intermedia. In questo modo la capacità di trasmissione viene raddoppiata e viene compensato un malfunzionamento di un nodo.

L'obiettivo di protezione è l'anonimato del mittente e l'anonimato del destinatario attraverso l'invio del messaggio sull'anello completo. Il livello di sicurezza che può essere raggiunto sarà solo a livello computazionale se la crittografia utilizzata per i messaggi in uscita si basa su ipotesi crittografiche (crittografia asimmetrica). In caso contrario, è possibile raggiungere un livello incondizionato di sicurezza. Per quanto riguarda il modello avversario, questa tecnica offre protezione contro un attaccante che controlla alcune stazioni ad eccezione di quelle che si trovano prima e dopo l'emettitore. Il modello di fiducia per questo caso si riferisce al fatto che le stazioni vicine di un utente non devono collaborare contro di lui.

2.2.5 Bus

Come proposero in [13], questo sistema si basa sull'invio di messaggi in attesa del proprio turno, quindi è molto simile al meccanismo token di una rete token-ring. Si ritiene che per un osservatore sia molto difficile tracciare la traiettoria di un messaggio che si sposta nella rete ed è ancora più difficile se utilizza sezioni diverse per completare il suo percorso. Sono stati implementati tre tipi di sistemi bus:

- Il primo si basa su una topologia ad anello in cui i messaggi viaggiano sempre nella stessa direzione. È possibile inviare un singolo messaggio per ogni coppia mittente/destinatario. Viene utilizzata la crittografia asimmetrica e per garantire che un utente malintenzionato non possa vedere se una stazione desidera o meno inviare un messaggio, tutti inviano messaggi a tutti. In questo modo un avversario non può sapere se esiste una comunicazione reale tra le stazioni.

Se si considera solo il numero di messaggi sulla rete, le prestazioni sono ottimali ma poiché la lunghezza di ogni invio ha una crescita quadratica rispetto al numero di stazioni, i messaggi impiegano molto tempo per raggiungere la destinazione. Inoltre, un messaggio, nel peggiore dei casi, deve passare attraverso tutte le stazioni.
In alternativa, è possibile inviare più messaggi per coppia mittente/destinatario, crittografando i messaggi come cipolla o in livelli. Questo viene fatto crittografando il messaggio con la chiave pubblica del destinatario e quindi con la chiave pubblica delle altre stazioni. Quando il messaggio viene inviato, ogni stazione decodifica il layer corrispondente e lo inoltra alla stazione successiva se si rende conto che il messaggio non corrisponde ad esso. Se decodificando il messaggio può essere letto, significa che questa stazione è il destinatario del messaggio.

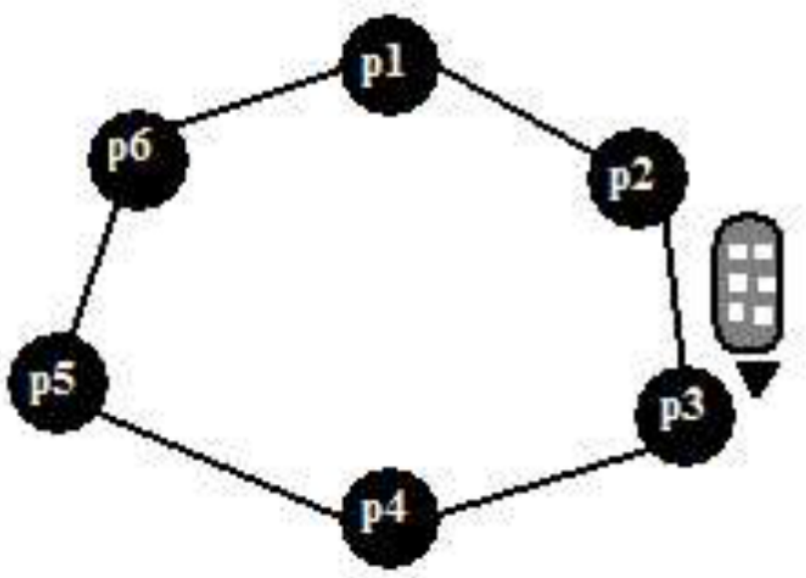

Figura 2.10: Rete ad Anello con un solo bus

- Il secondo tipo di sistema, introdotto sempre in [13], consiste nell'inviare messaggi in direzioni opposte, sempre in un percorso circolare. Ciò migliora i tempi di invio, ma ci sono degli svantaggi nel processo di comunicazione.
- Il terzo tipo di sistema introduce il concetto di cluster che è un gruppo di stazioni. I bus utilizzano una tecnica di instradamento circolare (anello), a un

livello di strato di comunicazione elevato (modello OSI), considerando qualsiasi topologia di rete (a livello di rete o livello di collegamento). Realizzando implementazioni pratiche di questo sistema, hanno scoperto che è utilizzabile solo per implementare reti di piccole dimensioni, inoltre ha bisogno di molto traffico di riempimento per nascondere le interazioni tra gli utenti.

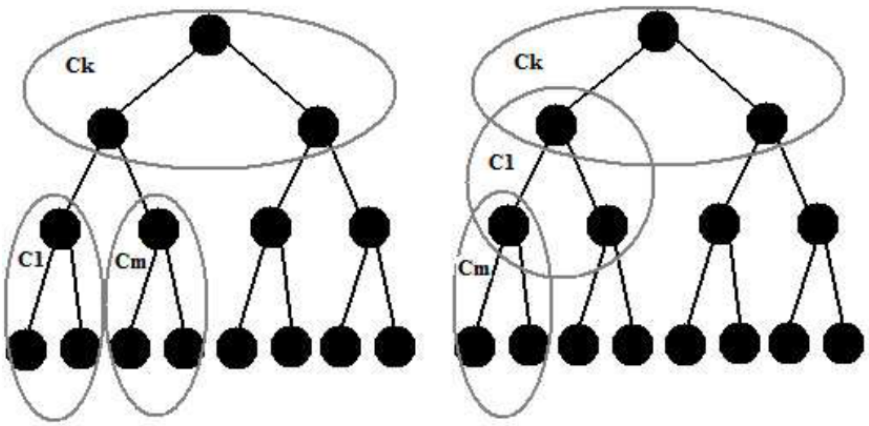

Figura 2.11: Rete ad Anello divisa in "cluster"

Gli obiettivi di protezione sono proteggere l'anonimato del mittente, del destinatario e della relazione. È possibile raggiungere solo un livello di sicurezza computazionale, perché la crittografia asimmetrica viene utilizzata per l'invio di messaggi, che si basa su ipotesi crittografiche. Per quanto riguarda il modello dell'attaccante, possono essere descritti due tipi: un attaccante che legge effettivamente i messaggi sulla rete e controlla alcune delle stazioni; gli attaccanti che possono manipolare, creare o eliminare messaggi. Questo sistema non può impedire un attacco Denial of Service (DoS).

2.2.6 Rete-DC

David Chaum ha proposto questa rete [14] allo scopo di fornire l'anonimato all'emittente su molte topologie di rete. DC è l'acronimo di Dining Cryptographers ed è compreso da un esempio relativo a un pasto. Il principio alla base di questa rete si chiama *supporto di invio* e consiste nell'inviare, da tutte le stazioni, un messaggio reale o fittizio in un determinato momento e la sovrapposizione di questi messaggi verrà ricevuta da tutte le stazioni.

Ogni stazione genera chiavi casuali che trasmette alle altre stazioni della rete attraverso un canale sicuro. Ogni stazione avrà quindi n-1 chiavi.

Se una stazione desidera inviare un messaggio, effettua ciò che viene chiamato *local overlap*, ovvero aggiunge il messaggio, le chiavi generate e l'inverso di tutte le chiavi ricevute. Se una stazione non desidera inviare un messaggio, deve inviare un messaggio vuoto, sovrapposto a tutte le chiavi note. Tutte le uscite si sovrappongono a livello globale e sono distribuite a ciascuna delle stazioni sulla rete. Poiché ogni chiave e il suo inverso sono stati aggiunti esattamente una volta, le chiavi si eliminano a vicenda nella sovrapposizione globale. Inoltre, il risultato della sovrapposizione globale è la somma di tutti i messaggi inviati. Se nessuna stazione desidera inviare un messaggio, la somma corrisponderà a un messaggio vuoto. Se solo un membro invia un messaggio, la somma sarebbe uguale a questo messaggio. Se due o più stazioni inviano contemporaneamente i loro risultati sovrapposti, potrebbero causare collisioni, il che rappresenta un problema nei sistemi di distribuzione nei canali con più accessi. Alla fine ogni partecipante al sistema ha il messaggio originale.

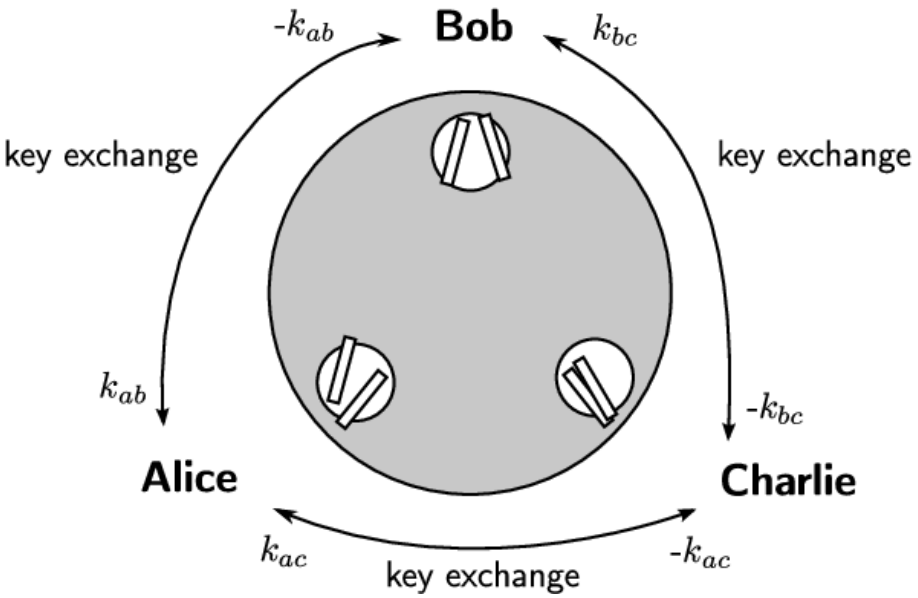

Figura 2.12: Crittografia di Chaum

Ogni chiave viene usato solo una volta, cioè le chiavi devono essere cambiate in ciascuno dei round. In caso contrario, l'output di una stazione che invia un messaggio vuoto sarebbe identico al precedente, che può essere utilizzato dall'aggressore per determinare questo fatto. Si preferisce utilizzare un meccanismo di crittografia per nascondere il contenuto del messaggio. L'uso di indirizzi impliciti potrebbe

preservare l'anonimato del destinatario. Le reti DC sono suscettibili agli attacchi DoS, perché se una stazione non funziona correttamente o smette di funzionare, verrebbero trasmessi solo messaggi privi di significato. Inoltre, è una tecnica molto costosa per il carico della rete perché quando si aggiungono partecipanti alla rete, il traffico aumenterà in modo lineare.

Gli obiettivi di protezione raggiunti sono l'anonimato del mittente, l'anonimato del destinatario quando si utilizzano la trasmissione e l'indirizzamento implicito, l'anonimato della relazione e la non osservabilità del mittente e del destinatario attraverso l'uso del traffico di riempimento o dummy. Questo modello fornisce protezione dagli aggressori che sono partecipanti interni, ma è vulnerabile agli attacchi Denial of Service, tuttavia, questi tipi di aggressori possono essere scoperti ed esclusi. Per quanto riguarda il modello di fiducia, tutti i partecipanti dovrebbero impegnarsi e tollerare il rispetto delle regole.

2.2.7 Mixer

Viene utilizzata la crittografia a chiave pubblica ed è stata progettata in modo tale che i sistemi di invio di posta elettronica garantiscano l'anonimato del mittente, del destinatario e della relazione senza la necessità di un servizio fiduciario centrale [15]. I mix possono essere intesi come una catena di proxy seguiti uno dopo l'altro. L'aggressore è considerato in grado di osservare tutte le comunicazioni e può controllare tutti i mix tranne uno. L'utente deve fidarsi di questo mix.

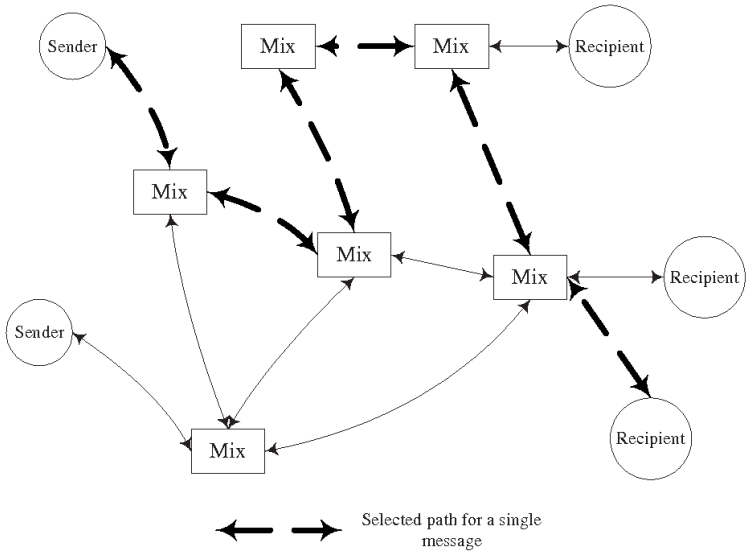

Figura 2.13: Rete di Mixer

Generalmente nella rete c'è più di un mix organizzato sotto forma di una catena che può cooperare in diversi metodi. Uno di questi è che i mix sono indipendenti e i partecipanti scelgono attraverso quali mix indirizzare i loro messaggi. Pertanto ogni nodo può comunicare con il resto creando una topologia chiamata *rete mix*. Un'altra possibilità è quella di utilizzare una stringa di mix predefinita chiamata *mix a cascata*. Queste due opzioni possono essere mescolate per creare soluzioni ibride, discusse in [4, 16].
Se l'utente sceglie i mix con cui desidera interagire, si fornisce un buon livello di scalabilità e flessibilità. Se, inoltre, i mix vengono scelti in modo casuale, un utente malintenzionato non può determinare quale di essi dovrebbe controllare per osservare un messaggio inviato, per questo dovrebbe controllare gran parte della rete. Gli attacchi che controllano tutti i mix tranne uno sono chiamati n-1. Un altro svantaggio è che alcuni mix possono essere inutilizzati, sottoutilizzati o sovraccaricati.
D'altra parte, un utente malintenzionato sa esattamente quali mix controllare in una rete in cascata (mix in cascata). Questo design è vulnerabile agli attacchi Denial of Service, poiché arrestando un singolo mix nella rete, sarebbe in grado di arrestare l'intero sistema.

Come funziona un mix può essere descritto attraverso una sequenza di operazioni. Gli utenti inviano messaggi ai mix, che per nascondere le comunicazioni dei clienti non inviano istantaneamente i messaggi che ricevono, ma li memorizzano insieme ad altri. Dopo un tempo definito, trasformano i messaggi e li inviano ai server di destinazione o ad altri mix. Un osservatore che può vedere tutti i messaggi in entrata e in uscita dello stesso mix non può determinare quali messaggi in arrivo corrispondono a quali messaggi in uscita.

Gli obiettivi di protezione raggiunti sono l'anonimato dell'emittente e la relazione. Fornisce protezione contro gli aggressori che possono osservare l'intera rete e che possono controllare molti mix. È suscettibile agli attacchi Denial of Service e agli attacchi n-1. Dal punto di vista della fiducia, bisogna fidarsi di almeno un mix del percorso selezionato.

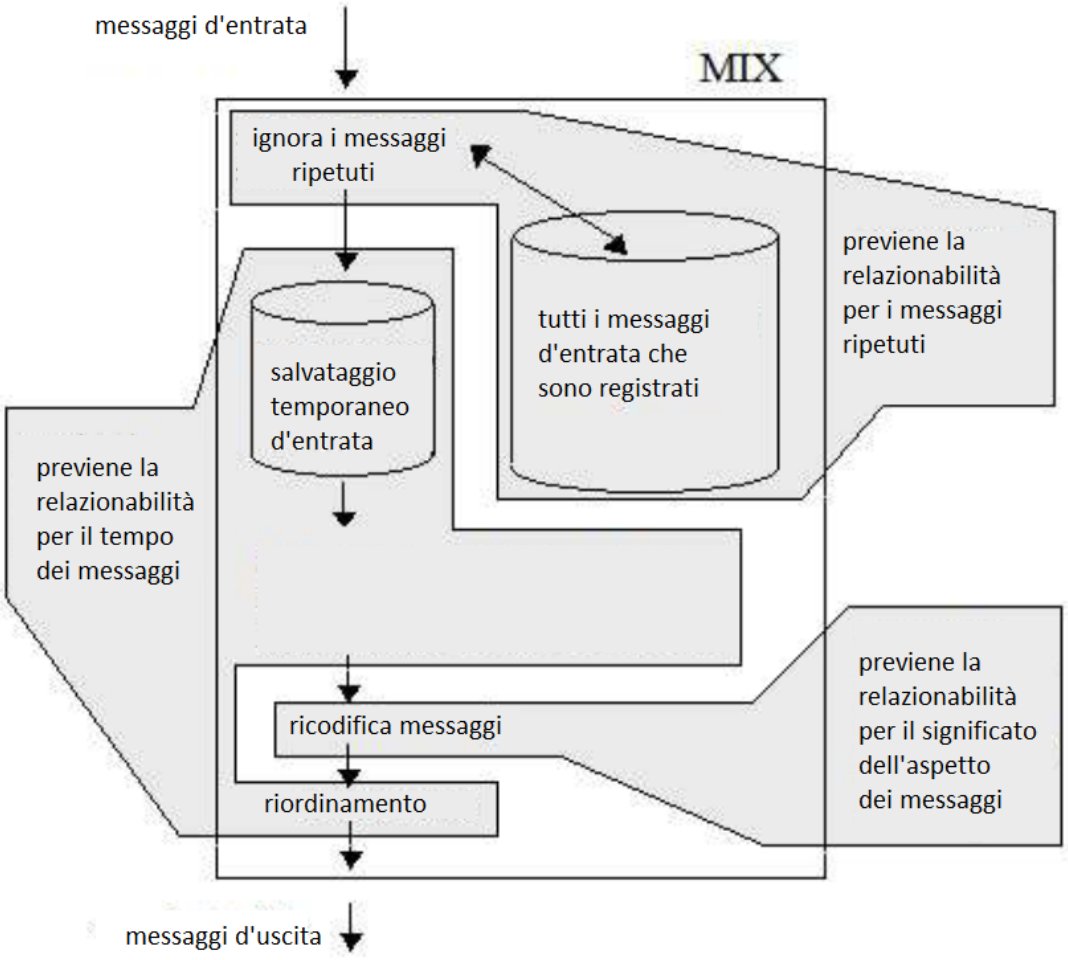

Figura 2.14: Fondamenti che sostengono la rete di mixer

2.3 Sistemi di comunicazione anonima

Con il passar del tempo sono stati sviluppati vari sistemi che migliorano la privacy nella rete e in questa sezione si descriveranno i più conosciuti con maggior interesse e con meno dettagli i restanti.

2.3.1 Sistema di rinvio Anon.penet.fi

Nel 1993 Helsingius iniziò a implementare un modello affidabile per l'inoltro della posta e che forniva account di posta elettronica anonimi e pseudonimi [9]. Dietro questo modello c'era un server che gestiva una tabella con gli indirizzi e-mail reali e gli indirizzi pseudonimi corrispondenti. Quando è stata ricevuta una e-mail con uno pseudonimo, questo è stato inoltrato all'utente reale attraverso la tabella. In questo modo tutte le informazioni dell'utente reale non erano interessate nell'invio di messaggi e non c'era possibilità di trovarle. Solo alla fine si inviava al vero destinatario. Questo modello era molto debole per gli attaccanti locali passivi o dallo stesso server che poteva propagare le vere identità degli utenti sfruttando le correlazioni temporali tra traffico in entrata e in uscita. Per motivi legali, questo sistema è stato chiuso nel 1996.

2.3.2 Anonymizer e SafeWeb

La società Anonymizer (http://www.anonymizer.com) fondata da Cottrell che forniva agli utenti abbonati a tale sistema un servizio di navigazione web anonimo. Nel sistema è presente un proxy Web che inoltra richieste e risposte Web, anche questo proxy è chiamato Anonymizer. Le informazioni degli utenti non possono essere ottenute dai server Web a cui si accede perché esisteva un filtro attivo dei contenuti, fornito dal sistema, che poteva eseguire codici sui computer locali degli utenti e dopo inviare le informazioni che identificano l'utente in rete.
L'integrità dell'azienda e dei suoi membri sono i fattori più importanti per l'anonimato. Questo sistema è più efficace contro gli attacchi di natura legale poiché un record non viene conservato nel tempo, ovvero quando termina una comunicazione tra un utente e un server Web, tutti i record vengono eliminati.
Un'altra società, chiamata Safeweb, ha fornito un servizio molto simile al precedente con la differenza che le connessioni erano crittografate con SSL, quindi il contenuto delle pagine accessibili era sicuro. Anche qui c'era un filtro per trovare contenuti attivi. Si scoprì che il sistema non poteva resistere ad alcuni tipi di attacco e venne

rimosso dal mercato.
Un osservatore, vale a dire un attaccante passivo, potrebbe mettere in relazione agli utenti quelli che hanno richiesto in assenza di traffico di riempimento o sistemi di miscelazione. La persona che guarda il traffico può compilare una libreria con le firme dei dati dell'utente e delle pagine Web che ha richiesto. Queste firme possono quindi essere confrontate con le caratteristiche del traffico sulla connessione SSL per dedurre le pagine. Questa procedura si chiama Analisi del traffico [17, 18, 19].

