

Biometric Prompt

Adriano Brugnoni,
Alessandro Fato,
Simone Festa.



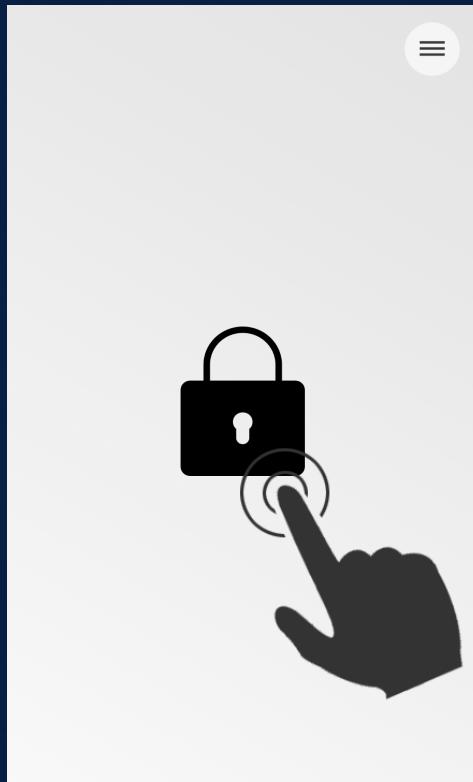
BiometricPrompt

- Metodo sicuro per la protezione di dati sensibili.
- Esistono due principali tipologie di lettori di impronte:
Lettore ottico (confronta il polpastrello con un'immagine dell'impronta) e Ultrasonico (che invia input a ultrasuoni).
- La registrazione e il riconoscimento delle impronte devono avvenire in una parte sicura dell'hardware, un ambiente di esecuzione affidabile noto come Trusted Execution Environment.

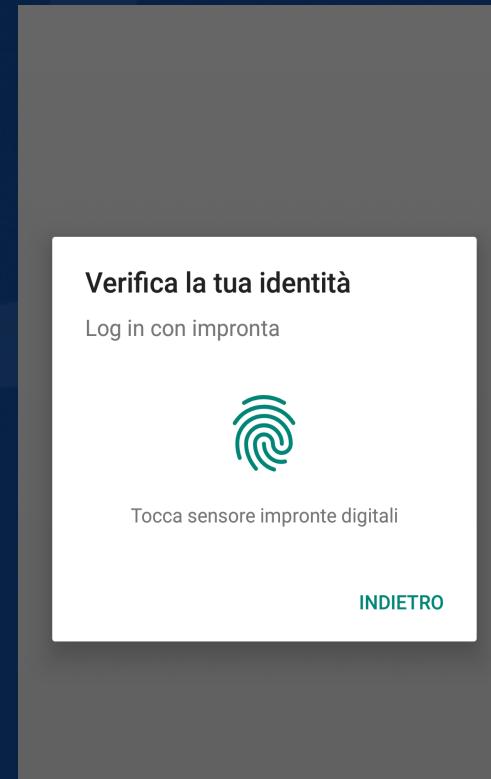


Un'applicazione di esempio

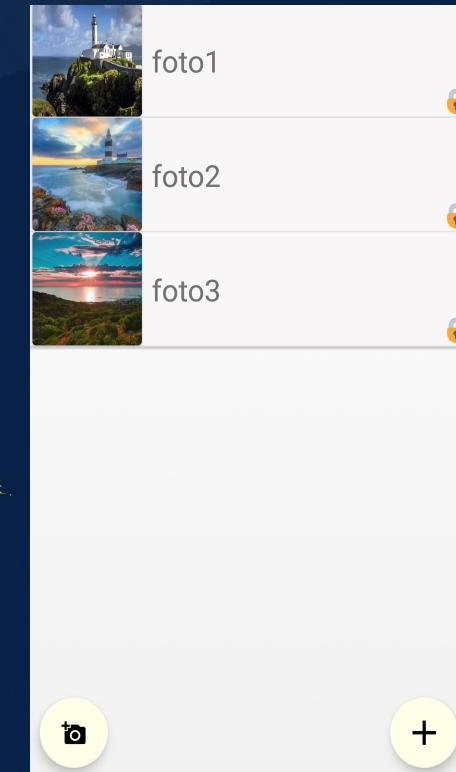
- Main Activity



- Main Activity con prompt



- Activity protetta



BiometricPrompt

- API<28 : La classe **FingerprintManager** coordina l'accesso all'hardware del sensore di impronte digitali (da implementare per ogni dispositivo diverso).
- API≥28 : **FingerprintManager** sostituito con **BiometricPrompt**, fornito dal sistema, si adatta ai vari tipi di dispositivi. Il prompt è automaticamente dismesso quando l'applicazione non è più in foreground, per ragioni di sicurezza. Per renderlo persistente, si dovrebbe ricreare **BiometricPrompt** ogni volta che si crea l'activity.



