Programmazione II – 2018-19

PRIMO PROGETTO

Prima implementazione

In questa implementazione ho scelto di usare due strutture dati differenti per la gestione delle credenziali e dei dati degli utenti.
Per la gestione delle credenziali, come nome utente e password, viene utilizzata una struttura dati di tipo *hashmap*. La chiave, di tipo String, è lo username mentre il valore associato è un oggetto di tipo *SecurePassword* contenente la password cifrata dell'utente.
Per la gestione dei dati generici viene utilizzata una lista di oggetti di tipo *SecureDataInfo* contenente il dato cifrato o in chiaro, il corrispettivo hash e una lista degli utenti che possono accedervi.
I dati cifrati sono privati e accessibili solo dal proprietario, mentre quelli in chiaro sono pubblici e accessibili da utenti con i dovuti permessi.
La ricerca dei dati all'interno della collezione avviene tramite un confronto tra hash.
Ogni metodo implementato, escluso *createUser*, prima di eseguire ogni operazione provvede ad autenticare l'utente.
Il metodo *void put(String owner, String passw)* prima di inserire il dato nella collezione provvede a cifrarlo.
Il metodo *E get(String owner, String passw, E data)* ritorna la copia decifrata dell'elemento cercato.
Il metodo *void copy(String owner, String passw, E data)* crea una copia cifrata del dato. Se l'elemento da copiare è condiviso con altri utenti la copia verrà salvata senza cifratura e sarà accessibile solo dall'utente fornito come "owner".
Il metodo *void share(String owner, String passw, String other, E data)* permette di dare i permessi di accesso ad un altro utente per un certo dato. L'elemento condiviso viene decifrato in modo irreversibile per permettere agli altri utenti di poterlo leggere correttamente.
Per informazioni riguardo il funzionamento delle classi *SecurePassword* e *SecureDataInfo* guardare il capitolo *Crittografia*.

Seconda implementazione

In questa implementazione ho scelto di usare due strutture dati di tipo *hashmap* sia per la gestione delle credenziali sia per la gestione dei dati degli utenti.
La tabella hash per le credenziali ha come chiave, di tipo String, lo username e il valore associato è un oggetto di tipo *UserCredentials* contenente l'oggetto della password cifrata *SecurePassword* e una lista di tutti gli hash dei corrispettivi dati dell'utente.
La tabella per la gestione dei dati generici dell'utente ha come chiave, di tipo String, l'hash dell'elemento e il valore associato è un oggetto di tipo *SecureDataInfo* contenente il dato cifrato o in chiaro. L'hash usato come chiave è una stringa di struttura *hash.username* o nel caso di un dato condiviso *hash.shared*.
I metodi e la cifratura funzionano esattamente come nella prima implementazione ad eccezione per la ricerca o per il metodo *void copy(String owner, String passw, E data)*.
A differenza della prima implementazione, viene ipotizzato che ad un utente non serva copiare un elemento già presente nella sua collezione, per questo è possibile effettuare una copia di un dato condiviso tramite il metodo *void copy(String owner, String passw, E data).*
Per informazioni riguardo il funzionamento delle classi *SecurePassword* e *SecureDataInfo* guardare il capitolo *Crittografia*.

Crittografia

Sicurezza della password

La classe *SecurePassword* contiene la password cifrata e le impostazioni di cifratura.
La cifratura avviene tramite una funzione di derivazione della chiave che genera un hash di 512bit.
La funzione usata è *PBKDF2 (Password Based Key Derivation Function 2)* che applica più volte una funzione di *HMAC (hash-based message authentication code)* all'input fornito unendolo a dei dati casuali (*salt*). Per *HMAC* si intende una funzione di autenticazione del messaggio utilizzando una funzione di hash che viene utilizzata per verificare l'integrità e l'autenticità del messaggio.

Si preferisce usare una funzione di derivazione della chiave anziché una funzione di hash perché l'output generato è crittograficamente sicuro e non possono essere effettuati attacchi di tipo brute force o rainbow table in tempo computazionalmente accettabile.

Specifiche tecniche:

- PBKDF2
  - 5000 iterazioni
  - 64 bytes di salt generati casualmente da *SecureRandom*
  - HMAC SHA-512
  - Lunghezza dell'output di 512 bit

Sicurezza dei dati

La classe *SecureDataInfo* contiene il dato cifrato o in chiaro, l'hash del dato in chiaro, le impostazioni di cifratura e una lista degli utenti che possono accedervi.
L'hash del dato in chiaro viene calcolato tramite la funzione SHA-512.
I dati sono cifrati tramite l'algoritmo *AES (Advanced Encryption Standard)* in modalità *GCM (Galois/Counter Mode)* con chiave a 128bit.
AES è un cifrario a blocchi considerato uno standard in crittografia, in *SecureDataInfo* viene utilizzato con modalità di cifratura del blocco *GCM*.
GCM è un algoritmo di cifratura autenticata che fornisce confidenzialità, autenticità e integrità.
La chiave di cifratura viene derivata dalla password dell'utente tramite *PBKDF2* con *Hmac SHA-512 [Vedi "Sicurezza della password"]*. Questo garantisce che per ogni dato, fornendo sempre la solita password utente, la chiave di cifratura derivata sia sempre differente dalle altre e dalla password cifrata.

Specifiche tecniche:

- AES
  - 12 byte generati casualmente da *SecureRandom* per il vettore di inizializzazione
  - Chiave di cifratura a 128 bit generata da PBKDF2
    - 5000 iterazioni
    - 64 bytes di salt generati casualmente da *SecureRandom*
    - HMAC SHA-512
    - Lunghezza dell'output di 128 bit