2.3.3 Rinvii di email Tipo I chyperpunk

Quando si parlò di una mailing list chiamata "cypherpunk", fu inventata la prima implementazione di rinvii di email di tipo I [9]. Questi nodi, prima di inviare le e-mail, rimuovevano tutte le informazioni sull'utente e poi le crittografavano con la loro chiave privata. Le funzioni di crittografia utilizzate erano la chiave pubblica del PGP (Pretty Good Privacy). Per l'esecuzione manuale, sono stati proposti schemi di codifica che utilizzavano strumenti standard per la gestione di testo ed e-mail. Per evitare la dipendenza e le debolezze dell'uso di un singolo rinvio, una soluzione adottata era quella di concatenare diversi rinvii.
Per l'invio degli indirizzi anonimi furono posti blocchi di inoltro. L'indirizzo e-mail dell'utente era crittografato con la chiave pubblica e quindi aggiunto come intestazione speciale. Quando un utente voleva inviare una e-mail anonima, il rinvio la decodificava e inoltrava il contenuto. Poiché le informazioni sull'indirizzo per l'inoltro dei messaggi erano incluse nei messaggi in forma crittografata, non era necessario tenere un registro in cui gli utenti e i loro indirizzi reali erano correlati.
Utilizzando uno schema di crittografia si evita la maggior parte degli attacchi che vengono effettuati attraverso l'osservazione passiva dei bit dei messaggi in arrivo e il loro confronto con quelli in uscita, lasciando informazioni che possono essere utili per un altro tipo di attacco. Ad esempio, è possibile seguire un messaggio che viaggia attraverso la rete osservando la sua dimensione perché la dimensione non viene presa in considerazione nei meccanismi PGP, quindi non è nascosta.

2.3.4 Crowds

Nei laboratori AT&T è stato sviluppato Crowds, che è l'implementazione della tecnica che porta lo stesso nome [20]. L'intento di questo sistema è nascondere l'identità impedendo ai server di determinare chi richiede i loro servizi. Per fare ciò, un utente invia una richiesta a un server centrale e risponde con un elenco dei nodi che com-

pongono il sistema. L'utente invia quindi la sua richiesta Web a un nodo nell'elenco selezionato in modo casuale. Quando un nodo intermedio riceve un messaggio, decide casualmente se inviarlo alla sua destinazione finale o ad un altro nodo intermedio. Alla fine, la risposta del server prende la rotta stabilita nel processo di invio.
Il punto di forza della sicurezza di questo sistema è l'impossibilità dell'attaccante di osservare i collegamenti. L'unica azione che un utente malintenzionato può eseguire è controllare un piccolo numero di nodi in ciascun gruppo e nel server finale. Nell'ambito della ricerca sull'anonimato questo sistema è ben noto. Qualsiasi nodo che riceve una richiesta non può sapere se il nodo precedente era un iniziatore di comunicazione o un altro nodo intermedio, questo concetto è chiamato *iniziatore anonimato*. Usando attacchi progettati da ricercatori, è stato scoperto che se un client richiede molte volte una particolare risorsa, queste richieste possono essere correlate perché hanno capito che un vero iniziatore che genera richieste ripetute sarà il predecessore di un nodo corrotto più spesso di qualsiasi nodo nel sistema. Pertanto questa proprietà per questo sistema è stata contrastata.

2.3.5 Server Nym

Sistema che attraverso i server Nym [21] memorizza blocchi di inoltro anonimi e li mappa con indirizzi e-mail pseudonimi. Un messaggio ricevuto da un indirizzo e-mail viene inoltrato in modo anonimo utilizzando il blocco del proprietario dello pseudonimo. In breve, il server Nym ha la funzione di un "porta di collegamento" o " gateway " tra il mondo convenzionale delle e-mail e il suo mondo anonimo. Un vantaggio di questi server è che l'utente non deve fidarsi di loro per salvaguardare la propria identità poiché non conserva informazioni sull'identificazione.
L'implementazione di sistemi di reputazione o altre misure che aiutano a prevenire l'abuso di questi sistemi è resa possibile dal fatto che identità pseudonime hanno una certa persistenza nel tempo.

2.3.6 Il mix di Chaum

Utilizzando la tecnica del mixer, in cui un nodo nasconde la corrispondenza tra i messaggi di input e i messaggi di output attraverso la crittografia, alla fine degli anni '70 "El mix de Chaum" [15] è nato quasi contemporaneamente alla crittografia RSA che fu utilizzata per la crittografia e la decrittografia.
I messaggi che dovevano essere anonimi venivano inoltrati attraverso un nodo chiamato "mix" che aveva una chiave pubblica RSA. I messaggi vengono suddivisi in

blocchi e crittografati con la chiave, ma il primo blocco contiene l'indirizzo del mix successivo. Quando un mix riceve il messaggio, lo decodifica ed estrae il primo blocco contenente l'indirizzo del destinatario o il mix successivo e aggiunge un blocco di bit casuali alla fine chiamato "junk". La lunghezza della posta indesiderata viene creata in modo che la dimensione del messaggio non cambi.
[22] Questo sistema presenta molti punti deboli come non fornire le proprietà di non relatività, la struttura matematica di RSA non utile nei sistemi di firma e che consente ad alcuni tipi di attacchi attivi di ottenere informazioni per trovare corrispondenza tra testo crittografato e testo normale. Inoltre, è possibile eseguire attacchi di tagging in cui i blocchi possono essere duplicati o sostituiti perché i blocchi crittografati mediante RSA non dipendono l'uno dall'altro.
Un'altra funzione di un mix è quella di mescolare i messaggi in modo che sia difficile per l'avversario mettere in relazione i messaggi di input e output. Un mix di tipo batch memorizza un numero specifico di messaggi, li ricodifica in ciascun periodo, li riordina lessicograficamente e li inoltra. Tecniche fittizie sul traffico possono essere utilizzate in questi tipi di sistemi.

2.3.7 Mixes ISDN, in tempo reale e web

Esiste un sistema che consente conversazioni telefoniche anonime utilizzando le tecnologie ISDN [23]. Successivamente è stato esteso per supportare il lavoro per le comunicazioni in tempo reale, con mix e per la navigazione web anonima. Questa versione era chiamata Web Mixes [24]. Parte di questo progetto è stato implementato come proxy web per l'anonimato ed è stato chiamato JP. La comunità anonima di Dresda è l'insieme dei tre disegni precedenti.
Lo scopo di utilizzare queste tre proposte è quello di ottenere comunicazioni anonime sicure nonostante la presenza di un avversario che possa osservare tutte le comunicazioni, generare ritardi, creare ed eliminare messaggi e controllare un certo numero di mix. Come topologia di base, tutti e tre i progetti utilizzano una rete a cascata per garantire che tutti i messaggi vengano elaborati da tutti i mix nello stesso ordine, per rimuovere la necessità di pesare o inviare informazioni sul percorso di messaggi e per proteggere il sistema da un insieme di attacchi [25], come anche gli attacchi con tag. Invece della topologia a cascata, grazie agli studi è stato possibile utilizzare nuove topologie [26, 27, 28].
Nella configurazione del percorso i messaggi vengono prima separati dal traffico di dati corrente sulla rete e questo è un nuovo concetto. Nei mix ISDN, viene trasmesso

il messaggio a più livelli, che contiene le chiavi di sessione di ciascun mix intermedio attraverso il canale di segnalazione. Ogni mix riconosce i messaggi appartenenti alla stessa catena e utilizza la chiave di sessione per preparare la crittografia del flusso e decodificare i messaggi. I messaggi per la configurazione della route e i messaggi che contengono dati sono mescolati con altri di forma simile, quindi tutti i messaggi sono inclusi nel processo di mixaggio.
I *punti di incontro* forniscono l'anonimato del mittente e del destinatario. Il mittente potrebbe utilizzare un tag anonimo collegato a uno switch ISDN per connettersi in modo anonimo al destinatario corrente, quindi equivale a un server Nym. Nella parte di stabilimento e disconnessione della comunicazione poiché vengono utilizzate linee occupate, per evitare attacchi passivi, sono stati utilizzati canali con finestre scorrevoli per sincronizzare questi eventi. Inviando traffico reale o falso per completare la durata della finestra scorrevole, lo stabilimento e la fine della chiamata avvengono in momenti particolari, che possono essere mescolati con molti altri eventi creando l'illusione che questi due eventi accadano in qualsiasi momento. In sintesi, il modello risultante ha consentito la trasmissione di grandi volumi di flussi di dati, salvaguardando la non relatività degli input e degli output e la miscelazione di tutte le informazioni pertinenti.

2.3.8 Babel e mixmaster

Sulla base di un approccio in cui i messaggi supportano l'invio di singoli messaggi, costituendo e-mail attraverso una rete di mix, Babel [29] e Mixmaster [30] sono stati proposti a metà degli anni '90.

In Babel, le parole "percorso di inoltro" e "percorso di ritorno" forniscono rispettivamente l'anonimato al mittente e al destinatario. Il mittente crea un messaggio stratificato crittografato per la parte di invio e inoltro e include anche l'indirizzo di inoltro. In questo modo, l'identità del destinatario è nascosta e l'anonimato bidirezionale è supportato tramite il percorso di inoltro e gli indirizzi di ritorno. Poiché il processo di invio utilizza timestamp, identificatori univoci e funzioni di hashing, ha un buon livello di sicurezza che consente la protezione da messaggi duplicati. Tuttavia, la stessa struttura non viene utilizzata per il processo di ritorno poiché l'hash del corpo del messaggio non è incluso nelle informazioni sull'indirizzo di ritorno. In questo modo gli avversari possono inviare messaggi duplicati.
Un'altra caratteristica di Babel è un sistema di diversione intermix in modo che il mittente non possa riconoscere i tuoi messaggi sulla rete. Per fare ciò, i messaggi da

miscelare possono essere riconfezionati da mix intermedi e quindi inviati attraverso un percorso casuale diverso da quello assegnato dal mittente.

A differenza di Babel, Mixmaster fornisce solo l'anonimato del mittente o "percorso di inoltro". Utilizzando un processo ibrido che combina le tre crittografie RSA, EDE e 3DES, le dimensioni del messaggio non vengono modificate, il rumore casuale viene aggiunto alla fine e i messaggi vengono elaborati a livello di bit introducendo non relatività. Successivamente è stata realizzata un'altra versione, la protezione dell'intestazione crittografata con RSA è stata migliorata attraverso una funzione hash che ha bloccato gli attacchi di tag. Un'altra versione ha generato il rumore con una chiave segreta condivisa tra il rinvio e il mittente del messaggio, inoltre la chiave era inclusa nell'intestazione del messaggio. Il rumore, essendo verificabile o prevedibile dal mittente, ha permesso di includere nell'intestazione un hash dell'intero corpo del messaggio, rendendo impossibile costruire repliche e proteggerne l'integrità.
In Mixmaster i messaggi sono divisi in piccoli frammenti e inviati in modo indipendente. In ricezione, se si tratta di un mix comune, il messaggio viene ricostruito in modo trasparente e ciò evita la necessità di utilizzare programmi di invio speciali. Contro gli attacchi alla reputazione, i messaggi vengono taggati e viene utilizzata una "lista nera" con gli indirizzi da cui non si desidera ricevere e-mail anonime.

2.3.9 Mixminion e Minx: rinvio di email tipo III

Nel 2003 sono stati introdotti Mixminion e Minx [31], riunendo le idee precedenti sull'inoltro della posta anonima. Le caratteristiche di questo sistema sono che imposta la dimensione di invio a 28 kb, supporta l'anonimato del mittente e del destinatario attraverso quello che viene chiamato *blocchi di inoltro monouso*, quindi supporta l'anonimato bidirezionale combinando i due precedenti in una catena di rinvii di posta Mixminion che mischiano i messaggi corrispondenti. Un'altra caratteristica è che i rinvii intermedi non conoscono la loro posizione nel percorso del messaggio, la lunghezza del messaggio e non sono in grado di distinguere tra altri rinvii e il mittente del messaggio [25]. Questo impedisce attacchi di partizioni.
Il formato di crittografia utilizzato in Mixminion e Minx era completamente diverso perché includeva messaggi divisi in un corpo e due intestazioni principali in cui ciascuna intestazione era a sua volta divisa in sotto-intestazioni crittografate con la chiave pubblica dei mix intermedi. Lo scopo era contrastare gli attacchi di tagging [22] costituiti da un avversario attivo o da un nodo corrotto che modifica

un messaggio per rintracciarlo attraverso la modifica e compromettere l'anonimato. Mixmaster ha risolto questo tipo di problema introducendo un controllo di integrità nell'intestazione di lettura, quindi se è stata rilevata una modifica illegale, il messaggio era inviato al primo nodo onesto nel percorso di invio. A differenza del precedente, Mixminion non può utilizzare questo meccanismo a causa del supporto del routing indistinguibile dall'inoltro anonimo. A sua volta, utilizza un processo di crittografia alla fine della seconda intestazione e del corpo del messaggio, quindi è più debole.

Il pacchetto Minx ha le stesse proprietà del Mixminion a un costo inferiore in elaborazione e sovraccarico. Minx [32] utilizza un singolo passaggio di crittografia chiamato "IGE" che propaga errori di testo crittografato durante l'inoltro per rendere impossibile scoprire l'indirizzo del destinatario quando il messaggio viene modificato e decrittografato. I messaggi hanno un aspetto casuale che non consente di ottenere informazioni anche in presenza di frodi sulla rete.
Il meccanismo di trasporto utilizzato da Mixminion è TLS con uno scambio di chiavi EDH (Ephemeral Diffie-Hellman) che fornisce sicurezza negli invii. Sia gli attacchi passivi che attivi sono inutili perché rispettivamente ciò che viene ricevuto è totalmente incomprensibile e viene rilevata la modifica. Per avere successo, l'avversario deve usare diversi nodi corrotti sulla rete.
In seguito sono state proposte due idee per migliorare la sicurezza degli invii e la resistenza alle compulsioni. Il primo [31] afferma che qualsiasi comunicazione lascia una traccia delle chiavi che possono essere utilizzate per decodificare le comunicazioni future. Questa traccia viene archiviata nei mix intermedi e quando si utilizza la chiave, viene cancellata e aggiornata utilizzando una funzione di indirizzo, poiché i messaggi successivi possono dipendere dai messaggi precedenti. Un mix che cancella le sue chiavi non potrebbe decifrare i messaggi intercettati sotto costrizione. Nella seconda idea [33] un vero ricevitore di un sistema di inoltro anonimo può funzionare simultaneamente come un nodo intermedio che emula una rete punto-punto.

2.3.10 Onion routing (OR) o Routing a cipolla

Questo sistema [34, 35, 36] a differenza di una rete di mix utilizzava un routing basato su circuito, cioè c'è un messaggio che apre un circuito ed etichetta un percorso, ogni pacchetto è etichettato e segue un percorso stabilito e alla fine c'è un pacchetto che chiude il percorso. I pacchetti che percorrono questi circuiti sono generalmente chiamati *anonimo*.

Lo scopo era quello di cercare di proteggere la non relatività dei due partecipanti che comunicano attraverso terzi e proteggere le loro identità. Poiché le reti ISDN sono difficili da implementare su Internet, ciò che è stato proposto in OR è usare questa idea nella rete anonima adattandola per l'esecuzione sul modello TCP/IP.
Il primo messaggio contiene dati da condividere tra il mittente e i router, come etichette e informazioni di indirizzamento per il nodo successivo, per questo si cifra a strati. Per decrittografare, vengono utilizzate le chiavi private di una catena di router cipolla. Per evitare attacchi passivi, viene fornita la non relatività a livello di bit. Un'altra caratteristica di questo sistema era che un tipo di routing dinamico è stato sfruttato per migliorare l'anonimato in cui i router che inoltrano il flusso lungo il percorso stabilito non sono specificati solo nel messaggio iniziale.
Ogni pacchetto che viaggia sulla rete è etichettato, per indicare a quale circuito appartiene, e crittografato con chiavi simmetriche dai router. Le etichette sono diverse per ciascun collegamento, garantendo così la non relatività, e sono crittografate con una chiave condivisa tra le coppie di router OR. Pertanto gli attacchi passivi non hanno successo. Gli attacchi nel tempo sono un punto debole di questo sistema, ovvero se non c'è molto traffico, i modelli possono essere analizzati da un utente malintenzionato. Cercando di fornire l'anonimato nella navigazione web che richiede bassa latenza, tutte le dinamiche del mixer sono state rimosse. A causa dell'assenza di mix, gli attacchi superati da quest'ultimo sono minacce di questo sistema, ad esempio l'attacco di correlazione del traffico, in cui è possibile determinare la corrispondenza tra i messaggi in entrata e in uscita in un router.
I router possono essere configurati per funzionare solo con un sottoinsieme di altri router o client, in questo caso suddivisi in zone o personalizzati.

2.3.11 Tor: la seconda generazione di OR

Nel 2004 uscì un nuovo progetto di OR che chiamarono TOR o "seconda generazione del router di cipolla" [37]. Come caratteristiche aveva l'inoltro del flusso TCP su una rete di inoltro e l'uso di Privoxy, appositamente progettato per il traffico web. Alcuni server forniscono al client un elenco di server volontari, di cui verranno scelti almeno tre a caso, e che consentiranno la comunicazione. A differenza di OR, i canali bidirezionali sono formati ad ogni passo tra i nodi intermedi e il client, per sviluppare uno scambio di chiavi autenticato DF (Diffie-Hellman). Ciò garantisce l'inoltro segreto e la resistenza alla compulsione, poiché sono necessarie solo chiavi a breve termine. Questo meccanismo è stato proposto in Onion [38] e non è coperto

dal brevetto OR [35].

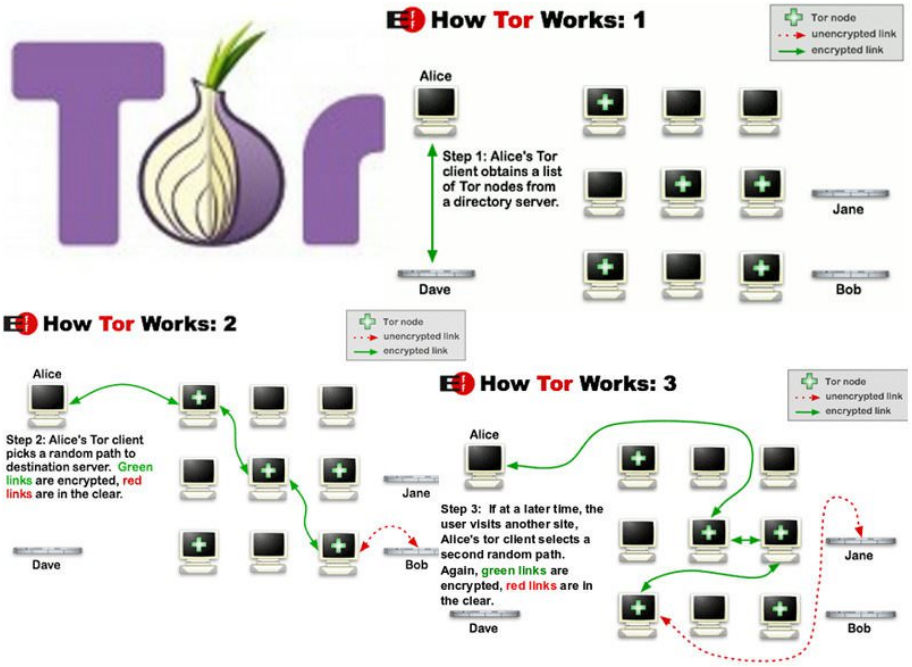

Figura 2.15: Routing a cipolla ToR

Un'altra caratteristica di TOR è che non offre sicurezza contro gli attacchi passivi globali [39, 40, 41, 42, 43], che gli studi hanno trovato molto difficili da contrastare. Due soluzioni a questo tipo di attacco sono tecniche e implicherebbero latenze elevate o richiedono l'iniezione di grandi quantità di traffico coperto. Entrambi i casi sono molto costosi e sono in contrasto con l'idea di base di TOR che vuole essere un sistema altamente utilizzabile ed economico [44, 45]. In breve, un utente malintenzionato può monitorare il flusso tra due punti della rete e generare lo stesso traffico, sfruttando un attacco di tagging.
In TOR si forniscono meccanismi per nascondere i server, ovvero il server nascosto apre una connessione anonima e la utilizza per pubblicare un punto di contatto che viene utilizzato da qualsiasi client che desidera contattare un server. Il client si connette al punto e negozia un canale anonimo che viene utilizzato per inoltrare la comunicazione corrente. Questa idea ha dimostrato di essere vulnerabile a un attacco [46] che comporta l'apertura di più connessioni allo stesso server nascosto e quindi il controllo del flusso. Per applicare ciò, l'utente malintenzionato deve controllare almeno un router che il server ha scelto come primo nodo di qualsiasi percorso anonimo.

2.3.12 Reti mix punto a punto

Man mano che cresceva l'interesse per le reti peer-to-peer, si è cercato di considerarle come un grande numero dinamico di mix. Si ricorda come detto in precedenza che nel lavoro di Chaum ogni partecipante al mix di reti funge anche da mix per gli altri, migliorando la sicurezza generale della rete.
Una rete punto-punto in cui ogni nodo agisce come un mix è stato presentato con il nome di Tarzan [47]. Questa rete consente di ottenere una connessione cifrata a "cipolla" attraverso la ripetizione di un processo. Quest'ultimo aspetta che quando un nodo inizia a trasportare un flusso crea un tunnel crittografato verso un altro nodo e gli chiede di inviare quel flusso a un altro. In questa topologia si formano strutture denominate *mimics*, ovvero ogni nodo mantiene connessioni persistenti solo con un piccolo insieme di altri nodi. Anche quando un flusso è a basso traffico, le rotte dei messaggi anonimi sono selezionate per garantire che non siano correlate. Lo schema di mimics è vulnerabile a spoofing o attacchi di spoof perché il meccanismo di selezione dei nodi vicini viene eseguito su un set di identificatori o indirizzi di rete di base.

