

# Implementazione di una chat in linguaggio Java con crittografia asimmetrica

Progetto del corso di Reti di Calcolatori

Valentino Marano, Amerigo Mancino

Giugno 2015



# Indice

<b>1</b>	<b>Presentazione degli obiettivi</b>	<b>3</b>
<b>2</b>	<b>Progetti a confronto</b>	<b>3</b>
<b>3</b>	<b>Sviluppo degli Obiettivi</b>	<b>3</b>
<b>4</b>	<b>Il package app</b>	<b>5</b>
4.1	Avvio dell'applicazione . . . . .	5
4.2	Interfaccia Grafica . . . . .	5
4.3	Criptaggio dei messaggi . . . . .	6
<b>5</b>	<b>Il package net</b>	<b>6</b>
5.1	Ricerca di utenti . . . . .	6

# 1 Presentazione degli obiettivi

Il nostro progetto per il corso di Reti di Calcolatori si è concentrato sullo sviluppo di una applicazione di messaggistica per reti locali (LAN). Si è progettato il programma in modo tale da sfruttare il protocollo UDP per lo scambio dei messaggi, i quali si ipotizzano essere criptati tramite crittografia asimmetrica con chiave pubblica/privata. Per di più, l'applicazione è stata pensata per essere in grado di inviare anche allegati (che definiamo, a questo proposito, come file generici quali immagini, audio, etc), anch'essi criptati. Il programma, teoricamente, deve essere in grado di identificare tutti gli utenti connessi alla LAN, così da rendere possibile scegliere con quale utente si desidera chattare, similmente a quanto avviene con applicazioni note quali Skype o Telegram Desktop. Inoltre deve godere di un'interfaccia grafica intuitiva per semplificare le operazioni.

# 2 Progetti a confronto

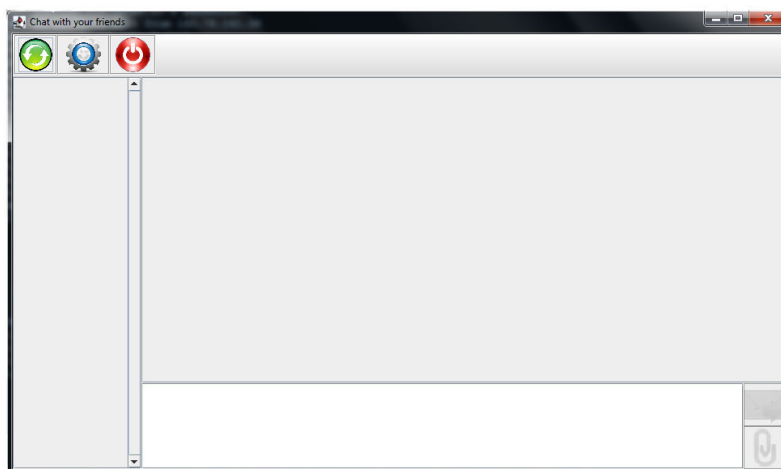
L'idea della chat non è di certo innovativa. Ne esistono di molti tipi da diversi anni e ognuna di loro ha manifestato il proprio momento di gloria, per poi abbandonare i riflettori e dirigere gli utenti verso applicazioni più nuove e immediate.

Nell'ambito dei progetti realizzati in passato per il corso di Reti di Calcolatori, in particolare, nel 2006 uno studente ha proposto un servizio di messaggistica su connessione cifrata con SSL ad architettura p2p, non realizzando tuttavia tutte le premesse e abbandonando quindi l'implementazione della cifratura con SSL e, di conseguenza, la gestione dei certificati. Questo lavoro era stato svolto sfruttando la libreria Axis di Apache Foundation.

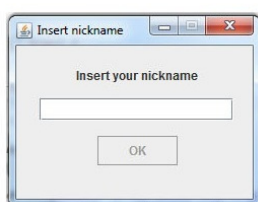
Nel 2014 altri due studenti hanno realizzato una chat prevedendo un'unità centrale che potesse immagazzinare i dati strettamente necessari sugli utenti, senza quindi affidare dati ridondanti a tutti i client. Il loro obiettivo era quello di creare un servizio che fosse a metà fra le chat peer to peer e le chat centralizzate.

# 3 Sviluppo degli Obiettivi

Quasi tutte le parti poste nella Presentazione degli Obiettivi sono state correttamente sviluppate e implementate. La chat realizzata possiede una GUI per l'interazione con l'utente avente il seguente aspetto:



Il riquadro in basso permette la scrittura di testo. I due tasti laterali ad esso affiancati consentono invece di inviare messaggi (criptati) o allegati (non criptati). In alto sono presenti tre bottoni: di essi, quello più a sinistra consente di scandire la rete per la ricerca di eventuali utenti che, in questo momento, stanno utilizzando la chat. Tale scansione viene in ogni caso eseguita in automatico all'avvio del programma e ripetuta saltuariamente durante la sua esecuzione. Quando un utente viene trovato, nel riquadro a sinistra dell'applicazione viene inserito un pulsante con il suo nickname. Cliccandoci sopra, diventa possibile chattare con lui e inviargli messaggi. Il bottone al centro, invece, gestisce le impostazioni (suoni, etc), mentre quello più a destra consente di chiudere l'applicazione. La prima volta che il programma viene lanciato, inoltre, compare una schermata di selezione del login:



In questa scheda l'utente crea il proprio account, scegliendo un nickname da usare in chat e vengono generate le chiavi pubblica e privata. Una volta digitato il nickname, il tasto di conferma diventa cliccabile e ed è possibile iniziare ad usare l'applicazione.

