Sicurezza Informatica – Laboratorio 4

Gruppo: Angelo Santoro, Christian Guarini, Marcello Carfagno

Il laboratorio 4 consiste nello sviluppo di un KeyRing per conservare chiavi crittografiche. Il KeyRing deve essere disponibile in due versioni: una privata, l'altra pubblica.

Sviluppo del KeyRing

Per la realizzazione del KeyRing nelle due versioni, sono state implementate:

- La classe astratta **KeyRing**, che implementa i metodi:
  - **public** Key getKey(String alias) **throws** Exception: metodo che restituisce la Key associata all'alias, se presente nel KeyRing. Se nel KeyRing non è presente un record avente l'alias passato come input al metodo, viene innalzata una eccezione;
  - **public void** setKey(String alias, Key key, String keyType): metodo per inserire una Key all'interno del KeyRing. Se nel KeyRing è già presente una Key con lo stesso alias, viene sostituita dalla nuova chiave;
  - **private** Record findRecord(String alias): metodo di comodo, utilizzato da getKey(String alias) e setKey(String alias, Key key, String keyType), per cercare all'interno del KeyRing il record avente l'alias passato come input;
- La classe **Record**, innestata alla classe **KeyRing**, i cui attributi sono:
  - **private** String alias;
  - **private byte**[] encodedKey;
  - **private** String keyType: attributo che indica il tipo di chiave memorizzata nel record, nella forma **Algorithm/Format** (e.g. RSA/X.509).

  Per ciascun attributo è stato implementato il metodo getNomeAttributo().
- La classe concreta **PrivateKeyRing** (singleton), figlia di **KeyRing**, che implementa i metodi:
  - **public void** load(InputStream is, **char**[] password) **throws** GeneralSecurityException, IOException, ClassNotFoundException: metodo che permette di caricare il KeyRing privato da file, sfruttando lo stream passato come parametro di ingresso. La password, passata come input, viene utilizzata per generare la SecretKey necessaria alla decifratura del KeyRing privato letto dallo stream;

- **public void** store(OutputStream os, **char**[] password) **throws** GeneralSecurityException, IOException: metodo che permette di memorizzare il KeyRing privato su file, sfruttando lo stream passato come parametro di ingresso. La password, passata come input, viene utilizzata per generare la SecretKey necessaria alla cifratura del KeyRing privato che, successivamente, viene salvato sul file;

- La classe concreta **PublicKeyRing** (singleton), figlia di **KeyRing**, che implementa i metodi:
  - **public void** load(InputStream is) **throws** IOException, ClassNotFoundException: metodo che permette di caricare il KeyRing pubblico da file, sfruttando lo stream passato come parametro di ingresso;
  - **public void** store(OutputStream os) **throws** IOException: metodo che permette di memorizzare il KeyRing pubblico su file, sfruttando lo stream passato come parametro di ingresso.

La struttura dati utilizzata per la realizzazione del KeyRing è un **ArrayList<Record>**.

Test del KeyRing sviluppato

Per il testing del KeyRing, sono state implementate cinque classi dotate di main:

- **RunGenSKR (da avviare solo una volta)**: classe il cui obiettivo è quello di generare il KeyRing privato. Nel main, vengono generate le coppie di chiavi (pubblica/privata) per cifrare e firmare, la chiave AES di 128 bit e la chiave DESede di 168 bit. Successivamente, le chiavi vengono salvate nel PrivateKeyRing tramite il metodo setKey(…). Infine, l'intero keyring viene memorizzato sul disco tramite il metodo store(..);

- **RunGenPKR (da avviare ogni qualvolta si vogliano aggiungere le chiavi pubbliche di un altro gruppo)**: classe il cui obiettivo è quello di generare il KeyRing pubblico. Nel main, le chiavi pubbliche per cifrare e firmare degli altri gruppi vengono salvate all'interno del PublicKeyRing, tramite il metodo setKey(…). Infine, l'intero keyring viene memorizzato sul disco tramite il metodo store(.);

- **RunGenPacket**: classe il cui obiettivo è quello di generare il pacchetto destinato ad un altro gruppo. La signature viene inserita all'interno del pacchetto;

- **RunReadPacket**: classe il cui obiettivo è quello di leggere il pacchetto, inviato da un altro gruppo, e decifrare correttamente il file contenuto in esso.

- **RunEncDec**: classe il cui obiettivo è quello di cifrare e decifrare un file utilizzando la chiave AES a 128 bit o DESede a 168 bit memorizzata nel PrivateKeyRing. Il file sul quale si

opera è generato *on-the-fly* e contiene lo stesso messaggio fornito dalla traccia, in cui sia il gruppo mittente che quello destinatario è il nostro.

Il nome del nostro gruppo è ***Foo***, ed abbiamo interagito con i gruppi: ***Ancora***, ***Doriana***, ***FrankAbba***, ***GalloSalvato***, ***GAS***, ***IPini***, ***Linneo***, ***LupiLupi*** e ***MakeNao***.