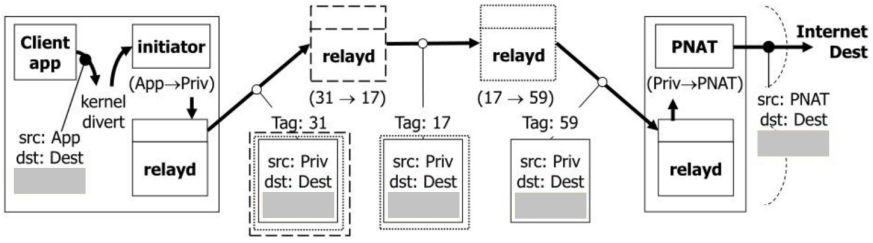

Figura 2.16: Reti mixer punto a punto Tarzan

Dato il dinamismo di una rete punto-punto, ogni nodo poteva originariamente conoscere casualmente solo il sottoinsieme di nodi con cui poteva stabilire la comunicazione. Su questa base, alcuni attacchi [48] che sono comparsi nelle prime versioni, hanno usato un nodo incluso nel percorso anonimo per determinare quale sia il nodo di origine semplicemente conoscendo tre nodi in totale: lo stesso nodo corrotto, il suo successore e il suo predecessore. Per evitare questo tipo di attacco nella versione finale di Tarzan, ogni nodo deve conoscere tutti gli altri.
Esiste un'architettura molto simile chiamata *MorphMix* [49] che differisce da Tarzan perché la rotta della comunicazione anonima è specificata in un nodo intermedio in cui l'emittente si fida ma non nell'origine. Terze parti o testimoni osservano il

nodo intermedio fa questo e l'attacco precedente potrebbe applicarsi nel processo di selezione dei testimoni. Un'altra caratteristica di MorphMix è un meccanismo di rilevamento delle collisioni che monitora qualsiasi ciclo nella selezione dei nodi di rotta.

2.4 Sistemi Anonimi Globali

Quando si progetta in primo luogo un sistema anonimo, in ciascun caso particolare vengono prese in considerazione le esigenze specifiche di comunicazione per determinare le caratteristiche del sistema in termini di struttura e politica di trasmissione. Alcuni esempi di esigenze sono la latenza bassa o alta, l'anonimato alto o basso, l'infrastruttura centralizzata o distribuita.
Se parliamo di sistemi globalizzati, i bisogni specifici non possono essere considerati perché per natura è eterogeneo, quindi sarebbe difficile progettarlo. È possibile mettere insieme alcune linee guida che possono essere utilizzate come base generale per adattarle a casi specifici. Per esempio:

- Sviluppare meccanismi per fornire livelli più elevati di anonimato senza ridurre la latenza.
- Sviluppare meccanismi che contrastano i sistemi di blocco e censura e risolvono anche il problema di scoprire i nodi iniziali per accedere alla rete anonima.
- Valutare continuamente le prestazioni per creare un modello e agire.

In ogni sottocapitolo successivo, viene proposta una soluzione relativa a ciascuno dei punti precedenti. Un buon modo per creare un sistema finale consiste nell'utilizzare l'ultima soluzione, ovvero un modello di ottimizzazione per valutare un sistema, in combinazione con le idee di soluzioni incomplete.

2.4.1 Progetto di un sistema makoviano

Creare un sistema che sia un compromesso tra bassa latenza e alto livello di anonimato non è affatto facile ed è un problema nato con l'intenzione di comunicare in modo anonimo.
Una prima soluzione è stata quella di utilizzare reti miste che offrano un elevato livello di anonimato, ma con gli svantaggi presentati sopra. Al fine di risolverli, è stato proposto un meccanismo di inoltro, che consiste nell'accumulare messaggi per brevi periodi e quindi inoltrarli alla loro destinazione successiva seguendo un ordine diverso dal loro arrivo in modo che la relazione input/output dei messaggi non sia ovvia. La politica di spedizione non deve essere chiaramente FIFO e può essere attuata seguendo un processo markoviano mostrato di seguito.
Quando un nodo intermedio riceve un messaggio, lo memorizza e il momento dell'invio viene determinato probabilisticamente dopo i suoi stati precedenti, che possono

essere rappresentati attraverso una catena di Markov. La probabilità di invio aumenta ogni volta che il messaggio rimane in memoria, vale a dire più è alto, meno tempo rimarrà in memoria.
È necessario capire qual è la probabilità di base e il tasso di aumento per ogni istante di tempo trascorso in memoria. Ad esempio, la probabilità di base potrebbe essere l'ora di arrivo del messaggio.

2.4.2 Modello per la comunicazione anonima in un ambiente restrittivo

È difficile per un utente in un ambiente limitato scoprire dinamicamente i nodi che consentono l'accesso alla rete anonima e le soluzioni pratiche finora non sono state soddisfacenti. Nella maggior parte dei casi, il meccanismo di rilevamento di questi nodi non è efficiente perché se funzionano come proxy, saranno bloccati da enti regolatori.
Il meccanismo di rilevamento deve dare continuità alle comunicazioni utente anonime perché se alcuni nodi sono bloccati, altri devono essere disponibili. Sarebbe bene ricordare che:

- I nodi a cui accedere non sono sempre infiniti.
- I nodi disponibili in un determinato momento potrebbero non essere disponibili in un altro perché bloccati.
- Se gli enti regolatori sono mascherati da utenti normali, possono ottenere informazioni dalla rete e quindi bloccare i nodi rilevati.

2.4.3 Modello per ottimizzare sistemi anonimi

In pratica, se i livelli di anonimato vengono aumentati, di conseguenza, la latenza aumenta e, al contrario, se la latenza diminuisce, il livello di anonimato segue la stessa direzione. Pertanto, è necessario creare un modello matematico in cui si trova un compromesso in cui si incontrano livelli accettabili dei casi precedenti. I calcoli dovrebbero essere effettuati con variabili che possono essere utilizzate in un ciclo e in numeri che, nel caso globalizzato, possono essere molti. Una semplificazione può essere quella di prendere come variabili: numero di nodi intermedi, meccanismo di anonimato e tipi di attacchi.
Ognuna di queste variabili e le loro combinazioni dovrebbero esserci per ciascuno

dei casi in cui vogliono essere implementati: ad esempio, in un sistema di servizi web, il tempo di risposta non deve superare un certo limite massimo. In sintesi, è necessario cercare le variabili, stabilire le restrizioni e creare le funzioni oggettive all'interno delle categorie proposte.

Capitolo 3

Sviluppo

Il progetto realizzato consiste nella creazione di un'infrastruttura di un sistema riservato e anonimo di trasporto con rotte dinamiche e aleatorie affinché due dispositivi qualunque, separati da una rete intermedia, possano comunicare. Lo scopo della rete intermedia è permettere che il traffico viaggi efficientemente e in un ambiente sicuro e affidabile.

Trovare un equilibrio tra sicurezza e velocità delle comunicazioni è stato il punto chiave, poiché l'uso eccessivo di misure che miglioravano un aspetto, influivano negativamente sull'altro e viceversa. Dati due obiettivi contrapposti, l'ottimizzazione di entrambi separatamente non si poteva raggiungere quando il sistema funzionava in un ambiente reale, per questo bisognava incontrare un punto stabile. Trovare un compromesso è la radice del problema dei sistemi anonimi implementati fino ad oggi. Per questo nel progetto si è pensato a un sistema che si adatti a molti cambi e circostanza, permettendo che sia scalabile, versatile, facile da modificare. Detto diversamente, non serviva una soluzione unica e universale bensì una versione che si potesse adottare a differenti mezzi e migliorare il suo rendimento dinamicamente attraverso l'apprendimento. Tutto questo affinché il sistema fosse fattibile tanto a corto come a lungo termine.

Per l'ingegneria del software, tutti i processi dello sviluppo si dividono in fasi che a loro volta si compongono di compiti specifici. I risultati raccolti nel tempo dicono che quante più risorse si dedicano alle prime fasi, che si concentrano nello studio del sistema a livello di specifiche, requisiti e pianificazione; migliore sarà il risultato in quanto più elaborato. Il valore che portano queste tecniche è notevolmente più visibile durante la fase di manutenzione del sistema. Riassumendo, di tutte le ore dedicate al progetto, molte di queste si dedicarono alla fase di analisi dei requisiti.

Si presentò un conflitto tra i requisiti funzionali e non. I primi sono quelli che definiscono il comportamento del programma, descrivono i servizi che offre e determinano i risultati; i secondi sono quelli che si avventurano su come si raggiunge la soluzione e sotto quali restrizioni.

I requisiti funzionali dell'infrastruttura possono essere riassunti in:

- L'infrastruttura include una rete di comunicazione multi-punto tra nodi posizionati casualmente.
- Attraverso protocolli opportuni si garantirà la sicurezza e l'affidabilità delle comunicazioni.
- Esisterà un meccanismo che aggiorna la rete e che le permetterà di gestire i dati. Questo meccanismo sarà un algoritmo di routing che calcolerà le route ottime tra i nodi.

L'infrastruttura, come si può dedurre, sarà composta da piccoli blocchi che saranno eseguiti individualmente e che dovranno collaborare per il corretto funzionamento dell'insieme. La collaborazione tra questi si consegue mediante tecniche di sincronizzazione tra thread, cioè, implementando semafori per evitare race condition nelle sezioni critiche. I requisiti non funzionali, cioè quelli che si avventurano su come si raggiunge la soluzione e sotto quali restrizioni, si inseriscono di seguito:

- Affidabilità: è il grado con il quale un sistema raggiunge o meno i requisiti specificati.
- Sicurezza: si riferisce alla protezione contro attacchi che possono produrre danni da parte di terze parti o software.
- Robustezza: si intende la capacità del sistema di resistere in ambienti estremi di ogni tipo.
- Disponibilità: quanto minore sia il tempo di utilizzo, migliore saranno le prestazioni.
- Capacità: è la quantità di utenti che possono utilizzare il sistema allo stesso tempo. Serve come indicatore per il carico che può sopportare in un momento di punta di lavoro senza che si riduca la qualità del servizio.
- Tempo di risposta: si può definire come la latenza tra una richiesta di un usuario e la risposta data dal sistema.

- Rendimento: misura la capacità di fare i compiti con il minor consumo di risorse possibili.
- Produttività: si riferisce alla quantità di dati o transazioni processate per unità di tempo.
- Usabilità: è il grado di integrazione del sistema nei compiti che realizza l'utente.
- Testabile: è la capacità di essere provato con facilità per poter individuare e correggere errori, analizzare incoerenze e estrarre conclusioni dai valori raccolti.
- Portabilità: si definisce come la capacità di essere trasformato con facilità in un'altra piattaforma hardware e/o software.
- Scalabilità: esprime il livello di miglioramento e ampliamento che ha fisicamente un sistema di modo che le prestazioni attuali migliorino.

Tra tutte le alternative disponibili, si necessita una soluzione che è un compromesso tra tutti i requisiti precedenti, inoltre più funzionalità si aggiungono a un sistema, più difficile è il suo sviluppo e mantenimento successivo, dato che richiedono migliorie, progettazione e nuove analisi di ambiente. Riassumendo, bisogna decidere in quali si sacrificano garanzie in cambio di rapidità, che protocolli si devono utilizzare e come lavoreranno affinché tutto il sistema risponda alle aspettative. Lo scopo di questa applicazione non è unico, bensì ci si aspetta che funzioni come servizio globale per qualunque rete. Con il fine di favorire la portabilità e la scalabilità del sistema e, di conseguenza, di prolungare il più possibile il suo ciclo di vita, si aggiunge più difficoltà e lavoro al progetto.

Nelle sezioni successive si definiscono gli strumenti utilizzati per lo sviluppo, le entità che formano l'infrastruttura, i loro compiti e le fasi delle quali consta tutto il processo. Si presenta, di seguito, una piattaforma di comunicazione, però non si specifica cosa si farà su essa, bensì si permette una configurazione libera affinché possa essere sfruttata in diversi contesti.

3.1 Strumenti utilizzati

Per realizzare lo sviluppo del progetto sono stati utilizzati diversi tool che si vanno a riassumere successivamente:

Standard :

- Java Cryptography Extension (JCE): fornisce funzioni crittografiche e si può trovare nel pacchetto javax.crypto.
- Java Secure Sockets Extension (JSSE): classi per implementare il protocollo SSL dentro la piattaforma Java, si può trovare nel pacchetto javax.net.
- Java Authentication and Authorization Service (JAAS): permette l'autenticazione dentro la piattaforma Java, si può incontrare nel pacchetto javax.security.

OpenSSL : è un tool distribuito usando una licenza di codice free. Si può usare in due forme: linea di comando e libreria crittografica. Questo tool si utilizza per: creare e gestire chiavi private/pubbliche, creazione e gestione di certificati X.509, calcolare sintesi di messaggi mediante funzioni di hash (firme digitali), cifrari simmetrici con un ampio insieme di algoritmi, sviluppo di applicazioni SSL/TLS client-server e gestione di messaggi S/MIME cifrati e firmati. In questo progetto è stato utilizzato per creare il certificato dell'Autorità di certificazione e firmare i certificati dei client e dei server.

KeyTool : è un tool di gestione di chiavi e certificati in java. Si utilizza per generare coppie di chiavi RSA e certificati autofirmati X.509, fornisce l'infrastruttura per utilizzare SSL nel codice Java e mettere le chiavi e i certificati in un deposito, anche chiamato keystore. Un keystore è un file protetto con una chiave che contiene due tipi di entry: "key entries", al cui interno ci sono la chiave privata e il certificato associato, e "truested certificate entries", che è il certificato di chiave pubblica di un'entità (nel presente caso l'Autorità di certificazione). Dato che in questo progetto si richiede l'autenticazione del client, sia il client che il server devono avere un Almacen e un AlmacenTrust che contiene solo il certificato di chiave pubblica dell'Autorità di certificazione (AC). Quando si stabilisce una connessione SSL, il server invia il suo certificato firmato da una AC al client e quest'ultimo verifica se è autentico grazie al suo AlmacentTrust.

D'altra parte, il server richiede l'autenticazione del client e riceve il suo Almacen che verifica con il suo AlmacenTrust. Nella figura seguente si possono vedere gli Almacen, la password usata per proteggerli, l'inizializzazione di un ServerSocketSSl o di un SocketSSL, la richiesta di autenticazione del client e il cifrario che si vuole utilizzare.

```
System.setProperty("javax.net.ssl.keyStore","Almacen");
System.setProperty("javax.net.ssl.keyStorePassword","oooooo");
System.setProperty("javax.net.ssl.trustStore","AlmacenTrust");
SSLServerSocketFactory sslserversocketfactory = (SSLServerSocketFactory)
    SSLServerSocketFactory.getDefault();
sslserversocket =(SSLServerSocket) sslserversocketfactory.
    createServerSocket(n.getPuerto());
sslserversocket.setNeedClientAuth(true);
String [] cifradores= sslserversocket.getSupportedCipherSuites();
String [] mycifrador  = {cifradores[1]};
sslserversocket.setEnabledCipherSuites(mycifrador);
/*String [] cifradores_enable =sslserversocket.getEnabledCipherSuites();
for (int i=0; i<cifradores_enable.length; i++)
   System.out.println("DESTINO CIFRADORES:\t" + cifradores_enable[i]);*/
```

Algoritmo di Dijkstra : nella maggior parte delle reti attuali si utilizzano algoritmi di routing dinamici perché il numero di router e dei loro link cambiano per motivi diversi, ad esempio è attivo il protocollo di spanning tree. Tra i diversi algoritmi di routing ci sono quelli isolati, distribuiti e centralizzati. In questo progetto è stato usato l'ultimo tipo citato, nel quale i nodi della rete inviano a un'entità interna o esterna il proprio stato affinché calcoli le loro tabelle di routing. Nel presente caso l'entità è esterna ed è il Sistema Gestore delle Rotte (SGR). Il SGR prepara tutte le strutture necessarie per eseguire l'algoritmo di calcolo delle route ottime. Con l'informazione di stato della rete fornita da ogni nodo e la tabella dei costi dei link, si calcola la tabella di routing ottima per tutta la rete utilizzando un algoritmo che è una modifica di quello dei cammini minimi di Dijkstra secondo il procedimento seguente:

1. Per ogni nodo si sceglie lo stesso come nodo iniziale mettendo distanza attuale 0 e il resto con infinito perché sono sconosciute all'inizio.
2. Stabilisce il nodo non visitato con la distanza attuale minore come il nodo attuale A.

3. Per ogni vicino V del nodo attuale A: somma la distanza attuale del nodo A con il peso del link che connette A con V. Se il risultato è minore della distanza attuale di V, stabilisce la nuova distanza attuale di V.
4. Marca il nodo attuale A come visitato.
5. Se ci sono nodi non visitati, tornare al passo 2.

```
//Dijkstra algorithm
public void computePaths(Map<Integer,Nodo> red){
   shortestDistances.put(puerto, 0.0);

   //implement a priority queue
   Queue<Nodo> queue = new LinkedList<Nodo>();
   queue.add(this);

   while(!queue.isEmpty()){
      Nodo u = queue.poll();

      //visit the adjacencies, starting from
      //the nearest node(smallest shortestDistance)

      for(Edge e: u.adjacencies.values().stream()
            .filter(Edge::getActive).collect(Collectors.toList())){

         Nodo v = red.get(e.getTarget());
         double weight = e.getWeight();

         //relax(u,v,weight)
         double distanceFromU = u.shortestDistances.get(puerto)+weight;
         if(distanceFromU<v.shortestDistances.get(puerto)){
            //remove v from queue for updating
            //the shortestDistance value
            queue.remove(v);
            v.shortestDistances.put(puerto, distanceFromU);
            v.parents.put(puerto,u);
            queue.add(v);
            queue = queue.stream().sorted((n1, n2)->
                  Double.compare(n1.shortestDistances.get(puerto),
                        n2.shortestDistances.get(puerto)))
                  .collect(Collectors.toCollection(LinkedList::new));
         }
      }
   }
```

```
}
```

Crittografia Simmetrica : per permettere lo scambio di messaggi UDP cifrati, si sfrutta questo tipo di crittografia nella quale c'è una chiave, conosciuta da emittente e ricevente, che è la stessa per cifrare e decifrare i messaggi. Lasciare la sicurezza nelle mani degli utenti non è sicuro. I dati necessitano di essere protetti con un chiave casuale e abbastanza lunga, cioè con un'alta entropia. Per questo non si consiglia di utilizzare una password per cifrare direttamente i dati. Una soluzione è usare una funzione chiamata funzione di generazione di chiavi basata su password (PBKDF2), che crea una chiave a partire da una password facendo hash molte volte con un sale. Un sale è una sequenza casuale di dati e fa in modo che una chiave derivata sia unica incluso se un'altra persona usa la stessa password. In questo progetto per generare la chiave in Java si è fatto uso dei pacchetti javax.crypto e java.security. In particolare, come si può vedere nell'immagine seguente si crea un oggetto della classe SecureRandom che fornisce una sequenza casuale crittografica sicura che sarà il sale. Ora si può porre il sale e la password in un oggetto di cifratura basato su password PBEKeySpec. Il generatore di oggetti utilizza anche una dimensione di chiave e un modulo di conteggi di iterazioni per fortificare la chiave. Di fatto, aumentare il numero di iterazione aumenta il tempo necessario per azionare un set di chiavi durante un attacco di forza bruta. La PBEKeySpec si passa a SecretKeyFactory, che finalmente genera la chiave come matrice di byte[].

Creata la chiave si possono creare cifrari e gli inversi che in questo progetto utilizzano un algoritmo di cifratura/decifratura autenticato AES con modalità di operazione GCM (Galois-Counter Mode). Si utilizza un'etichetta di autenticazione con una lunghezza di 16 ottetti (128 bits). Il testo cifrato con AES128+GCM consiste nell'aggiungere il tag di autenticazione fornito come output dall'operazione di crittografia GCM al testo crittografato emesso da tale operazione. L'ultima parte di questa catena si riferisce al modo di riempimento del blocco che in questo caso non si specifica.

```
//Key creation to encipher and decipher
SecureRandom random = new SecureRandom();
byte[] salt = new byte[128];
random.nextBytes(salt);
SecretKeyFactory skf = SecretKeyFactory.getInstance("
    PBEWithHmacSHA256AndAES_128");
```

```
KeySpec keyspec = new PBEKeySpec("UPMTFM_CLAVE_AES".toCharArray(), salt,
    65536, 128);
byte[] keybyte = skf.generateSecret(keyspec).getEncoded();
private SecretKey key = new SecretKeySpec(keybyte, "AES");
```

Le operazioni di cifratura e decifratura sono state realizzate attraverso due funzioni. L'operazione di cifratura prende un messaggio di tipo String e restituisce un ArrayByte pronto per essere inviato. L'operazione di decifratura fa esattamente il contrario. É stato utilizzato un vettore di inizializzazione che è un blocco di bytes casuali che dovrà essere in XOR con il primo blocco di dati dell'utente. Dato che ogni blocco dipende da tutti gli altri blocchi processati fino a quel punto, tutto il messaggio sarà cifrato in maniera unica; i messaggi identici cifrati con la stessa chiave non produrranno risultati identici. Quando il ricevente dovrà decifrare necessiterà lo stesso vettore di inizializzazione, quindi questo si aggiunge al messaggio cifrato quando è inviato.

```
private cipherEncrypt = Cipher.getInstance("AES/GCM/NoPadding");
private cipherDecrypt = Cipher.getInstance("AES/GCM/NoPadding");
...
public byte[] encrypt(String sendMensaje) throws Exception {
   byte[] iv = new byte[GCM_IV_LENGTH];
      //Initialization vector
   (new SecureRandom()).nextBytes(iv);
   GCMParameterSpec ivspec = new GCMParameterSpec(GCM_TAG_LENGTH * Byte.
      SIZE, iv);     //GCM parameters to encrypt/decrypt
   cipherEncrypt.init(Cipher.ENCRYPT_MODE, key, ivspec);
         //Mode encrypt
   byte[] cipherMensaje = cipherEncrypt.doFinal(sendMensaje.getBytes());
               //Cipher message
   byte[] encrypted = new byte[iv.length + cipherMensaje.length];
    System.arraycopy(iv, 0, encrypted, 0, iv.length);
    System.arraycopy(cipherMensaje, 0, encrypted, iv.length, cipherMensaje.
       length);
    return Base64.getEncoder().encodeToString(encrypted).getBytes();
             //Iv + cipher message
}
public String decrypt(byte[] recvBuf) throws Exception {
   byte[] cipherMensaje = Base64.getMimeDecoder().decode(new String(recvBuf
      ));
   byte[] iv = Arrays.copyOfRange(cipherMensaje, 0, GCM_IV_LENGTH);
    GCMParameterSpec ivSpec = new GCMParameterSpec(GCM_TAG_LENGTH * Byte.
       SIZE, iv);                //GCM parameters to encrypt/decrypt
```

```
    cipherDecrypt.init(Cipher.DECRYPT_MODE, key, ivSpec);
                        //Mode decrypt
    return new String(cipherDecrypt.doFinal(cipherMensaje, GCM_IV_LENGTH,
        cipherMensaje.length - GCM_IV_LENGTH)); //Plain message
}
```

Criptografía Asimétrica : Si potrebbe utilizzare una crittografia asimmetrica per distribuire la chiave condivida tra le entità per cifrare i messaggi UDP. Qualunque entità, che può essere un dispositivo dell'utente o un router, prima di inviare rispettivamente la richiesta o il proprio stato, invia un altro messaggio per notificare al SGR di voler entrare nella rete anonima. A questo messaggio si aggiunge la propria chiave pubblica che il SGR usa per cifrare la chiave simmetrica. In questo modo solo l'emittente del messaggio può decifrarlo e si soddisfa l'obiettivo della crittografia simmetrica. Anche se la crittografia a chiave privata sarebbe più forte contro gli attacchi, si preferisce non usarla perché richiede più capacità di calcolo e tempo per cifrare/decifrare un messaggio. Dato che si vuole velocità quando si interagisce con il SGR, si utilizza UDP cifrato con una chiave simmetrica. Una soluzione ulteriore consiste nell'incorporare il SGR in un sistema di distribuzione di chiavi mediante crittografia quantica, denominato QKD (Quantum Key Distribution).