In questa relazione, analizzeremo a grandi linee la codifica proposta, con particolare enfasi sul ruolo giocato da ognuna di esse. Mostreremo, poi, i risultati ottenuti e, infine, il funzionamento dell'intero programma qui presentato, esponendo anche i test fatti in corso d'opera in locale e sulla Virtual

Private Network di ateneo. Il lavoro svolto sarà, quindi, accompagnato dalle dovute conclusioni e da un commento sui possibili sviluppi futuri.

## 4 Il package app

Il codice è stato diviso in due package: uno che si occupa di gestire l'applicazione e la codifica dei messaggi, e l'altro che invece contiene tutte le classi inerenti allo scambio e alla trasmissione dei messaggi stessi. Il package **app** contiene tutte e sole le classi che si occupano del corretto funzionamento dell'applicazione, a partire dalla definizione della schermata di login fino ad arrivare alla gestione delle chiavi. Analizzeremo più dettagliatamente nei prossimi sotto-paragrafi l'uso delle singole classi.

### 4.1 Avvio dell'applicazione

La classe **Main** è la classe principale dell'applicazione. Possiede un unico metodo **main** che si occupa di lanciare il programma eseguendo **run**, ammesso che l'utente abbia effettuato il login in questa o in una precedente sessione. In caso contrario, viene fatto scegliere all'utente un nickname da usare nella chat mediante il pannello grafico mostrato in precedenza e, solo dopo questa operazione, l'applicazione può essere correttamente utilizzata.

### 4.2 Interfaccia Grafica

La classe **App**, che implementa **Runnable**, si occupa della creazione e della gestione della finestra grafica della chat. La GUI, in particolare, è stata realizzata mediante **Swing**, un framework per Java orientato allo sviluppo di interfacce grafiche. I principali metodi qui presenti si occupano, quindi, di generare la GUI, ma non solo: le conversazioni avvenute fra gli utenti vengono salvate in locale e, se la conversazione con un utente già noto viene ripresa in un qualche futuro, allora i precedenti messaggi scambiati con lui sono caricati da un file del tipo

`nickname.txt`

e mostrati nella schermata principale dell'applicazione.

Il file di configurazione **.chat**, invece, è un file che permette di tenere traccia del nickname dell'utente. Quando avvia l'applicazione per la prima volta, infatti, compare la schermata grafica mostrata in precedenza per l'inserimento di un username. Il tasto **OK** è disabilitato fintanto che l'utente non ha scritto almeno una stringa valida, ossia una stringa più lunga di zero

caratteri. Premendo il tasto di conferma, viene salvata nel file con estensione `.chat` una stringa del tipo:

`NICK: nickname`

insieme ad altre informazioni di configurazione, quali `time to sleep`, `buffer lenght`, etc.

### 4.3 Criptaggio dei messaggi

La classe principale che implementa la cifratura dei messaggi è la classe `Encryption`. La codifica sfrutta l'algoritmo a chiave pubblica RSA (dal nome di coloro che lo hanno proposto, Rivest, Shamir, Adleman), il quale utilizza operazioni in modulo per generare le chiavi e l'assunzione che violare la chiave privata sia un problema computazionalmente non trattabile. Per realizzarlo, è stata usata la classe `KeyPair` di Java, che permette di generare una coppia di chiavi (pubblica e privata). Queste vengono estratte e salvate in locale su due file nascosti `.public` e `.private` e usate poi per criptare e decriptare i messaggi.

## 5 Il package net

Nel package `net` sono organizzate tutte le classi che si occupano di effettuare e gestire lo scambio dei messaggi e di ricercare altri utenti nella rete.

### 5.1 Ricerca di utenti

La classe che si occupa della scansione è la classe `scan`. Per effettuare la ricerca viene utilizzato `Nmap`, un software libero distribuito con licenza GNU GPL creato originariamente per effettuare port scanning, cioè mirato all'individuazione di porte aperte su un computer bersaglio o anche su range di indirizzi IP, in modo da determinare quali servizi di rete siano disponibili. Un tipico esempio dell'uso di `Nmap` è il seguente:

```
nmap www.google.it
```

```
Starting Nmap 6.40 ( http://nmap.org ) at 2015-06-05 09:47 CEST
Nmap scan report for www.google.it (74.125.206.94)
Host is up (0.026s latency).
Not shown: 998 filtered ports
PORT      STATE SERVICE
80/tcp    open  http
443/tcp   open  https
```

Nello specifico, poi, per quanto concerne la nostra applicazione, **Nmap** viene utilizzato principalmente per individuare chi utilizza il servizio di messaggistica. Per eseguire **Nmap** viene utilizzata la classe **Process** di Java. Inoltre, quando viene avviata una nuova scansione, tutte le coppie

$\langle \text{Nickname}, \text{indirizzo} \rangle$

trovate fin'ora vengono buttate via per essere sostituite da quelle identificate per mezzo della nuova scansione.

## 5.2 Scambio di messaggi

Lo scambio di messaggi non sfrutta, come si è intuito dai paragrafi precedenti un'architettura client-server, ma il peer-to-peer. Fondamentalmente, tutti i nodi sono equivalenti e possiedono un server sempre in ascolto per la ricezione dei messaggi ed un client predisposto, invece, all'invio dei messaggi stessi.