Per iniziare

- In build.gradle bisogna aggiungere:

```
implementation 'androidx.biometric:biometric:1.0.1'
```

- Nel manifest invece:

```
<uses-permission android:name="android.permission.USE_BIOMETRIC"/>
```

Permette di accedere, con un unico permesso, a tutti i metodi di sblocco sicuri supportati dal device.



Le Activities

- **MainActivity**

Alla pressione del lucchetto, dentro l'apposito metodo **onClick**:

1. verifica la disponibilità delle impronte
2. visualizza il prompt delle impronte digitali
3. apri l'attività protetta su autenticazione riuscita

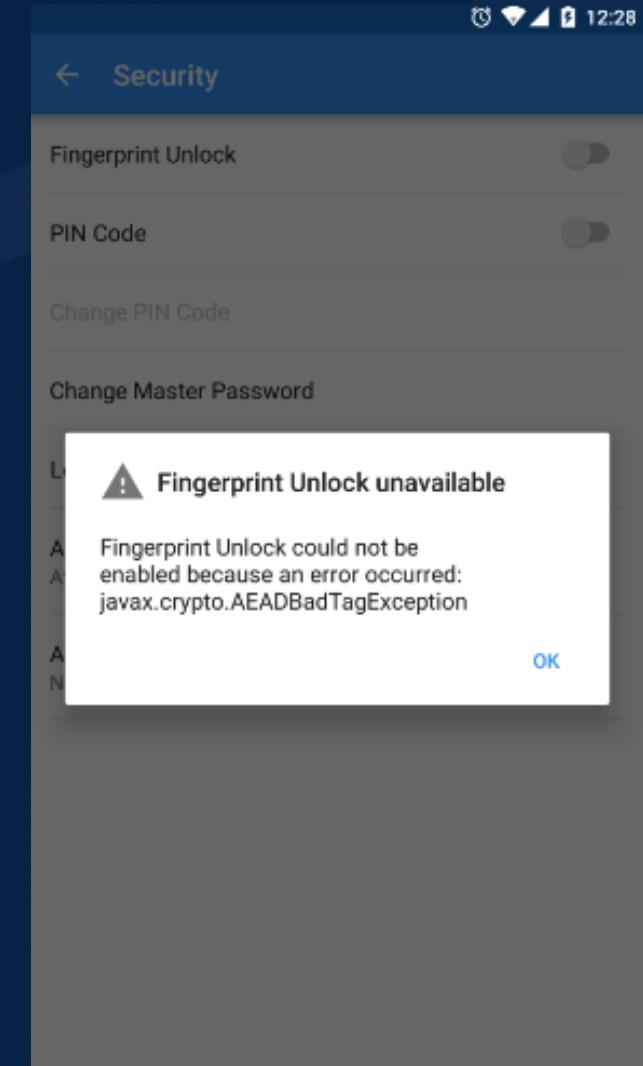
- **ProtectedActivity**

Nell'esempio precedentemente proposto, permette di memorizzare foto nello storage interno e di visualizzarle all'apertura dell'activity. Le foto sono ottenute tramite camera o importate dalla galleria.



Verifica disponibilità impronte

- Anche se oggi il sensore biometrico è presente su tantissimi dispositivi con tecnologie diverse, non bisogna dare per scontata la sua presenza o il suo funzionamento.
- E' dunque consigliato verificare tali requisiti.
- Poiché molti servizi non possono funzionare senza tale riconoscimento, possiamo fare in modo che le applicazioni che lo sfruttino non siano eseguibili su determinati terminali non dotati del sensore.