Guasti : Per creare una rete che sia il più possibile simile a una rete reale è stata creata una classe Java che simula un evento nel quale un link si rompe attraverso una variabile booleana. Si è scelta una probabilità di guasto del 5%, che è abbastanza alta rispetto alla realtà, questo perché si volevano fare prove che presentassero guasti, in modo da vedere la risposta del sistema. Quando un link si rompe (A -> B), al simmetrico succede lo stesso (B -> A).

```
//Creation of problems in links of network
for(int i=0;i<95;i++)   probabilidad.add(true);
for(int i=0;i<5;i++) probabilidad.add(false);//Error 5%
...
while(true) {
   try {
      nodos.values().stream().map(Nodo::getAdjacencies)  //Stream<Map>
            .map(Map::values)                    //Stream<Collection<Edge>>
            .flatMap(Collection::stream)            //Stream<Edge>
            .forEach(e -> e.setActive(probabilidad.get(random.nextInt(
               probabilidad.size()))));
```

```
        //symmetric problems between the nodes
        for(Nodo n : nodos.values())
            for(Edge e : n.getAdjacencies().values())
                nodos.get(e.getTarget()).getAdjacencies().get(n.getPuerto()).
                    setActive(e.getActive());

        synchronized(Red.semaforoRP){//wake up the network
            Red.semaforoRP.notify();
        }
        Thread.sleep(15000);
    } catch (Exception e) {
        System.out.println("ERROR PROBLEM:\t" + e.toString());
    }
}
```

3.2 Scenario

In questa sezione si descrive l'infrastruttura in maniera generale, cioè il contesto nel quale è posizionata e i protocolli con cui le entità interagiscono. Le entità presenti nel sistema realizzano i compiti per i quali furono progettati attraverso la cooperazione e la sincronizzazione tra essi, cioè si intrecciano tra loro e con il resto delle entità. Uno schema generale dell'infrastruttura si mostra di seguito.

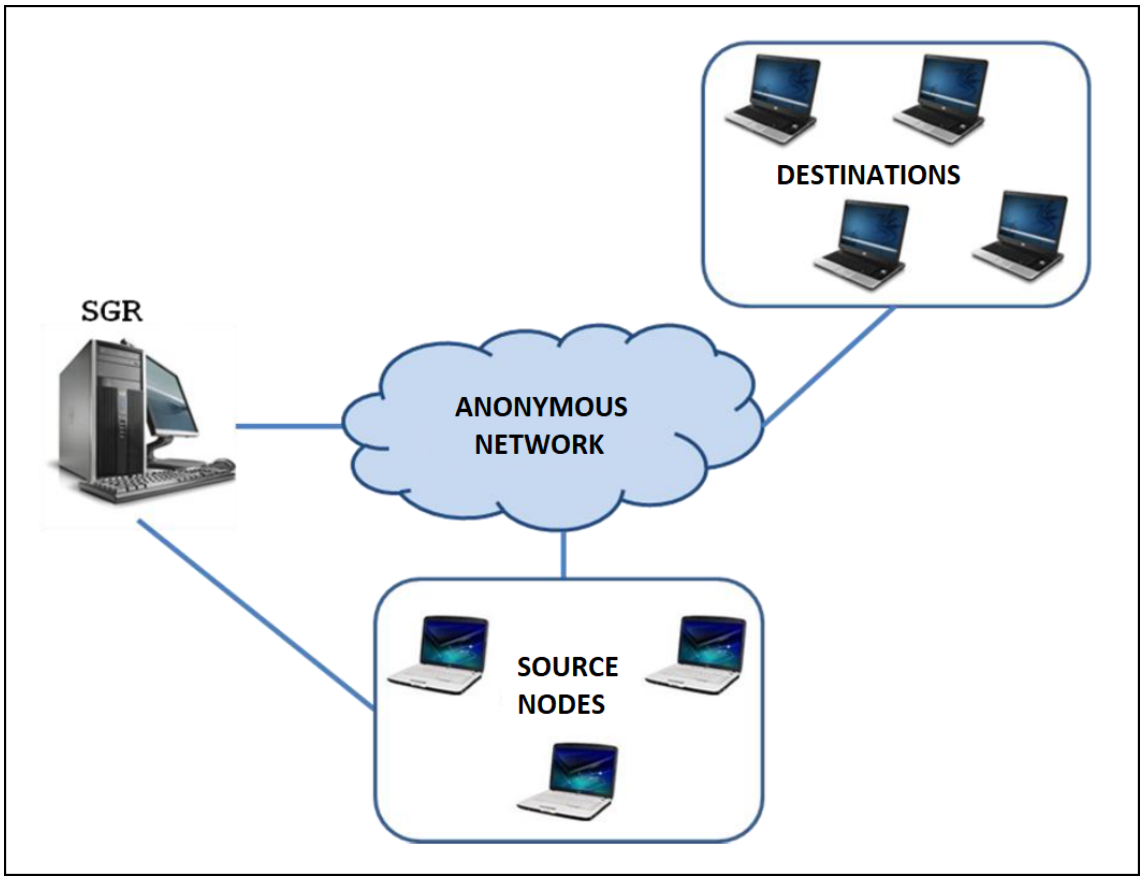

Figura 3.1: Scenario globale del Sistema Anonimo

Nell'immagine ci sono diversi elementi che rappresentano una rete globale, questo perché si vuole una soluzione il più possibile simile alla realtà. Dato che si vuole realizzare un'applicazione che risponde in tempo reale alle richieste dell'utente, si necessita un livello di interazione molto alto e la realizzazione del processo automaticamente in ogni momento. Ora si descrivono alcune caratteristiche dell'infrastruttura:

- Il modello utilizzato è del tipo client-server tra i partecipanti perché prima assumono il ruolo di agente attivo che genera l'informazione e dopo di agente passivo che la riceve.

- Si utilizza il modello TCP/IP per la trasmissione dell'informazione da un'origine fino alla destinazione attraverso la rete anonima.

- L'entità Sistema Gestore delle Rotte (SGR) svolge la maggior parte dei compiti che sono: raccogliere le informazioni inviate dai nodi della rete anonima, calcolare il percorso ottimo e mantenere informati i dispositivi finali e intermedi.

La situazione che si creerà avrà due estremi, cioè uno iniziale e uno finale, che vorranno comunicare attraverso una connessione sicura che coinvolge i nodi della rete anonima. In particolare, un estremo è l'origine delle comunicazioni e forma parte della rete interna, il secondo è la destinazione e rappresenta la frontiera con l'esterno, però può essere utilizzato per allargare il sistema.
Il messaggio viaggia per la rete anonima seguendo una route che è ottima perché calcolata dall'entità SGR tenendo in conto la condizione della rete in un istante di tempo. Per calcolare la route, il SGR riceve le informazioni dai nodi riguardo lo stato dei link, esegue l'algoritmo e gli consegna l'informazione di routing.
Una volta che i dispositivi hanno caricato i file di configurazione possono stabilire connessioni. Il nodo origine chiede il first hop al SGR e quando lo riceve si connette a esso per inviare il messaggio attraverso la rete anonima. A partire da questo momento, il pacchetto segue la route ottima ottenuta precedentemente fino a che non esce dalla rete anonima, dopo segue le classiche tecniche di routing.

Nell'infrastruttura tra le entità ci sono diverse relazioni che esigono diverse necessità, in particolare, c'è l'interazione dei dispositivi finali e la rete anonima con il SGR e l'interazione degli estremi con la rete. Nel primo caso lo scambio di messaggi deve essere il più rapido possibile, per questo la soluzione adottata è il protocollo UDP che è semplice e include meccanismi di sincronizzazione. Nel secondo caso si scambiano dati per questo si esige stabilità e protezione, quindi si sceglie il protocollo SSL, per l'autenticazione delle entità, e il protocollo TCP per offrire affidabilità nell'invio e nella ricezione. Si puntualizza che l'utilizzo del protocollo UDP, non essendo orientato alla connessione non fornisce mezzi di sicurezza, quindi si cifra l'informazione che viaggia per la rete. D'altra parte, si aggiunge una misura di protezione, però non si incrementa il carico di lavoro a causa della semplicità della

stessa. Nella figura 3.2 seguente si possono vedere i dettagli dei protocolli di livello di trasporto utilizzati nell'infrastruttura.

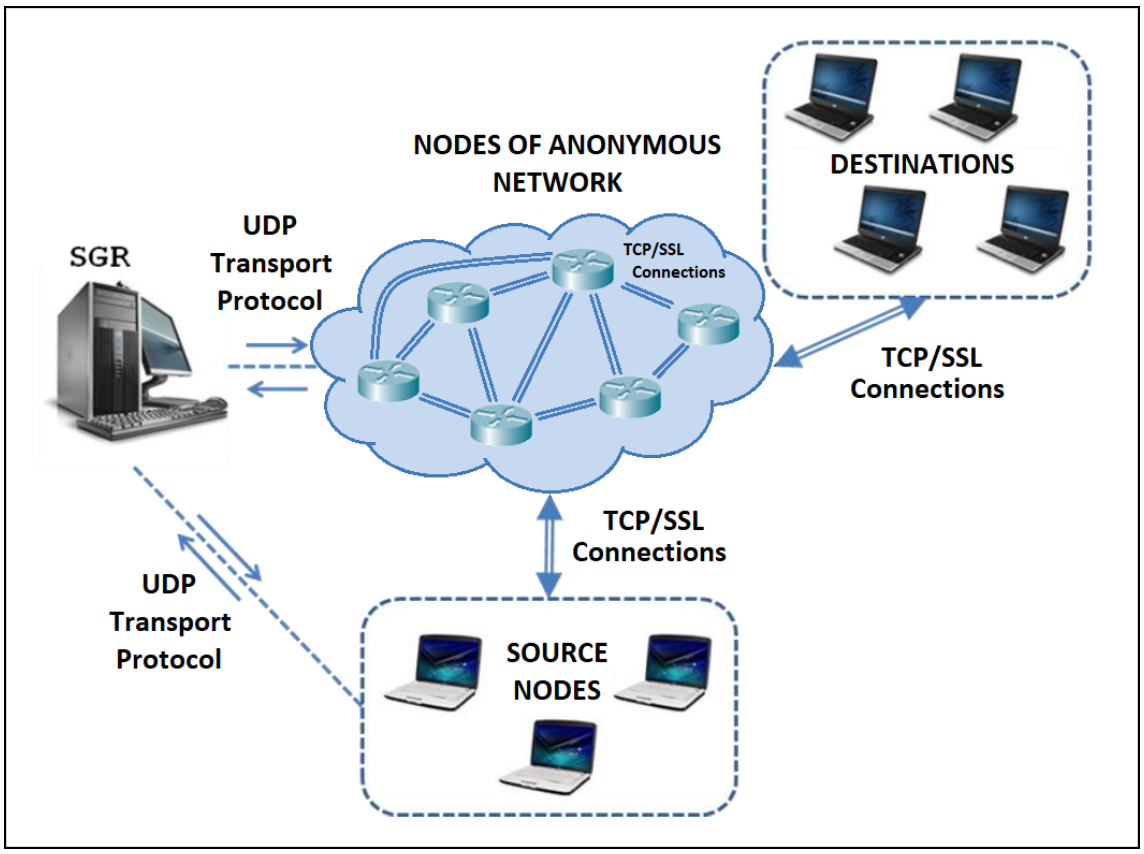

Figura 3.2: Vista dettagliata dei protocolli di comunicazione del sistema

3.3 Entità del sistema

In questa sezione si da enfasi alle entità che formano l'infrastruttura e alle loro funzioni. Come detto precedentemente le entità che si possono trovare nel sistema sono:

- Il Sistema Gestore delle Rotte (SGR).
- Gli estremi della comunicazione.
- I nodi della red anonima.

3.3.1 Sistema Gestore delle Rotte (SGR)

Il Sistema Gestore delle Rotte è l'entità che realizza più funzioni di tutte perché deve comunicare sia con il nodo che vuole inviare dati anonimamente sia con i nodi della rete anonima. I diversi compiti sono svolti da thread quindi richiedono sincronizzazione tra essi. Lo scambio di messaggi con l'origine e i nodi utilizzano il protocollo UDP, perché si esige velocità, integrato con la cifratura dei dati secondo un algoritmo di tipo simmetrico. D'altra parte, si deve specificare che il SGR ha un suo indirizzo pubblico che le altre entità possono utilizzare per inviargli pacchetti.

I compiti che realizza si possono riassumere in passi che si ripetono quasi periodicamente:

1. Raccoglie le informazioni di stato di ognuno dei nodi della rete.
2. Riceve la richiesta da un'origine.
3. Esegue l'algoritmo di Dijkstra e invia le tabelle ai nodi intermedi.
4. Invia all'origine l'indirizzo de nodo casuale della rete anonima che sarà utilizzato come first hop.

3.3.2 Estremi della rete

Gli estremi della rete sono due dispositivi di cui il primo vuole inviare un messaggio in maniera anonima al secondo. In dettaglio:

- **Nodo origine**: É munito di un deposito che contiene la sua chiave privata e il suo certificato firmato da una Autorità di certificazione. Con esso, si possono

assicurare le comunicazioni in termini di autenticazione a cui si aggiunge la cifratura dell'informazione con il protocollo crittografico dello strato di trasporto SSL. L'origine invia una richiesta al SGR che contiene l'indirizzo del nodo a cui vuole inviare un messaggio. Attende che il SGR risponda con il first hop e dopo stabilisce una connessione con il nodo ricevuto. Si crea un tunnel affinché tutto quello che si scrive da tastiera arrivi al nodo selezionato.

- **Nodo destino**: Come il nodo origine ha un deposito che contiene la sua chiave privata e il suo certificato firmato da una Autorità di certificazione. Con esso, si possono assicurare le comunicazioni in termini di autenticazione a cui si aggiunge la cifratura dell'informazione con il protocollo crittografico dello strato di trasporto SSL. Se la configurazione del nodo è stata realizzata senza errori, questo svolgerà una funzione di server bloccandosi in attesa di connessione. Una volta ricevuto il messaggio può processarlo. Essendo ubicato in una parte del mondo, la posizione della destinazione determinerà l'ultimo salto della rete anonima.

3.3.3 Nodi interni della rete anonima

Come tutte le reti, c'è un insieme di router che permettono di spostare un pacchetto da un'origine a una destinazione, ed è così anche in una rete anonima. É stata fatta una rete completamente scalabile affinché sia il più possibile simile alla realtà. Al contrario delle reti classiche, in questa rete anonima la connessione del livello di trasporto che si stabilisce non è tra gli estremi ma tra ogni entità che gestisce i dati che devono arrivare a destinazione. Ogni router sarà sia server che client in momenti successivi, quindi ha un deposito con la sua chiave privata e il suo certificato firmato da un'Autorità di certificazione. Con esso, si possono assicurare le comunicazioni in termini di autenticazione a cui si aggiunge la cifratura dell'informazione con il protocollo crittografico dello strato di trasporto SSL.

Una volta che ogni nodo si accende e termina la fase di configurazione dei certificati, crea tre thread per svolgere i suoi compiti. Un figlio informerà periodicamente il SGR riguardo lo stato dei suoi link attraverso pacchetti UDP cifrati. I due che rimangono rimarranno bloccati perché uno di essi deve aspettare i messaggi UDP con la route ottima inviati dal SGR e l'altro sarà il server delle connessioni SSL che arrivano. Quando arriva una connessione, il thread server si converte in client sia per inviare il messaggio alla destinazione sia per inviarlo a un altro nodo della rete.

Utilizzando UDP non si garantisce che i datagram arrivino. Non sarebbe un problema nemmeno la perdita di qualche pacchetto per qualunque motivo, perché il processo di invio degli stati è gestito da temporizzatori. Allo scadere effettuano la ritrasmissione dei messaggi per sanare gli errori che si sono presentati e fare in modo che l'informazione di stato sia il più aggiornata possibile.

Come nella realtà, si possono presentare casi nei quali i link si rompono mentre il pacchetto sta viaggiando per la rete. Si è creato un meccanismo di richiesta di route al SGR attraverso UDP per garantire velocità e poter continuare la trasmissione dei messaggi. Appena arriva il messaggio, si aggiorna la route in memoria e lo si invia al next hop.

In questo progetto non si tiene in conto il caso in cui un router si rompe perché è stato classificato come poco probabile, quindi quando si parlerà di guasti si devono intendere relativi ai link.

3.4 Protocollo

In Internet esistono molti protocolli che permettono a due dispositivi di comunicare attraverso lo scambio di messaggi. Gli scopi dei protocolli sono diversi, per esempio "spanning tree" permette in una rete di essere libera da cicli. In questo progetto le entità le entità devono scambiarsi messaggi, per questo è stato creato un protocollo molto semplice e simile a ICMP, cioè ogni messaggio è composto da un codice che lo identifica e un corpo con ulteriori informazioni. É un protocollo di livello applicazione e si incapsula in UDP come protocollo di livello di trasporto. Dato che si utilizza UDP come mezzo di trasporto, non è sicuro che i messaggi arrivino a destinazione, per questo si utilizzano meccanismi di rinvio periodico o temporizzati. I codici che si possono trovare:

Codice 1 : Indica una richiesta al SGR da un utente. Il corpo del messaggio contiene l'indirizzo IP alla quale l'utente vuole inviare un messaggio anonimo. Appena riceve questo messaggio, il SGR configura le sue strutture interne e esegue l'algoritmo di Dijkstra per calcolare i percorsi minimi tra l'utente, che sarebbe l'origine, e la destinazione scelta.

Codice 2 : É la risposta a un messaggio con codice 1 e contiene l'indirizzo IP del first hop scelto casualmente dal SGR. Quando un'origine riceve un messaggio con questo codice può aprire una connessione TCP/SSL con il nodo ricevuto e successivamente inviare il messaggio inserito da tastiera.

Codice 3 : Indica al SGR lo stato dei link e le destinazioni che può raggiungere un nodo della rete anonima. Si può assumere che ogni router della rete ha una posizione geografica diversa. Questo messaggio è inviato periodicamente perché si possono presentare guasti nei link o qualche destinazione può non essere più raggiungibile.

Codice 4 : É un messaggio che contiene la tabella di routing di ogni nodo della rete, cioè il next hop per ognuno di essi. Il corpo può essere quindi l'indirizzo IP di un nodo interno o direttamente la destinazione, questo secondo caso è inviato solo a un nodo. Si invia quando il thread del SGR che calcola i percorsi minimi ha finito o quando il SGR riceve una richiesta esplicita attraverso un codice 7. Questo secondo caso si presenta quando un messaggio con codice 3 non ha raggiunto un nodo della rete o quando un messaggio con codice 6 è arrivato in ritardo cancellando la tabella di routing di un nodo.

Codice 5 : Indica al SGR che il processo di invio anonimo è terminato con successo e che può reinizializzare le sue strutture interne. Questo tipo di messaggio è inviato dall'ultimo hop della rete. Questo tipo di messaggio, insieme a quello con codice 6, se non raggiungono le rispettive destinazioni non creano grandi problemi perché la successiva richiesta di un cliente sovrascrive le strutture delle entità implicate.

Codice 6 : É un messaggio inviato dal SGR a tutti i router della rete anonima eccetto all'ultimo salto, ovvero quello che invia il messaggio con codice 5. Si esclude un router per risparmiare tempo, per non sovraccaricare le code di uscita del SGR e per non incrementare il traffico che circola nella rete. Questo tipo di messaggio, insieme a quello con codice 5, se non raggiungono le rispettive destinazioni non creano grandi problemi perché la successiva richiesta di un cliente sovrascrive le strutture delle entità implicate.

Codice 7 : Indica una richiesta esplicita della tabella di routing da parte di un nodo della rete. É un caso eccezionale che può presentarsi perché si utilizza il protocollo UDP. I casi implicati sono la perdita di un pacchetto con codice 4 o un ritardo di un pacchetto con codice 6 che elimina la nuova tabella di routing. Un nodo della rete invia questo tipo di pacchetto dopo aver ricevuto il messaggio contenente i dati da un altro nodo, cioè il pacchetto anonimo sta viaggiando nella rete.

