Progetto di
Sicurezza Informatica e Internet

Progetto A2 - Encrypted 2D Barcodes

Giovanni Rossi - gio.rossi.1991@gmail.com
Pasquale Verlotta - pasquale.verlotta@gmail.com

Indice

Elenco delle figure

Capitolo 1

Introduzione

1.1 Scopo del progetto

Il progetto sviluppato in questi mesi per il corso di "Sicurezza Informatica e Internet" si propone come scopo quello di realizzare un'applicazione che permetta a due o più utenti di scambiarsi documenti cartacei in modo sicuro. In particolare questo significa che chi utilizza tale sistema deve essere in grado di compilare un foglio con informazioni sensibili e di mandarlo ad un insieme di utenti facendo in modo che nessun altro al di fuori dei destinatari possa leggerne il contenuto. Viceversa i destinatari devono essere altrettanto sicuri del fatto che durante il trasferimento nessuno possa aver alterato il documento senza che chi lo riceve se ne accorga. La sfida più importante è rappresentata dal fatto che le informazioni da proteggere vengono trasportate insieme al documento stampato. Un modo di realizzare questa funzionalità è quello di inserire dei *QR-Code*, contenenti le informazioni cifrate, all'interno della pagina da inviare, in modo tale che i destinatari, purché in possesso delle chiavi, riescano a decifrarlo e a leggerlo agevolmente. Tuttavia bisogna garantire anche che il documento in questione non venga contraffatto durante il percorso: se non ci fosse nessuna garanzia su questo, chiunque potrebbe cambiare i QR-Code relativi a certe informazioni (per esempio con un attacco Man In The Middle) e far credere che tutto sia rimasto inalterato ai reali destinatari.

Per lo sviluppo di questo progetto si è deciso di contestualizzare il tutto ad un ambiente di tipo amministrativo dove lo scambio di documenti è molto frequente. La contestualizzazione del progetto è molto importante perché da questo dipendono le modalità di utilizzo del servizio stesso, e quindi, di conseguenza l'architettura generale del sistema che cambia in modo considerevole a seconda dei casi. Nei prossimi capitoli sarà spiegato meglio questo

punto.

Riassumendo quindi, per quanto riguarda le funzionalità principali, il sistema dovrebbe permettere di:

- compilare un documento contenente qualsiasi tipo di informazione;
- cifrare in tutto o in parte il documento appena scritto;
- allegare le informazioni cifrate al documento stesso così che il o i destinatari possano leggerlo;
- firmare il documento in modo tale che il mittente non possa ripudiarne la provenienza e i destinatari possano accorgersi di eventuali manomissioni durante il trasferimento;
- specificare diversi livelli di privilegio gerarchici per la lettura delle informazioni.

Quest'ultimo punto in particolare rappresenta un requisito importante. Può capitare, infatti, di voler pubblicare un certo documento pur mantenendo segrete alcune informazioni sensibili da lasciare a disposizione di pochi eletti. In questo caso la soluzione più ovvia potrebbe essere quella di pubblicare versioni diverse dello stesso documento oscurando opportunamente sulle diverse copie le informazioni che si vogliono tenere segrete a ciascun gruppo. Questo processo può risultare lungo, macchinoso e poco sicuro: è possibile, infatti, che qualche copia possa finire al destinatario sbagliato se non si prendono opportune precauzioni. Ci si è quindi occupato di realizzare un sistema che permettesse di cifrare contenuti particolari, con chiavi speciali (che da ora in poi chiameremo **chiavi di livello**), le quali distribuite ad utenti opportuni, permettono di nascondere informazioni ad alcuni gruppi di utenti che hanno un **livello di fiducia** (o **Trust Level**) più basso.

1.2 Tecnologie Utilizzate

Gli obiettivi appena illustrati, danno un'idea delle tante problematiche che l'applicazione deve affrontare. Per questo sono molte le tecnologie che sono state messe in gioco. Di seguito si riportano quelle utilizzate maggiormente.

1.2.1 Strumenti di sviluppo

Per quanto riguarda gli strumenti, si è fatto uso di

- **Eclipse Luna** come IDE per la gestione del progetto e dei file sorgente;
- **Apache Maven v2.2** per la gestione delle dipendenze e del ciclo di vita dell'applicazione;
- **Git** per il *versioning* e la condivisione dei sorgenti nel team (in particolare si è usato un repository pubblico[1] su *GitHub* come piattaforma online di condivisione[2]);

1.2.2 Linguaggio e librerie utilizzate

In questo progetto è stato utilizzato `Java` come linguaggio di programmazione principale. I motivi di questa scelta ricadono principalmente su:

- il paradigma Object-Oriented che ha permesso l'utilizzo di architectural e creational pattern per risolvere le problematiche più comuni, come Façade, Factory Method, Abstract Factory, DAO e Singleton che sono stati ampiamente utilizzati nello sviluppo;
- l'alta manutenibilità del codice;
- la portabilità su altre piattaforme diverse da quella di sviluppo;
- la grande quantità di librerie presenti a supporto degli obiettivi descritti prima.

Per quanto riguarda invece i singoli moduli del software sviluppato, si sono utilizzati diversi framework e librerie. Per la parte di sicurezza c'è:

- `JCA/JCE` (Java Cryptographic Architecture/Java Criptographic Extension) come framework e provider dei principali algoritmi di cifratura, firma e hashing;
- `BouncyCastle` che si integra con JCA/JCE e fornisce un maggior numero di algoritmi di cifratura e firma nonché funzioni di generazione di certificati;

Per il modulo di elaborazione dei documenti:

- `ImageMagik` per la manipolazione delle immagini dei documenti acquisiti per la decodifica;

[1] Per certi versi si è seguito il principio di *Auguste Kerckhoffs* secondo cui "*in un sistema crittografico è importante tener segreta la chiave, non l'algoritmo di crittazione*".

[2] Link del repository:`https://github.com/WAFcoding/ProgettoSicurezza.git`

- `ZXing` per la generazione e lettura di QR-Code (in generale supporta molti tipi di codici a barre sia 1D che 2D);
- `iTextPDF` per la generazione di documenti ad alta risoluzione ed il posizionamento preciso di testo e QR-Code nel documento;
- `Tesseract` ed in particolare il wrapper Java `Tess4J` per il recupero del testo dopo la scansione dei documenti.

Per la parte di persistenza invece si è fatto uso di:

- `MySQL` come database relazionale per il sistema di *provisioning* delle chiavi e la gestione degli utenti;
- `SQLite` per la gestione dei dati locali di ogni singolo utente (sotto forma di database cifrato);
- `Hibernate` come framework per la gestione dei database (MySQL e SQLite), ed in particolare il modulo `ORM` (Object-Relational Mapping).

Capitolo 2

Scelte Progettuali

2.1 Diagrammi UML

2.2 Architettura

2.3 Elaborazione dei documenti

Capitolo 3

Test

3.1 Funzionamento dell'applicazione

3.2 Test Sperimentali

Capitolo 4

Conclusioni

Appendice A

Manuale di installazione