Verifica disponibilità impronte

- La documentazione Android consiglia di verificare eventuali errori con l'uso della classe **BiometricManager**.
- Il metodo **canAuthenticate** controlla lo stato delle impronte nel nostro dispositivo.
- In caso di errore, possiamo usare un **Alert** per informare l'utente e per preservare la nostra applicazione da eventuali errori.

```
BiometricManager biometricManager = BiometricManager.from(this);
switch (biometricManager.canAuthenticate()) {
    case BiometricManager.BIOMETRIC_SUCCESS:
        Log.d("MY_APP_TAG", "App can authenticate using biometrics.");
        break;
    case BiometricManager.BIOMETRIC_ERROR_NO_HARDWARE:
        Log.e("MY_APP_TAG", "No biometric features available on this device.");
        break;
    case BiometricManager.BIOMETRIC_ERROR_HW_UNAVAILABLE:
        Log.e("MY_APP_TAG", "Biometric features are currently unavailable.");
        break;
    case BiometricManager.BIOMETRIC_ERROR_NONE_ENROLLED:
        Log.e("MY_APP_TAG", "The user hasn't associated " +
            "any biometric credentials with their account.");
        break;
}
```



Verifica disponibilità impronte

- Alla pressione del lucchetto:

```
CheckFingerPrint checkFingerPrint = new CheckFingerPrint(activity);
checkFingerPrint.check();
```

- La classe **CheckFingerprint**:

```
boolean check() {
    BiometricManager biometricManager =
        BiometricManager.from(activity.getApplicationContext());
    switch (biometricManager.canAuthenticate()) {
        case BiometricManager.BIOMETRIC_SUCCESS:
            // App can authenticate using biometrics
            return true;
        case BiometricManager.BIOMETRIC_ERROR_NO_HARDWARE:
        case BiometricManager.BIOMETRIC_ERROR_HW_UNAVAILABLE:
            // Biometric features are currently unavailable
            // No biometric features available on this device
            ModalDialogManager.criticalAlert(activity);
            return false;
        case BiometricManager.BIOMETRIC_ERROR_NONE_ENROLLED:
            // The user hasn't associated any biometric credentials with their account
            ModalDialogManager.enrollmentAlert(activity);
            return false;
    }
    return false;
}
```

Comportamenti

- Nel caso in cui il dispositivo usato non abbia il sensore di impronte (**NO_HARDWARE**) o non funzioni (**HW_UNAVAILABLE**) possiamo usare un **criticalAlert** che avverte l'utente del problema, invitandolo a chiudere l'applicazione. (Questo se l'intera applicazione si basa sull'uso del sensore!).
- Il metodo **check** apre un Dialog a seconda dell'esito del controllo
- I Dialog sono gestiti nella classe **ModalDialogManager**

- Classe **ModalDialogManager**

```
static void criticalAlert(final MainActivity activity)
{
    AlertDialog.Builder alert = new AlertDialog.Builder(activity);
    alert.setCancelable(false);
    alert.setTitle("Critical problem");
    alert.setMessage("Your device has a critical problem with biometric hardw...");
    alert.setPositiveButton("Close app", new Holder(activity));

    alert.show();
}

static class Holder implements DialogInterface.OnClickListener
{
    private MainActivity activity;
    Holder(MainActivity activity) { this.activity = activity; }

    @Override
    public void onClick(DialogInterface dialogInterface, int i) { activity.finish(); }
}
```

Comportamenti

- Non tutti gli errori sono critici, infatti può capitare che l'utente non abbia configurato alcuna impronta (nel caso di ***NONE_ENROLLED***). In questa situazione, possiamo creare una Intent che reindirizzi l'utente nelle impostazioni, per configurare l'impronta digitale.

- Classe **ModalDialogManager**

```
static void EnrollmentAlert(final MainActivity activity)
{
    AlertDialog.Builder alert = new AlertDialog.Builder(activity);
    alert.setCancelable(false);
    alert.setTitle("No Panic!");
    alert.setMessage("Register your fingerprint");
    alert.setPositiveButton("Register", new HolderEnrollment(activity));

    alert.show();
}

static class HolderEnrollment implements DialogInterface.OnClickListener
{
    private MainActivity activity;
    HolderEnrollment(MainActivity activity) { this.activity = activity; }

    @Override
    public void onClick(DialogInterface dialogInterface, int i)
    {
        activity.startActivity(new Intent(android.provider.Settings.ACTION_SECURITY_SETTINGS));
    }
}
```

Visualizzazione Prompt

- Per richiamare a display il controllo dell'impronta dobbiamo importare:

```
import androidx.biometric.BiometricPrompt;
```

```
import java.util.concurrent.Executor;
```

- **Executor** è una interfaccia usata per sostituire la modalità di realizzazione diretta di Thread consentendo l'esecuzione di task asincroni e pool di Thread. BiometricPrompt ne richiede l'utilizzo, come vedremo.