Codice	**Emittente**	**Ricevente**	**Corpo**
1	Origine	SGR	Indirizzo Destinazione
2	SGR	Origine	Indirizzo first hop
3	Router	SGR	Stati link + zona geografica
4	SGR	Router	Tabella di routing
5	Router	SGR	-
6	SGR	Rete*	-
7	Router	SGR	-

Tabella 3.1: Sintesi del protocollo

3.5 Fasi di funzionamento

In questa sezione si descrive la sequenza temporale delle azioni che ogni entità svolge per permettere l'invio di un messaggio. Il lettore verrà aiutato con frammenti di codice commentato.

Per capire meglio la sequenza temporale che coinvolge tutte le entità, si decide di dividere in tre fasi ben differenziate quello che succede, con lo scopo di controllare, gestire e tracciare gli eventi. Il processo consta di:

- Fase 1: Attivazione del sistema.
- Fase 2: Stabilimento delle comunicazioni.
- Fase 3: Trasferimento dei dati.

L'applicazione è costituita da processi intercomunicanti e sincronizzati da scambi di messaggi, il che obbliga a applicare un ordine rigido di apparizione delle entità. Questo vuol dire che ogni entità deve agire nel momento esatto in cui si richiede dentro le fasi citate precedentemente, assumendo un'importanza diversa l'uno sull'altro in ciascun caso.

Così facendo, coincidendo con l'attivazione del sistema, il SGR sarà il primo programma da avviare, a questo seguiranno tutti i nodi finali che sono le destinazioni. Successivamente, apparirà la rete anonima di nodi per stabilire i link e per ultimo i nodi origine che saranno pronti a dare supporto all'utente nel trasferimento dei dati.

3.5.1 Fase 1: Attivazione del sistema

Come appena detto, il SGR è la prima entità a fare la sua apparizione, il cui compito è preparare il sistema, cioè inizializzare le strutture necessarie e lanciare i thread che svolgeranno le attività particolari.
Quando il SGR si avvia crea molte strutture interne quali un semaforo, una mappa delle zone geografiche per ogni nodo e una mappa di nodi della rete anonima prima vuota e che poi riempirà con il passare del tempo. Inoltre, crea il socket UDP su una porta conosciuta a tutte le entità e crea la chiave per cifrare/decifrare i messaggi UDP che rispettivamente invia e riceve da altre entità.

```
public class SGR {
   private Object semaforoOD;          //Shared between source and dijkstra
   private Map<Integer,Nodo> red;         //Network
   private int puerto_sgr;             //Port SGR
```

```
    static DatagramSocket socket;        //Socket SGR
    private int destino;              //Final destination
    private int destinoInterno;             //Internal destination
    private Map<Integer,Integer> destinos; //<destination,last hop node>

    static SecretKey key;
    final static int GCM_IV_LENGTH = 12;
    final static int GCM_TAG_LENGTH = 16;

    public SGR() {
        semaforoOD = new Object();
        red = new HashMap<Integer,Nodo>();
        destino = -1;
        destinoInterno = -1;
        destinos = new HashMap<Integer,Integer>();
        puerto_sgr = 10000;
        try {
            socket = new DatagramSocket(puerto_sgr);
            //Key creation to encipher and decipher
            ...
        } catch (Exception e) {
            System.out.println("ERROR SGR:\t" + e.toString());
            System.exit(0);
        }
    }
}
```

Terminato questo passo di inizializzazione, lancia un thread per il calcolo delle route ottime che sarà spiegato successivamente.
L'ultima azione del processo principale è rimanere sospeso fino a ricevere qualunque messaggio dalle altre entità, questo mediante una chiamata bloccante che si ripete periodicamente nel socket creato precedentemente. Appena arriva un messaggio, il processo lancia un thread, che elaborerà il messaggio, in modo da tornare il più rapidamente possibile ad attendere un altro.

```
//ROUTES MANAGEMENT SYSTEM
Thread t_dijkstra = new Thread(new SGR_DIJKSTRA(sgr));
t_dijkstra.start();//thread to calculate router
try {
    while(true) {
        try{
            byte[] recvBuf = new byte[1024];
```

```
            DatagramPacket recvPacket = new DatagramPacket(recvBuf,recvBuf.
                length);

            System.out.println("SGR:\t\tWAITING MESSAGES");
            SGR.socket.receive(recvPacket);
            (new Thread(new SGR_SEND(sgr, recvPacket))).start();
        }catch(Exception e){
            System.out.println("ERROR SGR:\t" + e.toString());
        }
    }
}catch(Exception e){
    System.out.println("ERROR SGR:\t" + e.toString());
}finally {
    SGR.socket.close();
    System.out.println("SGR:\tFINISHED");
}
```

Quando il SGR riceve un messaggio, si crea un thread che lo processa. Il primo passo è decifrare il messaggio per vedere che codice contiene secondo il protocollo menzionato precedentemente. In questa fase si può ricevere un solo codice:

Codice 3 : Il messaggio è stato inviato da un nodo della rete anonima e contiene lo stato dei suoi link e la zona geografica con le destinazioni che può raggiungere. É un messaggio inviato periodicamente. Si crea o aggiorna il nodo nella struttura interna del SGR. A causa dell'origine che non ha ancora inviato una richiesta, il SGR come da progetto non fa nient'altro, cioè non si realizza il calcolo delle route a ogni aggiornamento fino a che non riceve una richiesta.

```
public boolean updateRed(String sendMensaje, String nodo_ip, int
    nodo_puerto) throws Exception{
    if(!red.containsKey(nodo_puerto))
        red.put(nodo_puerto, new Nodo(nodo_ip, nodo_puerto));

    List<String> mensaje = Arrays.asList(sendMensaje.split("/"));
    Map<Integer,Edge> edges = new HashMap<Integer, Edge>();
    for(String s : mensaje.get(0).split(","))//node's neighbors
        edges.put(Integer.parseInt(s.split(":")[0]),new Edge(Integer.parseInt
            (s.split(":")[0]),Boolean.parseBoolean(s.split(":")[1])));
    if(mensaje.size() > 1)//node's destinations
        for(String s : mensaje.get(1).split(","))
            destinos.put(Integer.parseInt(s),nodo_puerto);
```

```
    return red.get(nodo_puerto).setAdjacencies(edges);
}

public boolean setAdjacencies(Map<Integer,Edge> adjacencies) {
    boolean cambio = false;
    if(this.adjacencies.size() != 0) {
        List<Boolean> estados_viejos = this.adjacencies.values()
                .stream().sorted().map(Edge::getActive)
                .collect(Collectors.toList());
        List<Boolean> estados_nuevos = adjacencies.values()
                .stream().sorted().map(Edge::getActive)
                .collect(Collectors.toList());
        cambio = estados_viejos.equals(estados_nuevos);
        if(!cambio) {//if changes occur
            for(Edge e : adjacencies.values()) {
                if(!this.adjacencies.containsKey(e.getTarget()))//new link
                    this.adjacencies.put(e.getTarget(),
                            new Edge(e.getTarget(),e.getActive()));
                this.adjacencies.get(e.getTarget()).setActive(e.getActive());//
                    update links state
            }
            System.out.println("SGR_THREAD_NODE_" + puerto + ":\tCHANGE STATE
                OF LINK");
        }
    }else this.adjacencies = adjacencies;//first time
    return cambio;
}
```

La seguente entità che deve apparire è il nodo finale che funge da destinatario. Le destinazioni devono solo incaricarsi di ricevere informazioni. A differenza del SGR, il protocollo di livello di trasporto che utilizza per connettersi con la rete anonima è il TCP. Per questo, implementa socket orientati a connessioni che richiedono lo stabilimento della stessa, le richieste di attesa, le conferme durante lo scambio di dati e la liberazione del canale. Inoltre, si combina con il protocollo di sicurezza dello strato intermedio SSL che fornisce comunicazioni riservate e ammette certificati digitali firmati da un'Autorità di certificazione, per l'autenticazione delle entità. Questo protocollo permette la negoziazione tra le parti degli algoritmi crittografici che useranno.

Il procedimento affinché il dispositivo finale stabilisca una connessione SSL come server con qualunque nodo interno della rete è il seguente:

1. Leggere l'informazione del suo deposito, accedendo a questo con una chiave

essendo protetto, e ottiene il suo certificato digitale firmato. D'altra parte, il suo certificato digitale firmato è inviato al nodo affinché questo lo autentichi. Il certificato dell'entrata di affidabilità serve per verificare l'identità del nodo trasmittente, perché la esige, confrontando il certificato che invia con la copia che ha nel suo deposito.

2. Con tutto questo si prepara il contesto della comunicazione, si apre un socket TCP/SSL e si effettua l'operazione bloccante in attesa di connessioni. Nel caso in cui qualcuno degli estremi non fornisca un certificato valido, si annullerà il processo e si darà per finita la comunicazione.

```
public class Destino implements Runnable {

   public Destino(int puerto) {
      //Reading certificate
      ...
   }
   ...
   while(true) {
      try{
         System.out.println("DESTINATION_" + puerto + ":\tWAITING
             CONNECTION");
         SSLSocket sslsocket = (SSLSocket)sslserversocket.accept();
         System.out.println("DESTINATION_" + puerto + ":\tCONECTED");

         //Processing message
         ...

         sslsocket.close();
      }catch(Exception e){
         System.out.println(e.toString());
      }
   }
}
```

Le ultime entità che appaiono nell'attivazione del sistema sono i nodi della rete anonima. All'inizio quello che fanno è conoscere le condizioni della rete per adeguare le loro strutture e risorse alla realtà presentata e sapere i dati dei nodi intermedi e delle destinazioni che si dovranno connettere. Il passo successivo di ogni nodo interno consiste nel creare tre thread che sono:

- Thread con funzione uguale a un nodo destinazione, cioè gestirà i depositi e sarà un server nelle comunicazioni TCP/SSL che creerà con altri router interni e con i nodi finali.
- Thread che rimane sospeso fino a ricevere messaggi dal SGR mediante una chiamata bloccante in un socket UDP che si ripete periodicamente.
- Thread che invia periodicamente messaggi UDP cifrati al SGR per avvisarlo delle destinazioni che può raggiungere e degli stati dei suoi link.

```
//ROUTER'S THREAD TO SSL CONNECTION
public class Nodo_SSL implements Runnable{
   public Nodo_SSL(Nodo n){
      //Reading certificate
   }
   ...
   while(true) {
      try {
         //getting connections
         System.out.println("NODE_" + n.getPuerto() + "_THREAD_SSL:\
            tWAITING CONNECTION ");
         SSLSocket sslsocket_entrada = (SSLSocket)sslserversocket.accept();
         System.out.println("NODE_" + n.getPuerto() + "_THREAD_SSL:\
            tCONNECTED");

         //processing message
         ...
      }catch(Exception e){
         System.out.println("ERROR NODE_" + n.getPuerto() + "_THREAD_SSL:\t
            " + e.toString());
      }
   }
}
```

```
//ROUTER'S THREAD TO RECEIVE MESSAGE UDP FROM SGR
public class Nodo_Recv implements Runnable {
   ...
   while(true) {//getting from SGR
      try{
         byte[] recvBuf = new byte[1024];
         DatagramPacket recvPacket = new DatagramPacket(recvBuf,recvBuf.
            length);
```

```
            System.out.println("NODE_" + n.getPuerto() + "_THREAD_RECV:\
                tWAITING SGR");
            n.getSocketUDP().receive(recvPacket);
            String recvMensaje = n.decrypt(recvPacket.getData());
            List<String> mensaje = Arrays.asList(recvMensaje.split("-"));
            if(Integer.parseInt(mensaje.get(0)) == 4) //partial route
               n.setNextHop(mensaje.get(1));
            else  n.setNextHop(",-1");          //message delivered
            System.out.println("NODE_" + n.getPuerto() + "_THREAD_RECV:\
                tMESSAGE RECEIVED:\t" + n.getNH_Puerto());
        }catch(Exception e){
            System.out.println("ERROR NODE_" + n.getPuerto() + "_THREAD_RECV:\
                t" + e.toString());
        }
    }
}
```

```
//ROUTER'S THREAD TO SEND MESSAGE UDP TO SGR
System.out.println("NODO_" + n.getPuerto() + "_HILO_SEND");
while(true) {//sending to SGR
    try{
        String sendMensaje = "3-";
        sendMensaje = sendMensaje + n.getAdjacencies().values()
                                  .stream()
                                  .map(Edge::toString)
                                  .collect( Collectors.joining( ","));
        sendMensaje = sendMensaje + n.getTabla().values()
                                  .stream()
                                  .map(i -> i.toString())
                                  .collect( Collectors.joining( ",","/",""))
                                      ;
        byte[] sendBuf = n.encrypt(sendMensaje);
        DatagramPacket sendPacket = new DatagramPacket(sendBuf,sendBuf.length
            ,Nodo.SGR_direccion,Nodo.SGR_puerto);
        n.getSocketUDP().send(sendPacket);
        System.out.println("NODE_" + n.getPuerto() + "_THREAD_SEND:\tMESSAGE
            SENT:\t" + sendMensaje);
        Thread.sleep(20000);
    }catch(Exception e){
        System.out.println("ERROR NODE_" + n.getPuerto() + "_THREAD_SEND:\t"
            + e.toString());
    }
}
```

Alla fine di questa fase ci sono i nodi interni che inviano le loro informazioni di stato al SGR affinché operi con loro. In questo momento non esiste connessione tra loro, sono stati trasmessi solo messaggi UDP e stanno aspettando che il SGR gli consegni la tabella di route dopo che un nodo origine si connette alla rete. D'altra parte, il SGR procede a registrare le informazioni che ha ricevuto e rimane in attesa che un nodo origine comunichi con esso.

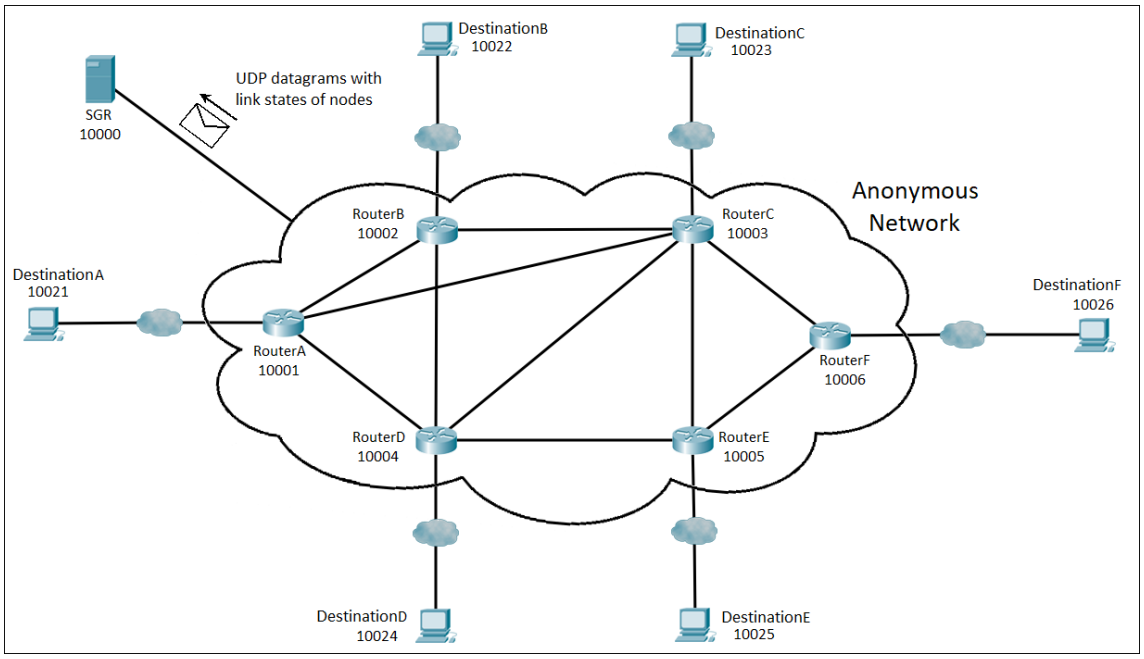

Figura 3.3: Rappresentazione del sistema alla fine della fase 1

3.5.2 Fase 2: Stabilimento delle comunicazioni

In questa fase di esecuzione, il sistema si trova in attesa che un utente interagisca con l'applicazione affinché il nodo origine si attivi. Nel mentre i nodi della rete anonima stanno inviando messaggi al SGR per aggiornare le sue informazioni riguardo la rete.

Quando un utente vuole inviare un messaggio anonimo, apre l'applicazione e crea tutte le strutture necessarie, cioè un socket UDP e la chiave per cifrare/decifrare i messaggi UDP inviati e ricevuti dal SGR. L'applicazione attende che l'utente inserisca la destinazione alla quale vuole inviare il messaggio anonimo. Dopo aggiunge la destinazione in un pacchetto con codice 1 del protocollo citato precedentemente per indicare che è una richiesta. Il pacchetto è cifrato, aggiunto in un pacchetto UDP con destinazione il SGR e inviato. L'applicazione rimane sospesa fino a che riceve

il first hop dal SGR. C'è un tempo limite di attesa della risposta, dopo il quale si riavvia l'applicazione.

```
public String toSGR() throws Exception {
   //Sending request + destination to SGR
   BufferedReader teclado = new BufferedReader(new InputStreamReader(System
       .in));
   System.out.print("Address IP(port) destination:\t" );
   String  mensaje = teclado.readLine();
   byte[] sendBuffer = new byte[1024];
   sendBuffer = encrypt("1-" + mensaje);
   packet = new DatagramPacket(sendBuffer,sendBuffer.length,SGR_direccion,
       SGR_puerto);
   socketUDP.send(packet);
   System.out.println("SOURCE_UDP_" + socketUDP.getLocalPort() + ":\MESSAGE
        SENT:\t" + mensaje);

   //Getting first hop
   byte[] recvBuffer = new byte[1024];
   packet = new DatagramPacket(recvBuffer,recvBuffer.length);
   System.out.println("SOURCE_UDP_" + socketUDP.getLocalPort() + ":\
       tWAITING FIRST HOP");
   socketUDP.receive(packet);
   String fh  = Arrays.asList((new String(decrypt(packet.getData()))).trim
       ().split("-")).get(1);
   System.out.println("SOURCE" + socketUDP.getLocalPort() + ":\tFIRST HOP:\
       t" + fh);

   socketUDP.close();
   return fh;
}
```

Il SGR riceve il pacchetto dall'origine, lo decifra e vede che ha codice 1, cioè è una richiesta, e contiene la destinazione al quale vuole inviare un messaggio. Internamente si configura la destinazione e l'ultimo salto della rete anonima. Dopo, attraverso una notify sul semaforo, si sveglia il thread di calcolo delle route e rimane sospeso fino a che lo stesso non lo sveglia.

Del SGR rimane solo da commentare un ultimo thread, questo di ambito interno, i cui scopi sono: generare casualmente i costi dei link, eseguire l'algoritmo di routing a partire dalle informazioni ricevute dalla rete anonima e inviare le route ottime ai nodi. Questo thread rimane sospeso fino a che è svegliato dal thread che processa i pacchetti ricevuti da un'origine. Il primo compito che svolge è generare casualmente

i costi per tutti i link del sistema attuale. In particolare, i costi sono simmetrici. Dopo crea un numero di thread uguale al numero di nodi nella rete attuale. Ogni thread calcola la sua tabella di routing utilizzando un algoritmo che è una modifica di quello dei cammini minimi di Dijkstra e la invia attraverso il codice 4 al suo nodo corrispondente della rete. Il messaggio, come nei casi precedenti, è cifrato e inviato attraverso UDP. Alla fine, si sveglia il thread che invia il first hop all'origine e si torna ad attendere un'altra richiesta attraverso una wait.

```
System.out.println("SGR_THREAD_DIJKSTRA");
while(true) {
   try {
      System.out.println("SGR_THREAD_DIJKSTRA:\tESPERANDO ORIGEN");
      synchronized(sgr.getSemaforoOD()){        //waiting source
         sgr.getSemaforoOD().wait();
      }

      //creation casual costs
      sgr.getRed().values().stream()
         .map(Nodo::getAdjacencies)             //Stream<Map>
         .map(Map::values)                    //Stream<Collection<Edge>>
         .flatMap(Collection::stream)           //Stream<Edge>
         .forEach(e -> e.setWeight(costes.get(random.nextInt(costes.size())
            )));

      //symmetric costs links
      for(Nodo n : sgr.getRed().values())
         for(Edge e : n.getAdjacencies().values())
            sgr.getRed().get(e.getTarget()).getAdjacencies()
               .get(n.getPuerto()).setWeight(e.getWeight());

      //creation threads to calculate and send tables
      for(Nodo n : sgr.getRed().values()) {
         try {
            (new Thread(new Runnable() {//Thread to receive
                  @Override
                  public void run() {
                     try { SGR_SEND.toNodo_Ruta(sgr, n.getDireccion(), n.
                        getPuerto());
                  } catch (Exception e) { System.out.println("ERROR
                     SGR_THREAD_DIJKSTRA:\t" + e.toString());   }
               }})).start();
            Thread.sleep(20);
         } catch (InterruptedException e) {
```

```
            System.out.println("ERROR SGR_THREAD_DIJKSTRA:\t" + e.toString
                ());
         }
      }
      System.out.println("SGR_THREAD_DIJKSTRA:\tCALCULATION TABLES DONE");

      synchronized(sgr.getSemaforoOD()) { //wake up source
            sgr.getSemaforoOD().notify();
      }
   }catch(Exception e) {
      System.out.println("ERROR SGR_THREAD_DIJKSTRA:\t" + e.toString());
   }
}
```

```
public static void toNodo_Ruta(SGR sgr, String nodo_ip, int nodo_puerto)
    throws Exception {
   for(Nodo n1 : sgr.getRed().values())
      n1.setSD(nodo_puerto);//reinitialize structures
   sgr.getRed().get(nodo_puerto).computePaths(sgr.getRed());//calculate
       route
   sgr.getRed().get(nodo_puerto).setTabla(sgr.getRed());//calculate table

   //add next hop in the packet
   String sendMensaje;
   if(sgr.getDestinos().containsKey(sgr.getDestino())) {//destination out
      if(sgr.getDestinos().get(sgr.getDestino()) == sgr.getRed().get(
          nodo_puerto).getPuerto())
         sendMensaje = "4-127.0.0.1," + sgr.getDestino();//sending final
             destination
      else  sendMensaje = "4-127.0.0.1," + sgr.getRed()
            .get(nodo_puerto).getTabla().get(sgr.getDestinoInterno()) + "";
               //sending next hop
   }else sendMensaje = "4-127.0.0.1," + sgr.getRed().get(nodo_puerto)
         .getTabla().get(sgr.getDestino()) + "";//destination out
   sendTo(sendMensaje,nodo_ip,nodo_puerto);
}
```

Ogni nodo della rete anonima quando riceve il messaggio, lo decifra e dato che il codice è 4, aggiorna la sua tabella di routing.

