Fingerprint

Una delle principali novità introdotte con Android Marshmallow sono le Fingerprint API, dedicate all'autenticazione di utenti tramite le **impronte digitali**. Ovviamente, questa funzionalità è disponibile solo per dispositivi che integrano un apposito scanner.

Il meccanismo sarà inoltre utilizzabile solo su dispositivi ove sia stata **registrata almeno un'impronta digitale**. Lo scopo è impedirne l'accesso non autorizzato, vincolandone la fruizione al superamento della verifica delle impronte digitali.

Per poter far uso di queste classi, è necessario:

- impostare il livello minimo di API alla versione 23 nel file *build.gradle* del modulo applicativo;
- definire la permission di tipo `FINGERPRINT` nel file *AndroidManifest.xml*
- installare una versione degli Android SDK Tools non inferiore alla 24.3.

Il lavoro di autenticazione viene reso piuttosto agevole dalle API disponibili. Infatti, nel metodo onCreate, richiederemo l'istanza di due servizi di sistema: **FingerprintManager**, che si occuperà di svolgere l'autenticazione tramite impronte digitali tramite il metodo authenticate, e **KeyguardManager**, che verificherà la presenza nel dispositivo di un blocco all'accesso (mediante PIN, sequenza di qualche genere o altro):

Successivamente, viene inizializzato un oggetto definito come CryptoObject. Questo sarà passato al FingerprintManager per rendere sicura la comunicazione con lo scanner di impronte. L'inizio dell'autenticazione coinciderà con il metodo onResume che,coincide con l'inizio dell'interazione utente con l'Activity.

Nel metodo sopracitato, l'autenticazione avrà realmente inizio solo se le permission necessarie sono state assegnate, e se almeno un'impronta digitale è stata registrata. Il meccanismo che utilizzeremo per impedire di usare l'app fino allo sblocco consisterà, per semplicità, in una normale AlertDialog non cancellabile.

Il metodo authenticate del FingerprintManager, al momento dell'invocazione, riceve il CryptoObject cui abbiamo già accennato, un CancellationSignal che useremo nell'onPause per interdire il meccanismo di autenticazione, ed un riferimento ad un listener i cui metodi saranno invocati all'esito del riconoscimento delle impronte:

Il metodo onAuthenticationSucceeded viene invocato quando viene riconosciuta una impronta digitale valida: interpreteremo ciò come autorizzazione ad utilizzare l'applicazione. Nel nostro esempio, lo sblocco si manifesterà semplicemente nella chiusura della finestra di dialogo.

Crittografia e Keystore

Vediamo ora da dove proviene il Cipher che abbiamo richiamato per creare il CryptoObject. In Android, sono disponibili le funzionalità di crittografia del mondo Java:

- **JCA (Java Cryptography Architecture)**, corrispondente al package *java.security*, che include molte funzionalità fondamentali su crittografia e firma digitale;
- **JCE (Java Cryptography Extension)**, etichettata con il package *javax.crypto*, che mette a disposizione API di più alto livello, come appunto il Cipher che useremo.

Un oggetto Cipher verrà proprio inizializzato con il metodo definito nella nostra classe CipherGenerator.

Il Cipher viene ottenuto con il metodo getInstance, prassi comune in JCE come applicazione del pattern Factory. Forniamo una stringa di determinazione dell'algoritmo detta, in genere, *transformation* che specifica in maniera compatta, rispettivamente, l'algoritmo crittografico da usare, la modalità di trattamento dei blocchi, ed il tipo di padding da applicare.

Si faccia attenzione che le combinazioni utilizzabili sono indicate nella documentazione ufficiale, e non tutte sono disponibili in ogni verisone di API Android. Cipher, inoltre, dovrà essere dotato di una chiave crittografica, che gestiremo mediante **AndroidKeyStore**, un meccanismo di sistema che permette di custodirle al sicuro.