Visualizzazione Prompt

- Alla pressione del lucchetto:

```
// Check fingerprint configuration
CheckFingerPrint checkFingerPrint = new CheckFingerPrint(activity);
if(!checkFingerPrint.check())
    return;

// Create a Biometric Prompt
Executor executor = ContextCompat.getMainExecutor(activity.getApplicationContext());
BiometricPrompt biometricPrompt = new BiometricPrompt(activity, executor, new BiometricOurAuthentication(activity));

BiometricPrompt.PromptInfo promptInfo = new BiometricPrompt.PromptInfo.Builder()
    .setTitle("Check your identity")
    .setSubtitle("Log in using your fingerprint")
    .setNegativeButtonText("Go back")
    .build();

biometricPrompt.authenticate(promptInfo);
```

- Il costruttore di **BiometricPrompt** ha come parametri l'activity, l'executor e una **AuthenticationCallback** (in foto **BiometricOurAuthentication**) che gestisce il callback dell'autenticazione.



AuthenticationCallBack

```
public class BiometricOurAuthentication extends BiometricPrompt.AuthenticationCallback
{
    private MainActivity activity;

    BiometricOurAuthentication(MainActivity activity) { this.activity = activity; }

    @Override
    public void onAuthenticationError(int errorCode, @NonNull CharSequence errString)
    {
        super.onAuthenticationError(errorCode, errString);
        Toast.makeText(activity.getApplicationContext(), errString, Toast.LENGTH_SHORT).show();
    }
}
```

Ha come metodi:

- **onAuthenticationError**: (errorCode & errString sono gestite da Android). Genera un toast con l'errore verificatosi.



AuthenticationCallBack

```
@Override  
public void onAuthenticationSucceeded(@NonNull BiometricPrompt.AuthenticationResult result)  
{  
    super.onAuthenticationSucceeded(result);  
  
    Intent intent = new Intent(activity, ProtectedActivity.class);  
    activity.startActivity(intent);  
}  
  
@Override  
public void onAuthenticationFailed()  
{  
    super.onAuthenticationFailed();  
    Toast.makeText(activity.getApplicationContext(), "Authentication failed", Toast.LENGTH_SHORT).show();  
}
```

- **onAuthenticationSucceded**: permesso consentito, esegui l'activity protetta
- **onAuthenticationFailed**: permesso negato oppure c'è stato un errore nella verifica: abbiamo mosso il dito, il sensore è sporco, etc ...



Creazione della schermata

```
// Create a Biometric Prompt
Executor executor = ContextCompat.getMainExecutor(activity.getApplicationContext());
BiometricPrompt biometricPrompt = new BiometricPrompt(activity, executor, new BiometricOurAuthentication(activity));

BiometricPrompt.PromptInfo promptInfo = new BiometricPrompt.PromptInfo.Builder()
    .setTitle("Check your identity")
    .setSubtitle("Log in using your fingerprint")
    .setNegativeButton("Go back")
    .build();

biometricPrompt.authenticate(promptInfo);
```

- Con **BiometricPrompt.PromptInfo**, mediante Builder, costruiamo il prompt.
- Possiamo impostare titolo, descrizione, sottotitolo, un **NegativeButton** (cancellazione) e **PositiveButton** (conferma azione).



Esempio di BiometricPrompt

setTitle

Check your identity!