Per ultimo, si sceglie casualmente un nodo che sarà il first hop dell'origine. Si inserisce l'indirizzo del nodo scelto in un messaggio del protocollo creato con codice 2, si costruisce il pacchetto che sarà cifrato e si invia all'origine. Si invia tre volte

nel caso un messaggio si perda. L'origine riceverà solo un pacchetto e scarterà gli altri.

```
public void toOrigen_FH(int destino) throws Exception {
      sgr.setDestino(destino);
      if(sgr.getDestinos().containsKey(destino))//destination out
         sgr.setDestinoInterno(sgr.getDestinos().get(destino));
      else  sgr.setDestinoInterno(destino);//internal destination

      System.out.println("SGR:\tDESTINATION:\t\t" + sgr.getDestino());
      System.out.println("SGR:\tINTERNAL DESTINATION:\t" + sgr.
          getDestinoInterno());

      synchronized(sgr.getSemaforoOD()) { //wake up thread_dijkstra
            sgr.getSemaforoOD().notify();
      }
         synchronized(sgr.getSemaforoOD()){  //waiting thread_dijkstra
         try {
            sgr.getSemaforoOD().wait();
         } catch (InterruptedException e) {
            e.printStackTrace();
         }
      }
         Random random = new Random();
         Nodo fh = sgr.getRed().get(sgr.getRed().keySet().stream()
               .collect(Collectors.toList())
               .get(random.nextInt(sgr.getRed().keySet().size())));
         sendMensaje = "2-" + fh.getDireccion() + "," + fh.getPuerto();
         for(int i=0;i<3;i++) sendTo(sendMensaje,nodo_ip,nodo_puerto);
   }
```

L'origne riceve il messaggio, lo decifra, salva l'indirizzo del first hop e attende l'input dell'utente.

Da qui in poi, per ogni messaggio con codice 3 che riceve il SGR, si verifica se ci sono o meno aggiornamenti dei link del nodo. Se gli stati dei link sono cambiati, cioè si è presentato un guasto, si ricalcola e invia la route parziale rispetto al nodo in questione. I problemi che possono presentarsi sono: route precedenti perse, link guasti o il SGR non ha ricevuto il messaggio di consegna avvenuta. Si inserisce la route in un messaggio del protocollo creato con codice 4, si costruisce il pacchetto che sarà cifrato e si invia al nodo. Non si fa un invio ripetuto perché in caso di perdita del pacchetto, il nodo invia un altro messaggio con codice 7 per richiedere la route.

Il sistema è pronto per trasmettere il messaggio dall'origine alla destinazione attraverso la rete come si può vedere nella figura 3.4.

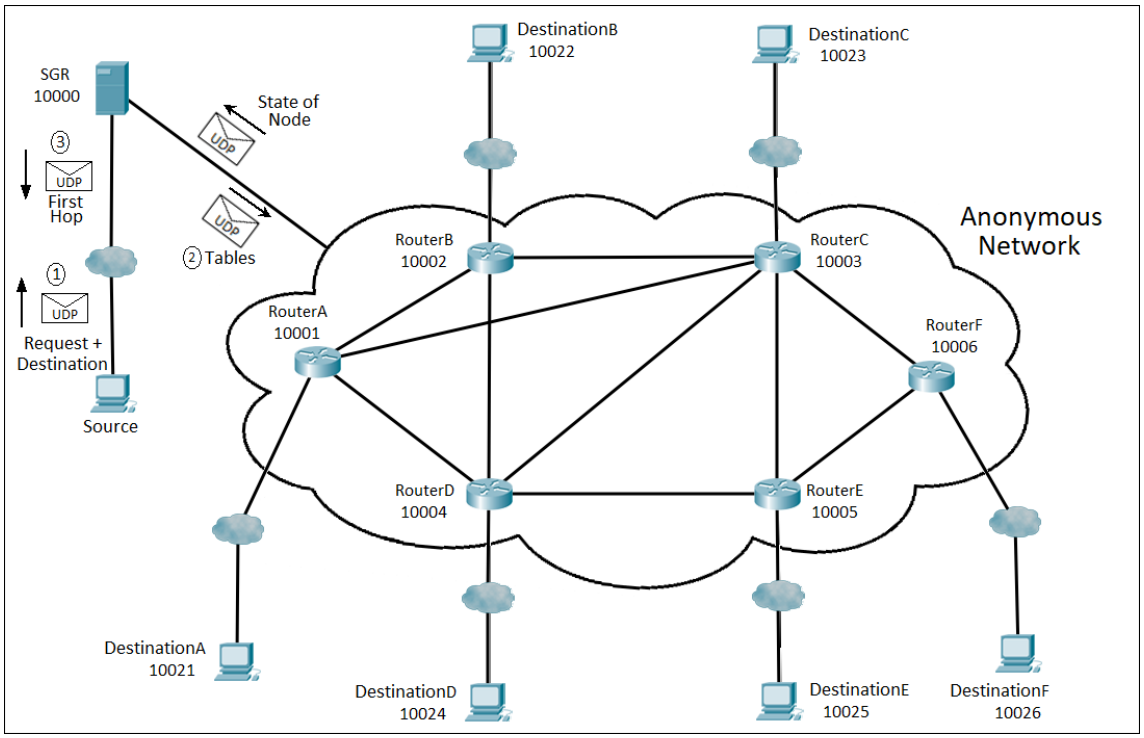

Figura 3.4: Rappresentazione del sistema durante la fase 2

3.5.3 Fase 3: Trasferimento di dati

Ci si avvicina alla parte finale del funzionamento del sistema nel quale tutti gli elementi sono interconnessi e informati su ciò che accade attorno a loro. I dispositivi finali e la rete di nodi interni sono pronti a trasmettere l'informazione in maniera sicura. Dispongono dei mezzi necessari affinché quel flusso di dati si instradi correttamente in termini di affidabilità e sicurezza.

L'applicazione è a disposizione dell'utente, cioè è in attesa che l'utente scriva il messaggio che vuole inviare anonimamente. Quando il messaggio è pronto, si caricano i depositi per svolgere la funzione di client. I depositi comprendono la chiave privata e il suo certificato firmato da un'Autorità di certificazione. Il messaggio è inviato al first hop attraverso il protocollo SSL e l'algoritmo crittografico scelto.

```
public void toFH(String fh) throws IOException {
   //Reading certificate and connection to first hop
   List<String> first_hop = Arrays.asList(fh.split(","));
```

```
    System.setProperty("javax.net.ssl.trustStore","AlmacenTrust");
    System.setProperty("javax.net.ssl.keyStore","Almacen");
    System.setProperty("javax.net.ssl.keyStorePassword", "oooooo");
    SSLSocketFactory sslsocketfactory = (SSLSocketFactory) SSLSocketFactory.
        getDefault();
    SSLSocket sslsocket = (SSLSocket) sslsocketfactory.createSocket(
        first_hop.get(0),Integer.parseInt(first_hop.get(1)));

    //Writing message
    BufferedReader teclado = new BufferedReader(new InputStreamReader(System
        .in));
    System.out.print("Write Message:\t" );
    String mensaje = teclado.readLine();

    //Sending message
    BufferedWriter bufferedwriter = new BufferedWriter(new
        OutputStreamWriter(sslsocket.getOutputStream()));
    bufferedwriter.write(mensaje + '\n');
    bufferedwriter.flush();
    System.out.println("SOURCE_SSL:\tMESSAGE SENT:\t" + mensaje);

    sslsocket.close();
}
```

Quando riceve dati, il thread del nodo interno iniziale si sveglierà, facendo uso della tabella analizzerà se questi sono per esso e deve salvarli o se deve ritrasmetterli a un altro nodo o se deve inviarli alle sue destinazioni già. Questo si ripete con il next hop nel percorso costruito fino ad arrivare alla destinazione.
Se il messaggio è per esso o per una delle sue destinazioni, lo stesso thread invia un messaggio al SGR con codice 5 in un pacchetto UDP cifrato per avvisarlo che il messaggio anonimo è stato consegnato.

```
public void toNodo_Fin() {
    if(sgr.getDestino() != -1) {//If there is not new request
        if(nodo_puerto == sgr.getDestinoInterno()) {//corrispondence request
            - delivered message
                System.out.println("SGR_:\tMESSAGE DELIVERED TO:\t" + sgr.
                    getDestino());
                int excluido = nodo_puerto;
                sgr.setDestino(-1);
                sgr.setDestinoInterno(-1);
                sendMensaje = "6-DELIVERED";
                sgr.getRed().values().stream()
```

```
                .filter(n -> n.getPuerto() != excluido)
                .forEach(n -> {
                try { sendTo(sendMensaje,n.getDireccion(),n.getPuerto());
                } catch (Exception e) { System.out.println("ERROR
                    SGR_THREAD_SEND:\t" + e.toString()); }

                });
            System.out.println("SGR_THREAD_SEND:\tMESSAGE SENT TO NETWORK:\
                t" + sendMensaje);
        }
    }
}
```

D'altra parte, il SGR lo riceve e capisce che il messaggio è stato inviato dal nodo della rete anonima che era l'ultimo salto del percorso. Se non ci sono nuove richieste e c'è corrispondenza tra il nodo in memoria e l'ultimo salto del pacchetto, si reinizializzano le strutture interne. Inoltre, si invia un messaggio agli altri nodi della rete per notificare la consegna avvenuta e la terminazione del processo. Questo si aggiunge in un messaggio del protocollo creato con codice 6, si costruisce il pacchetto che sarà cifrato e si invia al nodo. Dato che i messaggi che invia il SGR sono pacchetti UDP, c'è la possibilità che non siano ricevuti ma non è un problema perché nel progetto sono stati implementati meccanismi di rinvio periodo o su richiesta.

```
while(true) {
    try {
        //getting message
        ...
        //sending message
        SSLSocketFactory sslsocketfactory = (SSLSocketFactory)
            SSLSocketFactory.getDefault();
        SSLSocket sslsocket_salida;

        //if not arrive the table => request table
        //if message to close arrives late => request table
        do {
            if(n.getNH_Puerto()==-1){
                byte[] sendBuf = n.encrypt("7-Table");
                DatagramPacket packet = new DatagramPacket(sendBuf,sendBuf.
                    length,Nodo.SGR_direccion,Nodo.SGR_puerto);
                n.getSocketUDP().send(packet);//best-effort
                System.out.println("NODE_" + n.getPuerto() + "_THREAD_SSL:\
                    tREQUEST TABLE");
```

```
        }
    }while(n.getNH_Puerto()==-1);
    boolean entrega = false;
    if(n.getNH_Puerto() != n.getPuerto()) {
        //send to next hop/destination
        if(n.getTabla().containsKey(n.getNH_Puerto())) { //destination out
            sslsocket_salida = (SSLSocket) sslsocketfactory.createSocket("
                127.0.0.1",n.getTabla().get(n.getNH_Puerto()));
            entrega = true;
        }else{//internal destination
            //break link => request table
            do{
                if(!n.getAdjacencies().get(n.getNH_Puerto()).getActive()){
                    byte[] sendBuf = n.encrypt("7-Table");
                    DatagramPacket packet = new DatagramPacket(sendBuf,
                        sendBuf.length,Nodo.SGR_direccion,Nodo.SGR_puerto);
                    n.getSocketUDP().send(packet);//best-effort
                    System.out.println("NODE_" + n.getPuerto() + "_THREAD_SSL
                        :\tREQUEST TABLE");
                }
            }while(!n.getAdjacencies().get(n.getNH_Puerto()).getActive());

            sslsocket_salida = (SSLSocket) sslsocketfactory.createSocket("
                127.0.0.1",n.getNH_Puerto());
        }
        //sending to next hop
        BufferedWriter bufferedwriter = new BufferedWriter(new
            OutputStreamWriter(sslsocket_salida.getOutputStream()));

        bufferedwriter.write(mensaje + '\n');
        bufferedwriter.flush();
        System.out.println("NODE_" + n.getPuerto() + "_THREAD_SSL:\
            tMESSAGE SENT TO " + sslsocket_salida.getPort() + ":\t" +
            mensaje);

        sslsocket_salida.close();
    }else{//message to node
        System.out.println("NODE_" + n.getPuerto() + "_THREAD_SSL:\
            tMESSAGE RECEIVED:\t" + mensaje);
        entrega = true;
    }
    if(entrega) {
        byte[] sendBuf = n.encrypt("5-DELIVERED");
```

```
            DatagramPacket packet = new DatagramPacket(sendBuf,sendBuf.length,
                Nodo.SGR_direccion,Nodo.SGR_puerto);
            n.getSocketUDP().send(packet);//best-effort
            System.out.println("NODE_" + n.getPuerto() + "_THREAD_SSL:\
                tMESSAGE SENT TO SGR:\t5-DELIVERED");
            n.setNextHop(",-1");
        }
    }catch(Exception e){
        System.out.println("ERROR NODE_" + n.getPuerto() + "_THREAD_SSL:\t" +
             e.toString());
    }
}
```

Può presentarsi il caso in cui la tabella di routing non arrivi a un nodo e quando cerca di ritrasmettere un pacchetto a un altro nodo del percorso, non può perché la tabella è vuota. Un altro caso può essere che il link per il next hop sia rotto. In entrambi i casi, il nodo della rete anonima invia una richiesta eccezionale al SGR nella quale chiede la tabella. Per fare questo si genera un messaggio con codice 7 del protocollo creato e si invia con UDP.
Quando il SGR riceve un pacchetto che contiene il codice 7, capisce che il messaggio è stato inviato da un nodo qualunque della rete anonima in caso eccezionale perché si è presentato un problema. Come nel caso 3 i problemi che possono presentarsi sono: perdita di route precedenti, link rotti o il SGR che non ha ricevuto il messaggio di consegna avvenuta. Si calcola la route rispetto al nodo che ha individuato il problema, si inserisce la tabella in un messaggio del protocollo creato con codice 4, si costruisce il pacchetto che sarà cifrato e lo si invia al nodo. Non si fa un invio ripetuto perché in caso di perdita del pacchetto, il nodo invia un altro messaggio con codice 7 per richiedere la route.

Fin qui arriva la descrizione dei compiti che svolgono tutte le entità, i quali sono molto dettagliati. In particolare, il programma è ottimizzato rispetto al numero di iterazioni che si eseguono nei cicli di inserimento dei dati nelle strutture. Considerando che il costo dei link tra i nodi è lo stesso per entrambi i versi, la complessità delle operazioni quali completare la tabella dei costi, calcolare le diverse route e riempire la tabella di route ottime, si riduce alla metà.

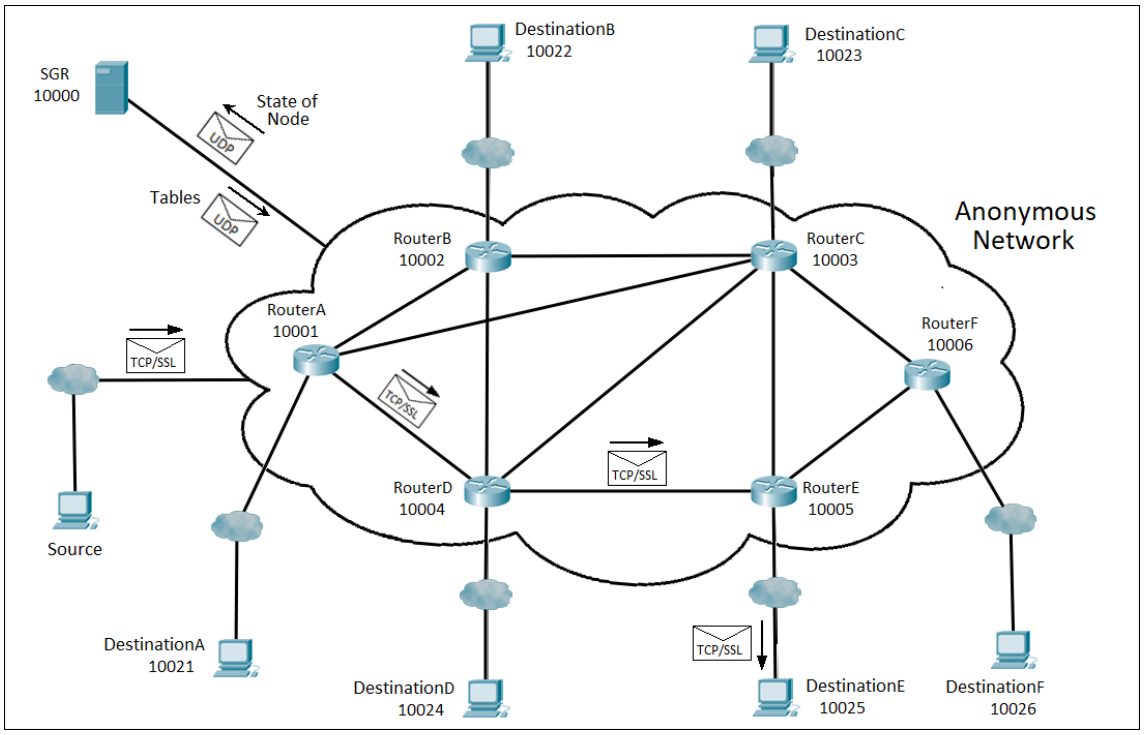

Figura 3.5: Rappresentazione del sistema durante la fase 3

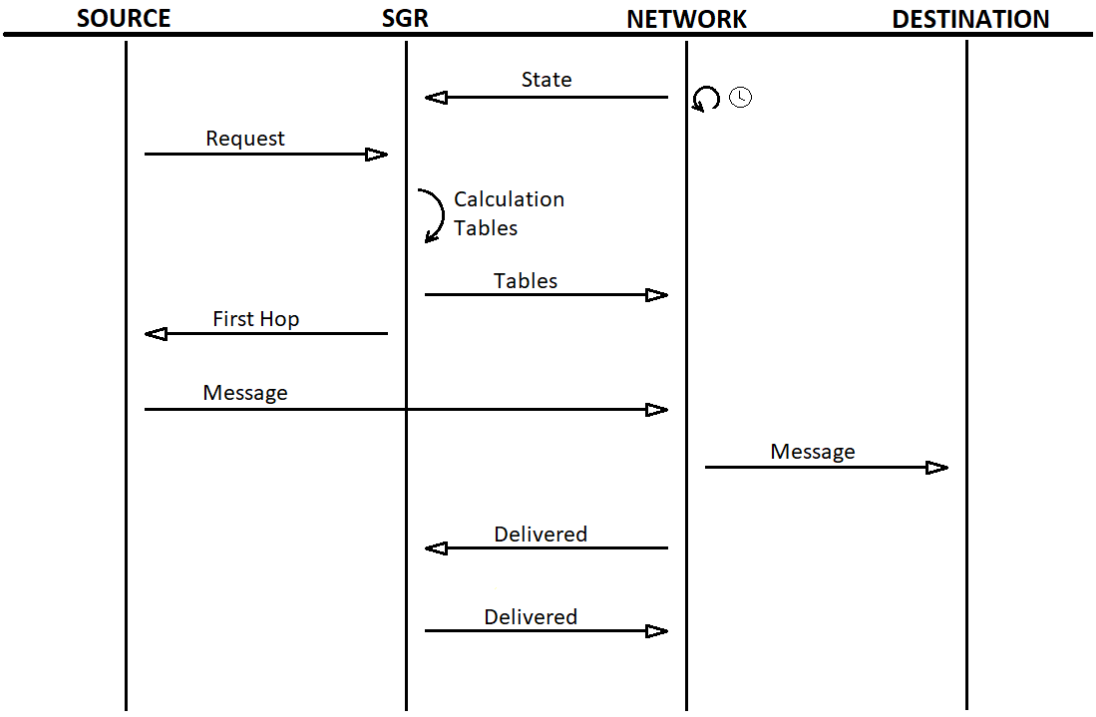

Figura 3.6: Diagramma temporale

Capitolo 4

Risultati

In questo capitolo si confermano o confutano le teorie sulle quali si è basato lo sviluppo del progetto e si verifica se sono stati raggiunti i compromessi pattuiti con il cliente in un primo momento. É cruciale verificare e valutare la qualità del prodotto costruito in modo che si individuino il prima possibile gli errori per minimizzare l'impatto negativo che potrebbe implicare la correzione di questi.

Dopo la fase di implementazione si passa a quella di test, un processo tecnico che richiede professionisti altamente capaci nei linguaggi di programmazione, metodi, tecniche di test e tool specializzati. Con tutto questo, si elabora un parere nel quale si espongono tutte le conclusioni estratte, questioni che serviranno per conoscere in profondità i punti forti e deboli dell'applicazione e così da poterla migliorare in aggiornamenti futuri.

Le informazioni che si descrivono in seguito corrispondono alla risposta ottenuta dal sistema in un ambiente con un nodo origine, sei nodi intermedi e le destinazioni associate. Pertanto, ogni punto tratta più di un aspetto, includendo situazioni anomale, e non si deve associare con un'unica e semplice esecuzione. In ognuno di essi si vedrà si vedrà come lavora ogni entità, riflettendo la coesione esistente e mostrando la reazione di alcune allo stimolo delle altre.

