Relazione Progetto 1 Programmazione 2

SCELTE PROGETTUALI

Le implementazioni sono più di due, ma sono divise in due categorie distinte, che dipendono dalle diverse interfacce dato e utente.

FUNZIONI AGGIUNTIVE COMUNI ALLE DUE CATEGORIE DI IMPLEMENTAZIONI

-GetUserId() non necessita di alcun parametro, può essere eseguita senza registrarsi come utente e senza accedere come utente. Permette di stampare su schermo tutti gli Id che sono già stati usati

IMPLEMENTAZIONE 1

Nella prima versione dell'implementazione ogni dato è caratterizzato da una tripla di informazioni:

- Owner: l'utente che ha creato il dato e l'unico utente che può accedere al dato in scrittura con operazioni di modifica
- Data : il dato vero e proprio, di tipo generico
- Others: una collezione degli utenti che possono accedere al dato in lettura. L'autorizzazione alla lettura può essere concessa solo da Owner, che è anche il primo di questa collezione.

In questa versione i dati non sono copiati se condivisi, ma, per la stessa struttura non è possibile cifrare i dati. Infatti i dati dovrebbero essere cifrati e inseriti al momento della creazione con la chiave del creatore, che tuttavia è sconosciuta agli altri.

IMPLEMENTAZIONE 1.1 MySecureDataContainer11

Questa prima implementazione fa uso dei vettori come struttura di supporto.
All'interno sono presenti due vettori, uno di utenti e uno di dati. Ogni dato contiene un vettore per gli utenti che possono accedervi in lettura e un singolo utente in scrittura

IMPLEMENTAZIONE 1.2 MySecureDataContainer12

Questa seconda implementazione fa uso di TreeSet come struttura di supporto, sia per mantenere gli utenti e i dati, sia per mantenere, all'interno di ogni dati, gli utenti che possono accedervi in lettura

FUNZIONI AGGIUNTIVE

-public int GetOwnerSize(String Id, String passw); che restituisce il numero di elementi creati dall'utente

-public Iterator<E> GetOwnerIterator(String Id, String passw); che restituisce gli elementi creati dall'utente

Questi due nuovi metodi non sono stati aggiunti direttamente all'interfaccia poiché nelle successive implementazioni coincidono con dei metodi già presenti

IMPLEMENTAZIONE 2

Nella seconda versione dell'implementazione, ogni dato è caratterizzato da una coppia di valori:

-il dato vero e proprio, di tipo byte[] (usando i metodi data.toString().getbytes())
-l'Id dell'utente che ha creato il dato, di tipo stringa

Mentre ogni utente è caratterizzato da una quadrupla di valori:
-L'Id (Una stringa)
-La password (una stringa)
-Una chiave segreta e una chiave pubblica di tipo byte[]

I dati sono inseriti in una collezione cifrati con la chiave pubblica del creatore mediante il cifrario RSA. Per leggere un dato non cifrato è necessario avere la chiave privata che inverte la funzione di cifratura
Al momento della condivisione di un dato viene creato un nuovo dato con il creatore colui che è l'altro id nella condivisione e cifrato con la sua chiave pubblica.
Per come è fatta l'implementazione, ogni dato è accedibile in lettura e scrittura o non è accedibile affatto, ma non in sola lettura (si dovrebbe conoscere la chiave privata del creatore).
La classe EsempioDiCifratura mostra come funziona il cifrario per uno scambio di messaggi.

IMPLEMENTAZIONE 2.1

Questa prima implementazione fa uso dei vettori come struttura di supporto.
All'interno sono presenti due vettori, uno di utenti e uno di dati

FUNZIONI AGGIUNTIVE

-seeData(String Id, String password) tenta di cifrare ogni dato creato con la propria chiave privata. La funzione decrypt solleva un'eccezione nel caso in cui la chiave non sia quella giusta e su schermo verrà scritto il messaggio "non è possibile leggere il dato" in quel caso

TEST DI PROVA

Ogni classe può essere testata tramite il suo main. le istruzioni da eseguire compaiono su terminale e ogni funzione può essere eseguita

Marcello Matteucci
546273