Log in using your fingerprint



Tocca sensore impronte digitali

Indicazione fornita da Android

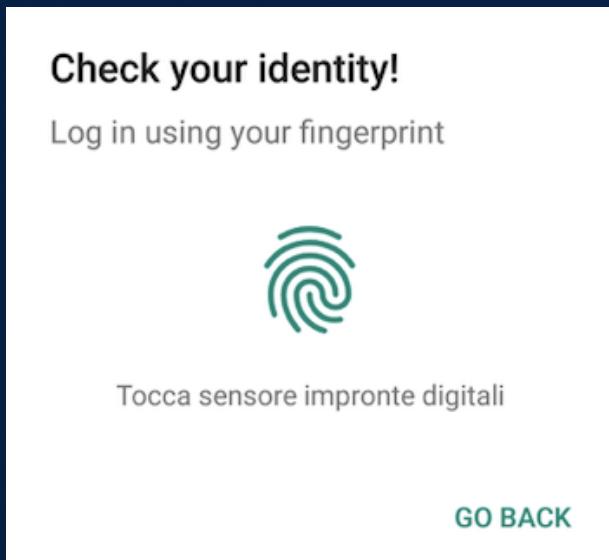
setSubtitle

GO BACK

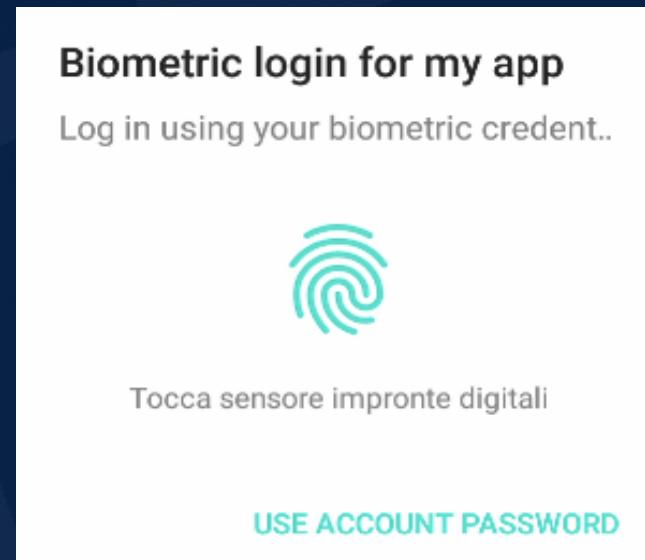
setNegativeButton

Grafica

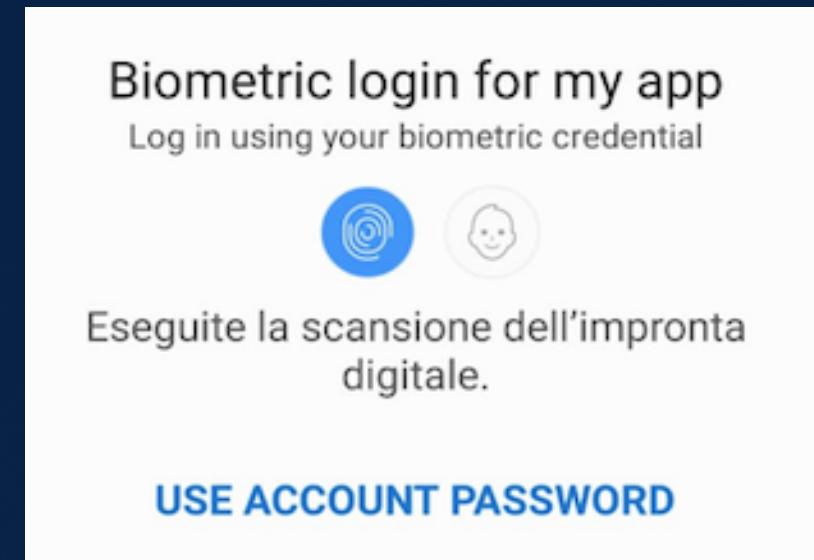
- Ogni produttore utilizza una propria personalizzazione, essa è riscontrabile nella verifica delle impronte digitali. La componente grafica è gestita dal sistema, che rileva anche metodi di sblocco alternativi (sblocco facciale nella terza foto).



Android Stock 8.0



EMUI 8.0



ONE UI 2.0

Grafica

Anche la resa grafica in caso di impronta non correttamente rilevata può essere tranquillamente deliberata ad Android!

Check your identity!

Log in using your fingerprint



Impossibile elaborare l'impronta digitale.
Riprova.

[GO BACK](#)

BiometricPrompt

Presentazione e codice mostrati
sono stati realizzati da :

- Adriano Brugnoni
- Alessandro Fato
- Simone Festa