Si precisa che le prove sono state fatte in un solo computer, cioè sono stati simulati la rete anonima, il SGR e gli estremi attraverso processi che creano thread. Per questo, dato che non si poteva uscire dalla scheda di rete, nei risultati seguenti sono state utilizzate le porte del livello di trasporto come indirizzi IP. Inoltre, nei codici presentati precedentemente molte funzioni richiedono un indirizzo IP e porta, per esempio la "send", quindi la soluzione adottata è stata porre come indirizzo IP "127.0.0.1" che è l'indirizzo di localhost.

4.1 Attivazione del sistema

Il primo passo è dato dal SGR, che seguendo le linee fissate, esegue le sue istruzioni, lancia un thread interno che riunirà tutte le informazioni raccolte per calcolare le rotte ottime e per ultimo rimane in attesa passiva fino a che si attiva la rete di nodi interni che invierà il suo stato.

```
-----------------------------------------------------------------
|                    ROUTES MANAGEMENT SYSTEM                   |
-----------------------------------------------------------------
=================================================================
=================================================================
|SGR:   Internal structures createds                            |
=================================================================
|SGR:   Socket UDP port 10000 created                           |
=================================================================
|SGR:   Key to encipher/decipher UDP messages created           |
=================================================================
|SGR_THREAD_DIJKSTRA                                            |
=================================================================
|SGR_THREAD_DIJKSTRA:  Waiting request from users               |
=================================================================
|SGR:   Waiting messages                                        |
=================================================================
```

Al SGR seguono tutte le destinazioni, che in questo progetto sono sei come i router della rete anonima. Le destinazioni caricheranno i certificati e svolgeranno la funzione di server per attendere gli invii di segmenti TCP/SSL, estremo a estremo attraverso i nodi interni, dall'origine.

```
=================================================================
|DESTINATION_10021:  Certificates loaded                        |
=================================================================
|DESTINATION_10021:  Waiting connection                         |
=================================================================
|DESTINATION_10022:  Certificados cargados                      |
=================================================================
|DESTINATION_10022:  Waiting connection                         |
=================================================================
|DESTINATION_10023:  Certificados cargados                      |
```

```
=================================================================
|DESTINATION_10023:  Waiting connection                          |
=================================================================
|DESTINATION_10024:  Certificados cargados                       |
=================================================================
|DESTINATION_10024:  Waiting connection                          |
=================================================================
|DESTINATION_10025:  Certificados cargados                       |
=================================================================
|DESTINATION_10025:  Waiting connection                          |
=================================================================
|DESTINATION_10026:  Certificados cargados                       |
=================================================================
|DESTINATION_10026:  Waiting connection                          |
=================================================================
```

Le entità che finiscono l'inizializzazione del sistema sono i nodi della rete anonima. Questi caricheranno i certificati e creeranno tre thread dei quali due sono per inviare e ricevere messaggi UDP dal SGR e l'ultimo per trasmettere il messaggio dell'origine alla destinazione o ad un altro nodo su una connessione SSL. Riassumendo, un thread rimarrà in attesa passiva fino a che il SGR invierà la tabella di routing, l'altro thread aspetta che il nodo origine si connetta e invii dati.

```
-----------------------------------------------------------------
|                         ANONYMOUS NETWORK                     |
-----------------------------------------------------------------
=================================================================
=================================================================
|NODE_10001_THREAD_SSL                                          |
=================================================================
|NODE_10001_THREAD_SSL:   Certificates loaded                    |
=================================================================
|NODE_10001_THREAD_SSL:   Waiting connection                     |
=================================================================
|NODE_10001_THREAD_RECV                                         |
=================================================================
|NODE_10001_THREAD_RECV:  Waiting message from SGR               |
=================================================================
|NODE_10001_THREAD_SEND                                         |
=================================================================
|NODE_10001_THREAD_SEND:  MESSAGE SENT:                         |
|3-10002:true,10003:true,10004:true/10021                       |
```

```
================================================================
|NODE_10002_THREAD_SSL                                         |
================================================================
|NODE_10002_THREAD_SSL:   Certificates loaded                  |
================================================================
|NODE_10002_THREAD_SSL:   Waiting connection                   |
================================================================
|NODE_10002_THREAD_RECV                                        |
================================================================
|NODE_10002_THREAD_RECV:  Waiting message from SGR             |
================================================================
|NODE_10002_THREAD_SEND                                        |
================================================================
|NODE_10002_THREAD_SEND:  MESSAGE SENT:                        |
|3-10001:true,10003:true,10004:true/10022                      |
================================================================
|NODE_10003_THREAD_SSL                                         |
================================================================
|NODE_10003_THREAD_SSL:   Certificates loaded                  |
================================================================
|NODE_10003_THREAD_SSL:   Waiting connection                   |
================================================================
|NODE_10003_THREAD_RECV                                        |
================================================================
|NODE_10003_THREAD_RECV:  Waiting message from SGR             |
================================================================
|NODE_10003_THREAD_SEND                                        |
================================================================
|NODE_10003_THREAD_SEND:  MESSAGE SENT:                        |
|3-10001:true,10002:true,10004:true,10005:true,10006:true/10023|
================================================================
|NODE_10004_THREAD_SSL                                         |
================================================================
|NODE_10004_THREAD_SSL:   Certificates loaded                  |
================================================================
|NODE_10004_THREAD_SSL:   Waiting connection                   |
================================================================
|NODE_10004_THREAD_RECV                                        |
================================================================
|NODE_10004_THREAD_RECV:  Waiting message from SGR             |
================================================================
|NODE_10004_THREAD_SEND                                        |
================================================================
|NODE_10004_THREAD_SEND:  MESSAGE SENT:                        |
```

```
|3-10001:true,10002:true,10003:true,10005:true/10024                |
=====================================================================
|NODE_10005_THREAD_SSL                                              |
=====================================================================
|NODE_10005_THREAD_SSL:   Certificates loaded                       |
=====================================================================
|NODE_10005_THREAD_SSL:   Waiting connection                        |
=====================================================================
|NODE_10005_THREAD_RECV                                             |
=====================================================================
|NODE_10005_THREAD_RECV:  Waiting message from SGR                  |
=====================================================================
|NODE_10005_THREAD_SEND                                             |
=====================================================================
|NODE_10005_THREAD_SEND:  MESSAGE SENT:                             |
|3-10003:true,10004:true,10006:true/10025                           |
=====================================================================
|NODE_10006_THREAD_SSL                                              |
=====================================================================
|NODE_10006_THREAD_SSL:   Certificates loaded                       |
=====================================================================
|NODE_10006_THREAD_SSL:   Waiting connection                        |
=====================================================================
|NODE_10006_THREAD_RECV                                             |
=====================================================================
|NODE_10006_THREAD_RECV:  Waiting message from SGR                  |
=====================================================================
|NODE_10006_THREAD_SEND                                             |
=====================================================================
|NODE_10006_THREAD_SEND:  MESSAGE SENT:                             |
|3-10003:true,10005:true/10026                                      |
=====================================================================
```

D'altra parte, il SGR riceve i messaggi UDP, li decifra, analizza il codice e dato, che è uguale a 3, aggiorna la sua struttura interna che rappresenta la rete anonima.

```
=====================================================================
|SGR:  MESSAGE RECEIVED FROM 10001:                                 |
|3-10002:true,10003:true,10004:true/10021                           |
=====================================================================
|SGR:   Network updated                                             |
=====================================================================
|SGR:   Waiting messages                                            |
```

```
=====================================================================
|SGR:  MESSAGE RECEIVED FROM 10002:                                  |
|3-10001:true,10003:true,10004:true/10022                            |
=====================================================================
|SGR:   Network updated                                              |
=====================================================================
|SGR:   Waiting messages                                             |
=====================================================================
|SGR:  MESSAGE RECEIVED FROM 10003:                                  |
|3-10001:true,10002:true,10004:true,10005:true,10006:true/10023      |
=====================================================================
|SGR:   Network updated                                              |
=====================================================================
|SGR:   Waiting messages                                             |
=====================================================================
|SGR:  MESSAGE RECEIVED FROM 10004:                                  |
|3-10001:true,10002:true,10003:true,10005:true/10024                 |
=====================================================================
|SGR:   Network updated                                              |
=====================================================================
|SGR:   Waiting messages                                             |
=====================================================================
|SGR:  MESSAGE RECEIVED FROM 10005:                                  |
|3-10003:true,10004:true,10006:true/10025                            |
=====================================================================
|SGR:   Network updated                                              |
=====================================================================
|SGR:   Waiting messages                                             |
=====================================================================
|SGR:  MESSAGE RECEIVED FROM 10006:                                  |
|3-10003:true,10005:true/10026                                       |
=====================================================================
|SGR:   Network updated                                              |
=====================================================================
|SGR:   Waiting messages                                             |
=====================================================================
```

4.2 Stabilimento delle comunicazioni

Con la prova precedente si mostra il corretto funzionamento delle entità implicate nella fase 1. Si vede che esiste una comunicazione fluida tra i nodi della rete anonima e il SGR e che non ci sono problemi nella messa a punto del sistema. Partendo dai dati già raccolti si dimostra che la fase 2 si porta a termine con successo. Tutti i processi attivi sono in stato di blocco fino a che il nodo origine verrà eseguito.

Quando un utente lancia l'applicazione per trasmettere traffico anonimo, invia una richiesta al SGR con la destinazione. Il SGR risponde con un nodo scelto casualmente che sarà il first hop per l'utente.

```
----------------------------------------------------------------------
|                          SOURCE - USER                             |
----------------------------------------------------------------------
======================================================================
======================================================================
|SOURCE:   Socket UDP port 55916 created                             |
======================================================================
|SOURCE:   Key to encipher/decipher UDP messages created             |
======================================================================
|SOURCE_UDP_55916:  Destination IP(port) Address:       10026        |
======================================================================
|SOURCE_UDP_55916:  MESSAGE SENT:  10026                             |
======================================================================
|SOURCE_UDP_55916:  WAITING FIRST HOP                                |
======================================================================
|SOURCE_UDP_55916:  FIRST HOP:    10001                              |
======================================================================
```

Prima di inviare il first hop, il thread "Dijkstra" del SGR, come detto in precedenza, genera le tabelle dei costi e esegue i calcoli opportuni i cui risultati sono le route ottime tra i nodi interni per arrivare alla destinazione scelta dall'origine.

```
======================================================================
|SGR:  MESSAGE RECEIVED FROM 55916:                                  |
|1-10026                                                             |
======================================================================
|SGR:  DESTINATION:          10026                                   |
|SGR:  INTERNAL DESTINATION:  10006                                  |
```

```
|=====================================================================|
|SGR:   Waiting messages                                              |
|=====================================================================|
|SGR_HILO_DIJKSTRA:  CALCULATING TABLES                               |
|=====================================================================|
|Routes of 10001:                                                     |
|10001 -> 10001 : 0.0    Path: [10001]                                |
|10001 -> 10002 : 6.0    Path: [10001, 10002]                         |
|10001 -> 10003 : 6.0    Path: [10001, 10003]                         |
|10001 -> 10004 : 4.0    Path: [10001, 10004]                         |
|10001 -> 10005 : 14.0   Path: [10001, 10004, 10005]                  |
|10001 -> 10006 : 15.0   Path: [10001, 10003, 10006]                  |
|                                                                     |
|Table of 10001:  <Destination> -> <Next Hop>                         |
|10001 -> 10001                                                       |
|10002 -> 10002                                                       |
|10003 -> 10003                                                       |
|10004 -> 10004                                                       |
|10005 -> 10004                                                       |
|10006 -> 10003                                                       |
|=====================================================================|
|Routes of 10002:                                                     |
|10002 -> 10001 : 6.0    Path: [10002, 10001]                         |
|10002 -> 10002 : 0.0    Path: [10002]                                |
|10002 -> 10003 : 1.0    Path: [10002, 10003]                         |
|10002 -> 10004 : 7.0    Path: [10002, 10004]                         |
|10002 -> 10005 : 9.0    Path: [10002, 10003, 10005]                  |
|10002 -> 10006 : 10.0   Path: [10002, 10003, 10006]                  |
|                                                                     |
|Table of 10002:  <Destination> -> <Next Hop>                         |
|10001 -> 10001                                                       |
|10002 -> 10002                                                       |
|10003 -> 10003                                                       |
|10004 -> 10004                                                       |
|10005 -> 10003                                                       |
|10006 -> 10003                                                       |
|=====================================================================|
|Routes of 10003:                                                     |
|10003 -> 10001 : 6.0   Path: [10003, 10001]                          |
|10003 -> 10002 : 1.0   Path: [10003, 10002]                          |
|10003 -> 10003 : 0.0   Path: [10003]                                 |
|10003 -> 10004 : 8.0   Path: [10003, 10002, 10004]                   |
|10003 -> 10005 : 8.0   Path: [10003, 10005]                          |
|10003 -> 10006 : 9.0   Path: [10003, 10006]                          |
```

```
|                                                                    |
|Table of 10003:  <Destination> -> <Next Hop>                        |
|10001 -> 10001                                                      |
|10002 -> 10002                                                      |
|10003 -> 10003                                                      |
|10004 -> 10002                                                      |
|10005 -> 10005                                                      |
|10006 -> 10006                                                      |
======================================================================
|Routes of 10004:                                                    |
|10004 -> 10001 : 4.0   Path: [10004, 10001]                         |
|10004 -> 10002 : 7.0   Path: [10004, 10002]                         |
|10004 -> 10003 : 8.0   Path: [10004, 10002, 10003]                  |
|10004 -> 10004 : 0.0   Path: [10004]                                |
|10004 -> 10005 : 10.0  Path: [10004, 10005]                         |
|10004 -> 10006 : 17.0  Path: [10004, 10002, 10003, 10006]           |
|                                                                    |
|Table of 10004:  <Destination> -> <Next Hop>                        |
|10001 -> 10001                                                      |
|10002 -> 10002                                                      |
|10003 -> 10002                                                      |
|10004 -> 10004                                                      |
|10005 -> 10005                                                      |
|10006 -> 10002                                                      |
======================================================================
|Routes of 10005:                                                    |
|10005 -> 10001 : 14.0  Path: [10005, 10003, 10001]                  |
|10005 -> 10002 : 9.0   Path: [10005, 10003, 10002]                  |
|10005 -> 10003 : 8.0   Path: [10005, 10003]                         |
|10005 -> 10004 : 10.0  Path: [10005, 10004]                         |
|10005 -> 10005 : 0.0   Path: [10005]                                |
|10005 -> 10006 : 7.0   Path: [10005, 10006]                         |
|                                                                    |
|Table of 10005:  <Destination> -> <Next Hop>                        |
|10001 -> 10003                                                      |
|10002 -> 10003                                                      |
|10003 -> 10003                                                      |
|10004 -> 10004                                                      |
|10005 -> 10005                                                      |
|10006 -> 10006                                                      |
======================================================================
|Routes of 10006:                                                    |
|10006 -> 10001 : 15.0  Path: [10006, 10003, 10001]                  |
|10006 -> 10002 : 10.0  Path: [10006, 10003, 10002]                  |
```

```
|10006 -> 10003 : 9.0   Path: [10006, 10003]                          |
|10006 -> 10004 : 17.0  Path: [10006, 10005, 10004]                   |
|10006 -> 10005 : 7.0   Path: [10006, 10005]                          |
|10006 -> 10006 : 0.0   Path: [10006]                                 |
|                                                                     |
|Table of 10006:  <Destination> -> <Next Hop>                         |
|10001 -> 10003                                                       |
|10002 -> 10003                                                       |
|10003 -> 10003                                                       |
|10004 -> 10005                                                       |
|10005 -> 10005                                                       |
|10006 -> 10006                                                       |
=======================================================================
|SGR_HILO_DIJKSTRA:  Calculation tables done                          |
=======================================================================
|SGR_HILO_DIJKSTRA:  Waiting request fro users                        |
=======================================================================
```

Dopo invia a ogni nodo della rete anonima la sua tabella di routing e all'utente il first hop scelto.

```
=======================================================================
|SGR_THREAD_SEND:  MESSAGE SENT TO 10001:  4-10003                    |
=======================================================================
|SGR_THREAD_SEND:  MESSAGE SENT TO 10002:  4-10003                    |
=======================================================================
|SGR_THREAD_SEND:  MESSAGE SENT TO 10003:  4-10006                    |
=======================================================================
|SGR_THREAD_SEND:  MESSAGE SENT TO 10004:  4-10002                    |
=======================================================================
|SGR_THREAD_SEND:  MESSAGE SENT TO 10005:  4-10006                    |
=======================================================================
|SGR_THREAD_SEND:  MESSAGE SENT TO 10006:  4-10026                    |
=======================================================================
|SGR_THREAD_SEND:  MESSAGE SENT TO 55916:  2-127.0.0.1,10001          |
|SGR_THREAD_SEND:  MESSAGE SENT TO 55916:  2-127.0.0.1,10001          |
|SGR_THREAD_SEND:  MESSAGE SENT TO 55916:  2-127.0.0.1,10001          |
=======================================================================
```

Inoltre, ogni nodo interno riceverà la sua tabella di routing ottima parziale che consisterà nel next hop verso la destinazione scelta.

```
=====================================================================
|NODE_10001_THREAD_RECV:  MESSAGE RECEIVED:       10003             |
=====================================================================
|NODE_10001_THREAD_RECV:  WAITING SGR                               |
=====================================================================
|NODE_10002_THREAD_RECV:  MESSAGE RECEIVED:       10003             |
=====================================================================
|NODE_10002_THREAD_RECV:  WAITING SGR                               |
=====================================================================
|NODE_10003_THREAD_RECV:  MESSAGE RECEIVED:       10006             |
=====================================================================
|NODE_10003_THREAD_RECV:  WAITING SGR                               |
=====================================================================
|NODE_10004_THREAD_RECV:  MESSAGE RECEIVED:       10002             |
=====================================================================
|NODE_10004_THREAD_RECV:  WAITING SGR                               |
=====================================================================
|NODE_10005_THREAD_RECV:  MESSAGE RECEIVED:       10006             |
=====================================================================
|NODE_10005_THREAD_RECV:  WAITING SGR                               |
=====================================================================
|NODE_10006_THREAD_RECV:  MESSAGE RECEIVED:       10026             |
=====================================================================
|NODE_10006_THREAD_RECV:  WAITING SGR                               |
=====================================================================
```

INTERNAL NODE SOURCE	**INTERNAL DESTINATION**					
	A	**B**	**C**	**D**	**E**	**F**
A	0	6	6	4	-	-
B	6	0	1	7	-	-
C	6	1	0	-	8	9
D	4	7	-	0	10	-
E	-	-	8	10	0	7
F	-	-	9	-	7	0

Tabella 4.1: Esempio di tabella dei costi dei link tra i nodi

INTERNAL NODE SOURCE	INTERNAL DESTINATION					
	A	**B**	**C**	**D**	**E**	**F**
A	0	6	6	4	14	15
B	6	0	1	7	9	10
C	6	1	0	8	8	9
D	4	7	8	0	10	17
E	14	9	8	10	0	7
F	15	10	9	17	7	0

Tabella 4.2: Esempio di tabella dei costi minimi tra i nodi

INTERNAL NODE SOURCE	INTERNAL DESTINATION					
	A	**B**	**C**	**D**	**E**	**F**
A	-	B	C	D	D	C
B	A	-	C	D	C	C
C	A	B	-	B	E	F
D	A	B	B	-	E	B
E	C	C	C	D	-	F
F	C	C	C	E	E	-

Tabella 4.3: Esempio di tabella di route ottime tra i nodi

Come si può apprezzare in entrambe le tabelle, le caselle assumono valori nulli quando coincide un nodo interno con se stesso. D'altra parte, prendendo come riferimento la tabella dei costi, si può assicurare che: se per andare da un nodo interno origine a uno destino si seguono le indicazioni della seconda tabella indicante il next hop da prendere, la route tracciata è migliore tra tutte le possibili alternative.

L'interpretazione grafica della tabella dei costi mostrata precedentemente si può analizzare nella figura 4.1. A partire da questa si calcolano tutte le possibili route e si scelgono quelle il cui costo è minimo.

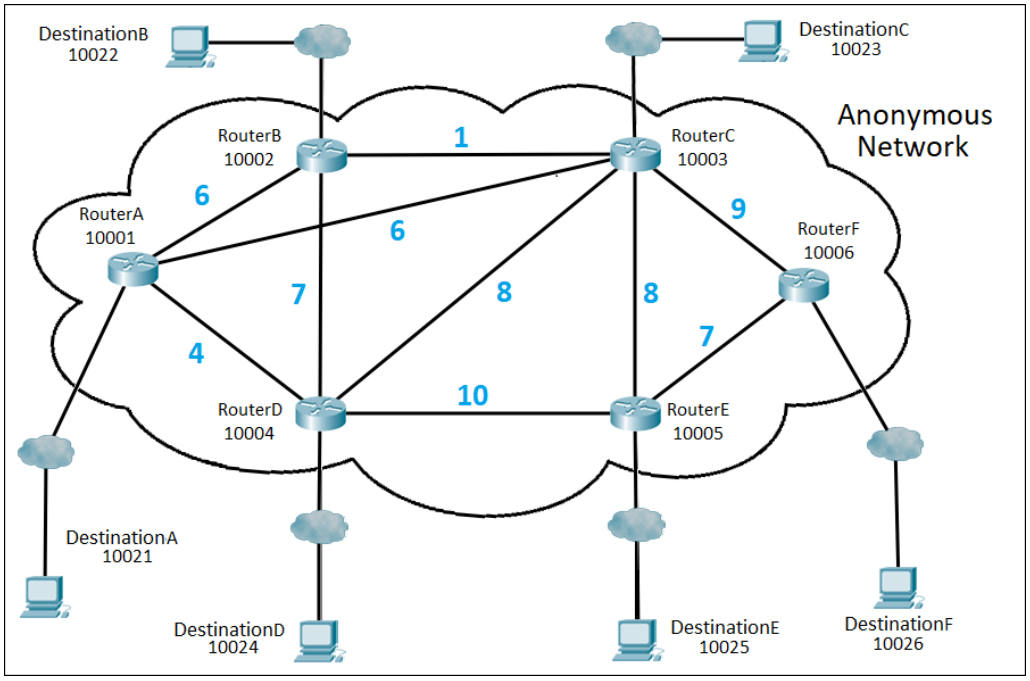

Figura 4.1: Interpretazione grafica della tabella dei costi

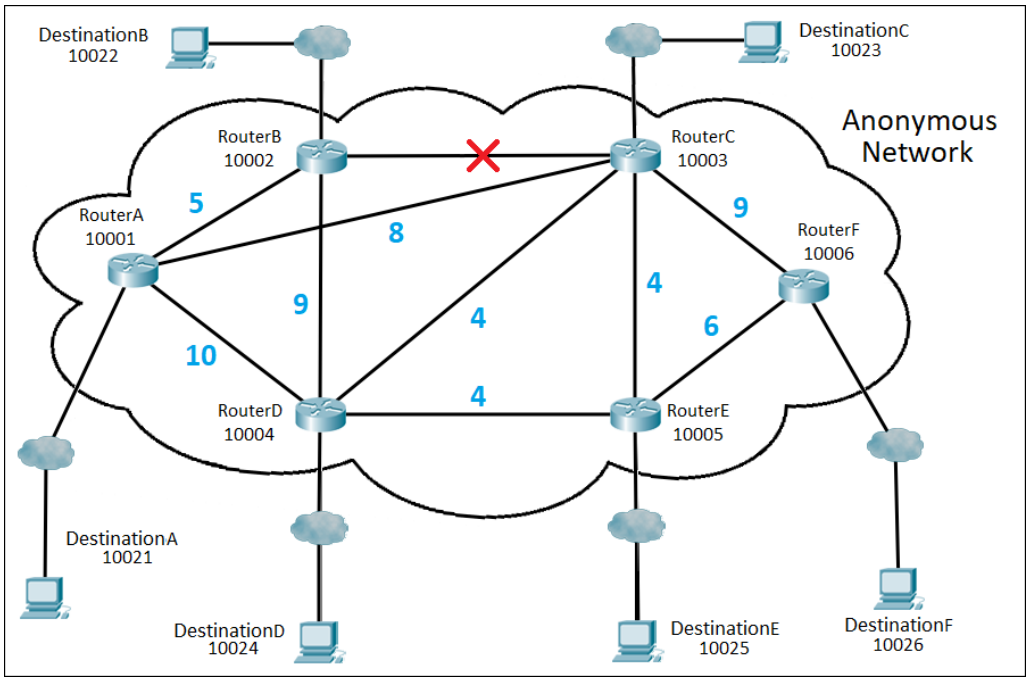

Figura 4.2: Secondo esempio di costi aleatori

4.3 Trasferimento dei dati

Tutto il sistema è pronto per permettere un invio sicuro e anonimo da un'origine a una destinazione. L'ultimo passo da effettuare è trasmettere i dati che un usuario immette da tastiera. Il client stabilisce la connessione SSL con il first hop inviato dal SGR. Come primo passo cliente e first hop si scambiano i certificati in quanto è stata richiesta l'autenticazione del client. Normalmente solo l'entità con funzione di server deve inviare al client il suo certificato in una connessione SSL.

```
=================================================================
|SOURCE_SSL:  Connection opened                                  |
=================================================================
|Write Message:  Test SSL message Final Project Anonynous System |
=================================================================
|SOURCE_SSL:  Message Sent:                                      |
|Test SSL message Final Project Anonynous System                 |
=================================================================
|SOURCE:  FINISHED                                               |
=================================================================
```

Il pacchetto viaggia per la rete nella quale per ogni tratta, i nodi verificano i certificati e dopo inviano il messaggio. Quando arriva all'ultimo nodo della rete, questo verifica il certificato della destinazione e trasmette il pacchetto. Per terminare il processo, questo nodo invia un pacchetto UDP al SGR per notificare la consegna del messaggio.

```
=================================================================
|NODE_10001_THREAD_SSL:  Connected                               |
=================================================================
|NODE_10001_THREAD_SSL:  Message received:                       |
|Test SSL message Final Project Anonynous System                 |
=================================================================
|NODE_10003_THREAD_SSL:  Connected                               |
=================================================================
|NODE_10001_THREAD_SSL:  Message sent to 10003:                  |
|Test SSL message Final Project Anonynous System                 |
=================================================================
|NODE_10003_THREAD_SSL:  Message received:                       |
|Test SSL message Final Project Anonynous System                 |
=================================================================
```

```
|NODE_10001_THREAD_SSL:  Waiting connection                          |
======================================================================
|NODE_10006_THREAD_SSL:  Connected                                   |
======================================================================
|NODE_10003_THREAD_SSL:  Message sent to 10006:                      |
|Test SSL message Final Project Anonynous System                     |
======================================================================
|NODE_10006_THREAD_SSL:  Message received:                           |
|Test SSL message Final Project Anonynous System                     |
======================================================================
|NODE_10003_THREAD_SSL:  Waiting connection                          |
======================================================================
|NODE_10006_THREAD_SSL:  Message sent to 10026:                      |
|Test SSL message Final Project Anonynous System                     |
======================================================================
|NODE_10003_THREAD_SSL:  Waiting connection                          |
======================================================================
|NODE_10006_THREAD_SSL:  Message sent to SGR:  5-DELIVERED           |
======================================================================
|NODE_10006_THREAD_SSL:  Waiting connection                          |
======================================================================
```

La destinazione riceve il pacchetto e verifica che l'emittente è l'ultimo nodo della rete anonima, riuscendo di fatto a occultare l'emittente originale. Dopo torna ad attendere un'altra connessione.

```
======================================================================
|DESTINATION_10026:  Connected                                       |
======================================================================
|DESTINATION_10026:  Message received:                               |
|Test SSL message Final Project Anonynous System                     |
======================================================================
|DESTINATION_10026:  Waiting connection                              |
======================================================================
```

Il SGR riceve il pacchetto, lo decifra e vede che la destinazione ha ricevuto i dati, di seguito notifica i restanti nodi della rete anonima dell'avvenuto e reinizializza le strutture.

```
======================================================================
|SGR_THREAD_SEND:  MESSAGE RECEIVED FROM 10006:                      |
```

```
|5-DELIVERED                                                          |
======================================================================
|SGR:  MESSAGE DELIVERED TO:  10026                                   |
======================================================================
|SGR_THREAD_SEND:  MESSAGE SENT TO 10001:  6-DELIVERED                |
|SGR_THREAD_SEND:  MESSAGE SENT TO 10002:  6-DELIVERED                |
|SGR_THREAD_SEND:  MESSAGE SENT TO 10003:  6-DELIVERED                |
|SGR_THREAD_SEND:  MESSAGE SENT TO 10004:  6-DELIVERED                |
|SGR_THREAD_SEND:  MESSAGE SENT TO 10005:  6-DELIVERED                |
|SGR_THREAD_SEND:  MESSAGE SENT TO NETWORK:  6-DELIVERED              |
======================================================================
```

In seguito, i restanti nodi della rete ricevono la conferma di consegna inviata dal SGR.

```
======================================================================
|NODE_10001_THREAD_RECV:  Message received:  Message delivered      |
======================================================================
|NODE_10001_THREAD_RECV:  Waiting SGR                               |
======================================================================
|NODE_10002_THREAD_RECV:  Message received:  Message delivered      |
======================================================================
|NODE_10002_THREAD_RECV:  Waiting SGR                               |
======================================================================
|NODE_10003_THREAD_RECV:  Message received:  Message delivered      |
======================================================================
|NODE_10003_THREAD_RECV:  Waiting SGR                               |
======================================================================
|NODE_10004_THREAD_RECV:  Message received:  Message delivered      |
======================================================================
|NODE_10004_THREAD_RECV:  Waiting SGR                               |
======================================================================
|NODE_10005_THREAD_RECV:  Message received:  Message delivered      |
======================================================================
|NODE_10005_THREAD_RECV:  Waiting SGR                               |
======================================================================
```

Capitolo 5

Conclusioni

L'obiettivo di questo capitolo è plasmare, in alcune linee, l'insieme di conclusioni che sono state estratte quando si sviluppava il progetto per identificare i punti forti dello stesso e apprezzare gli aspetti negativi che richiedono una miglioria. Dopo aver elaborato un piano di prove per l'applicazione sviluppata, aver verificato il rendimento della stessa con diverse configurazioni e il suo corretto funzionamento, si è soddisfatto uno degli obiettivi principali del progetto: l'implementazione di un Sistema anonimo. Utilizzando tutto ciò che è stato detto precedentemente, si è verificato l'efficacia e l'efficienza dell'algoritmo di routing e dei meccanismi di sicurezza per controllare il trasferimento dei dati per la rete, soddisfacendo tutti gli obiettivi che si proposero all'inizio del lavoro. Per questo possiamo affermare che sono state coperte di forma soddisfacente le aspettative che si erano poste, perciò successivamente si affrontano le conclusioni ottenute.

Per capire in modo più chiaro le conclusioni alle quali si è arrivato dopo l'implementazione del progetto, si elencano:

- Si è realizzato un Sistema Anonimo il quale è dotato di meccanismi di sicurezza. Questi meccanismi consistono nella protezione con procedimenti di autenticazione e tecniche di cifratura di tutti i dati che si trasmettono per la red provenienti dalle diverse entità, ottenendo un canale di comunicazione sicuro.

- Per lo stabilimento delle comunicazioni tra dispositivi che formano il sistema sono stati adottati diversi protocolli di trasporto secondo le necessità, che sono mutevoli. In alcuni casi si ha lavorato con socket TCP perché era imprescindibile riservare risorse per dare un servizio in linea, mentre in altre sono

stati implementati socket UDP dove prevalevano la fluidità dei messaggi e la rapidità di gestione delle operazioni.

- L'accesso alla rete TCP/SSL è ristretto a dispositivi che presentano certificati digitali validi. Questi file sono depositi che comprendono informazioni di identificazione che assicurano l'autenticazione e la riservatezza dei proprietari, sempre e quando sono firmati da un'autorità di certificazione. I segmenti TCP sono protetti da uno strato di sicurezza SSL che applica metodi crittografici per evitare la sottrazione di dati e assicurare l'integrità degli stessi.

- Riguardo i datagrammi UDP, la cifratura si realizza attraverso l'algoritmo simmetrico AES, il quale utilizza per cifrare e decifrare la stessa chiave, condivisa tra le entità partecipanti in una comunicazione sicura.

- É stato incorporato un algoritmo di routing che calcola le rotte più efficienti secondo le condizioni dell'ambiente, le distribuisce e le aggiorna nel corso del tempo. Questo permette di ampliare notevolmente la rete senza preoccuparsi del fatto che le prestazioni del sistema siano ridotte dall'aumento del numero di dispositivi connessi.

- Per garantire la non rintracciabilità dei pacchetti che viaggiano per la rete, per ogni richiesta dell'usuario o attraverso un temporizzatore, si cambiano i costi dei link tra i router. Di conseguenza per ogni esecuzione dell'algoritmo di instradamento, i cammini sono differenti. Questo si applica anche al caso in cui lo stesso cliente invia diversi dati allo stesso destinatario. Inoltre, le rotte non sono memorizzate nel SGR per prevenire un furto di dati da parte di un attaccante esterno.

- I pacchetti inviati a qualunque destinazione includono come sorgente IP l'indirizzo dell'ultimo nodo della rete anonima che ha maneggiato il pacchetto. In questo modo le informazioni riguardo l'emittente originale si perdono quasi totalmente. É stata utilizzata la parola "quasi" perché se un avversario può attuare un attacco passivo globale, cioè ottenere una visione globale della rete, può estrarre corrispondenze tra le entrate e le uscite dei nodi e tracciare il cammino di un pacchetto. Inoltre, si ricorda che la rete anonima realizzata nel progetto si deve intendere come una rete globale, quindi attuare un attacco passivo globale significherebbe controllare tutti o la maggior parte dei router della rete mondiale, il che è impossibile. Un insieme di tecniche di Analisi

di Traffico sono state sviluppate negli anni per tracciare il flusso continuo di traffico che viaggia per la rete di bassa latenza come TOR. In questi studi è stato dimostrato che questo tipo di attacchi sono molto difficili da contrastare, a meno che si utilizzino tecniche che implicherebbero latenze elevate o che richiedano l'iniezione di grandi quantità di traffico coperto; entrambe rappresentano soluzioni molto costose.

- É stato realizzato un Sistema anonimo per ottenere un livello di sicurezza che si possa raggiungere in un sistema altamente utilizzabile e molto economico.

- Il sistema registra tutte le operazioni delle entità ed è capace di individuare falli, notificare e prendere una decisione (chiudere una sessione, ristabilire la comunicazione o reinizializzare l'applicazione)

- Grazie alla tecnologia Java con la quale si ha sviluppato integralmente il progetto, l'applicazione è dotata di un gran rendimento e della caratteristica di multi-piattaforma. Sono state utilizzate le tecniche più moderne per ridurre il numero di iterazioni che svolge il programma, grazie per esempio agli stream di Java 8. Riguardo all'essere multi-piattaforma, non è necessario riscrivere le applicazioni sviluppate se si vuole cambiare piattaforma di esecuzione. Questo è stato verificato con successo per questa applicazione in scenari distinti, garantendo la portabilità dell'applicazione sviluppata su diverse piattaforme.

Capitolo 6

Linee Future

In quest'ultimo capitolo si lasciano alcune possibili vie future di investigazione e sviluppo che si potrebbero realizzare per completare o ampliare il sistema proposto in questo lavoro con lo scopo di incrementare efficacia e efficienza.

- Sviluppare un'interfaccia utente che permetta di scegliere preferenze d'uso sulla rete e offra diverse opzioni sui servizi disponibili. Per esempio, lasciar scegliere all'utente il grado di anonimato o la necessità di velocità, informandolo sul fatto che sono inversamente proporzionali. Una soluzione pratica potrebbe essere implementare code nei router della rete anonima in modo simile alle code minimamente utilizzate nel traffico di qualità delle reti. Si produrranno gli stessi problemi che si hanno nella rete classica, però in una rete anonima la quantità di traffico è minore attualmente, quindi non si genereranno colli di bottiglia.

- Sviluppare un'interfaccia grafica per l'usuario che, oltre alle opzioni precedenti, includa informazioni di stato, grafici di uso e consumo e simulazioni di carico di lavoro. Quando si lavora con obiettivi grafici, si necessita di persone molto preparate per garantire che il risultato sia il più possibile "user friendly", cioè sia molto gradevole e facile da usare per gli utenti. Se un cliente non si trova comodo con un'applicazione per circostanze tali che non è intuitiva o il suo progetto è poco attrattivo o non risponde alla celerità raccomandabile; per quanto forte, utile e efficiente che possiamo pensare sia, sarà condannata al fallimento.

- Mettere l'infrastruttura creata in un ambiente reale per verificare il rendimento come risultato di carichi reali. Si ricorda che il traffico in una rete reale è molto

altalenante, sia in quantità che nel tempo, cioè non si può stimare. Notare che possono sorgere ritardi non citati nel lavoro, come per esempio, dovuti a server DNS quando l'utente invia la richiesta. In un ambiente reale i router e il SGR devono avere degli indirizzi pubblici. Trattando i casi concreti gli indirizzi possono essere IPv4, dei quali ne rimangono pochi, e IPv6, cioè il nuovo protocollo per fornire indirizzamento ai dispositivi. In quest'ultimo caso si possono trovare problemi di interoperabilità tra i router perché ancora non è molto utilizzato.

Bibliografia

[1] I. C. Office. Data protection technical guidance note: Privacy enhancing technologies (pets). *Technical report*, Abril 2006.

[2] O. de las Naciones Unidas. Derechos humanos para todos. *Declaración Universal de los Derechos humanos*, 1948.

[3] A. Pfitzmann and M. Hansen. Anonymity, unobservability, and pseudonymity: A consolidated proposal for terminology. *Draft*, July 2000.

[4] G. Danezis. Better anonymous communications. *PhD thesis*, July2004.

[5] A. Serjantov. On the anonymity of anonymity systems. *PhD thesis*, June 2004.

[6] C. Díaz. Anonymity and privacy in electronic services. *PhD thesis*, December 2005.

[7] K. Chatzikokolakis. robabilistic and information-theoretic approaches to anonymity. *PhD thesis*, October 2007.

[8] I. Goldberg. A pseudonymous communications infrastructure for the internet. *PhD thesis, UC Berkeley*, December 2000.

[9] G. Danezis and C. Diaz. A survey of anonymous communication channels. technical report msr-tr-2008-35. *Microsoft Research*, January 2008.

[10] E. I. S. Work Package 13. D13.1 identity and impact of privacy enhancing technologies. *Technical report*, May 2007.

[11] A. Acquisti, S. Gritzalis, C. Lambrinoudakis, and S.D.C. di Vimercati. Digital privacy. *Auerbach Publications*, 2008.

[12] V. Shmatikov. Probabilistic model checking of an anonymity system. *Journal of Computer Security*, 2004.

[13] A. Beimel and S. Dolev. Buses for anonymous message delivery. *Journal of Cryptology*, 2003.

[14] D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, October 1985.

[15] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, February 1981.

[16] O. Berthold, A. Pfitzmann, and R. Standtke. The disadvantages of free mix routes and how to overcome them. *Lecture Notes in Computer Science*, 2001.

[17] G. Danezis, C. Díaz, and C. Troncoso. Two-sided statistical disclosure attack. *Proceedings of the Seventh Workshop on Privacy Enhancing Technologies (PET 2007)*, June 2007.

[18] M. Liberatore and B. Ñ. Levine. Inferring the source of encrypted http connections. *Proceedings of the 13th ACM conference on Computer and Communications Security (CCS 2006)*, October 2006.

[19] V. Shmatikov and M.-H. Wang. Timing analysis in low-latency mix networks: Attacks and defenses. *Proceedings of ESORICS 2006*, September 2006.

[20] M. Reiter and A. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, June 1998.

[21] D. Mazières and M. F. Kaashoek. The design, implementation and operation of an email pseudonym server. *Proceedings of the 5th ACM Conference on Computer and Communications Security (CCS 1998)*, November 1998.

[22] B. Pfitzmann and A. Pfitzmann. How to break the direct rsa-implementation of mixes. *Proceedings of EUROCRYPT 1989*, 1990.

[23] A. Pfitzmann, B. Pfitzmann, and M. Waidner. Isdn-mixes: Untraceable communication with very small bandwidth overhead. *Proceedings of the GI/ITG Conference on Communication in Distributed Systems*, February 1991.

[24] O. Berthold, H. Federrath, and S. Kopsell. Web mixes: A system for anonymous and unobservable internet access. *Lecture Notes in Computer Science*, 2001.

[25] O. Berthold, A. Pfitzmann, and R. Standtke. The disadvantages of free mix routes and how to overcome them. *Springer-Verlag*, July 2000.

[26] C. Díaz, G. Danezis, C. Grothoff, A. Pfitzmann, and P. F. Syverson. Panel discussion - mix cascades versus peer-to-peer: Is one concept superior? *Privacy Enhancing Technologies*, 2004.

[27] G. Danezis. Mix-networks with restricted routes. *???*, 2003.

[28] R. Dingledine, V. Shmatikov, and P. Syverson. Synchronous batching: From cascades to free routes. *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*, May 2004.

[29] C. Gulcu and G. Tsudik. Mixing e-mail with babel. *Proceedings of the Network and Distributed Security Symposium - NDSS 96*, February 1996.

[30] U. Moller, L. Cottrell, P. Palfrader, and L. Sassaman. Mixmaster protocol - version 2. *IETF Internet Draft*, July 2003.

[31] G. Danezis, R. Dingledine, D. Hopwood, and N. Mathewson. Mixminion: Design of a type iii anonymous remailer protocol. *???*, 2002.

[32] G. Danezis and B. Laurie. Minx: A simple and eficient anonymous packet format. *???*, ???

[33] G. Danezis and J. Clulow. Compulsion resistant anonymous communications. *Proceedings of Information Hiding Workshop (IH 2005)*, June 2005.

[34] S. Mauw, J. Verschuren, and E. de Vink. A formalization of anonymity and onion routing. *roceedings of ESORICS 2004*, 2004.

[35] P. Syverson, M. Reed, and D. Goldschlag. Onion routing access configurations. *Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX 2000)*, 2000.

[36] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr. Towards an analysis of onion routing security. *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, July 2000.

[37] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[38] Z. Brown. Cebolla: Pragmatic ip anonymity. *Proceedings of the 2002 Ottawa Linux Symposium*, June 2002.

[39] G. Danezis. The traffic analysis of continuous-time mixes. *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*, May 2004.

[40] B. N. Levine, M. K. Reiter, C. Wang, and M. K. Wright. Timing attacks in low-latency mix-based systems. *Proceedings of Financial Cryptography (FC '04)*, February 2004.

[41] A. Serjantov and P. Sewell. Passive attack analysis for connection-based anonymity systems. *Proceedings of ESORICS 2003*, October 2003.

[42] Y. Zhuand X. Fu, B. Graham, R. Bettati, and W. Zhao. On flow correlation attacks and countermeasures in mix networks. *In Proceedings of Privacy Enhancing Technologies workshop*, May 2004.

[43] X. Wang, S. Chen, and S. Jajodia. Tracking anonymous peer-to-peer voip calls on the internet. *Proceedings of the ACM Conference on Computer and Communications Security*, November 2005.

[44] A. Back, U. Moller, and A. Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. *Proceedings of Information Hiding Workshop (IH 2001)*, April 2001.

[45] N. Mathewson and R. Dingledine. Practical traffic analysis: Extending and resisting statistical disclosure. *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*, May 2004.

[46] L. Overlier and P. Syverson. Locating hidden servers. *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, May 2006.

[47] M. J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002)*, November 2002.

[48] G. Danezis and B. Wittneben. The economics of mass surveillance and the questionable value of anonymous communications. *Proceedings of the Fifth Workshop on the Economics of Information Security (WEIS 2006)*, June 2006.

[49] M. Rennhard and B. Plattner. Introducing morphmix: Peer-to-peer based anonymous internet usage with collusion detection. *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2002)*, November 2002.