Progetto Ingegneria dei sistemi distribuiti Authenticator

Stefano Bognanni
Dipartimento di Matematica e Informatica
Università di Catania
stefano.bognanni97@gmail.com

Abstract

Il processo di autenticazione è diventato ormai fondamentale in ogni aspetto della vita professionale e privata. Dalla classica immissione di una password all'autenticazione mediante biometria, esistono ormai svariati modi per provvedere a questa esigenza, ma quanto tempo e quante risorse siamo disposti a spendere a riguardo? L'intento di questo progetto è quello di proporre un'applicazione che riesca ad autenticare un soggetto in modo veloce, sicuro ed economico.

1 Introduzione

L'obiettivo del progetto è quello di provvedere all'autenticazione di svariati utenti verso una struttura, dipartimentale o aziendale, in modo robusto utilizzando delle risorse tipicamente presenti in questi ambiti come, ad esempio, smartphone Android.

1.1 Scenario

Lo scenario di applicazione quindi è il seguente:
Si vuole autenticare e tener traccia di chiunque entri in una struttura o in aree di essa (ad esempio uffici, aule, laboratori ecc...) e, qualora sia necessario, impedire o consentire l'accesso dell'utente nell'area in questione.

1.2 Tipo di autenticazione

Da un punto di vista della Cyber Security, in ambito di autenticazione uomo-macchina, esistono tre diversi tipi di autenticazione:

Autenticazione per conoscenza: l'utilizzo di un codice che è a conoscenza della singola persona.

Autenticazione per possesso: il possesso di un oggetto affidato ad una ed una sola persona.

Autenticazione per biometria: l'utilizzo di caratteristiche biometriche uniche della singola persona.

Il tipo di autenticazione che si vuole fornire tramite questo progetto combina l'autenticazione per possesso e quella per biometria. Infatti, l'autenticazione per possesso viene perpetrata mediante l'utilizzo di uno smartphone assegnato ad una singola persona, mentre l'autenticazione per biometria è resa possibile mediante le API Android che consentono il rilevamento dell'impronta digitale.

1.3 Design pattern Authenticator

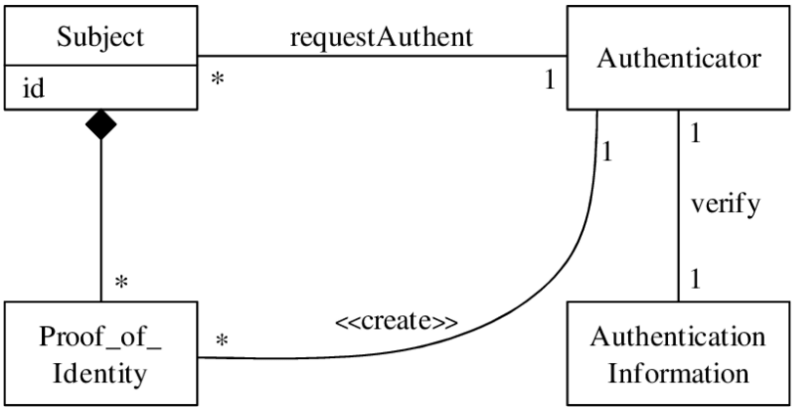
La letteratura dell'ingegneria del software fornisce un design pattern che si occupa delle problematiche relative all'autenticazione che prende il nome di Autenticathor [1].

Subject: utente da autenticare.

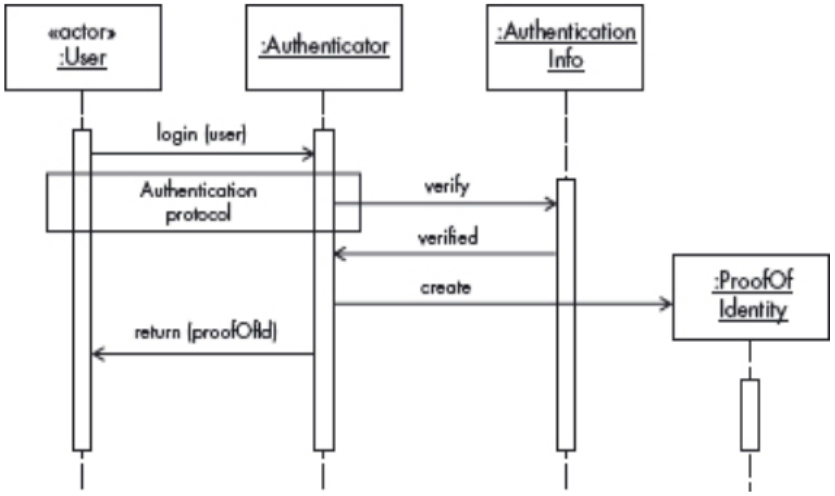
AuthenticationInformation: informazioni relative ai parametri di autenticazione noti.

ProofofIdentity: "Oggetto" assegnato al Subject che certifica l'esito positivo dell'autenticazione.

Authenticator: classe che si occupa di consultare le informazioni note e, nel caso in cui l'autenticazione vada a buon fine, fornire la ProofofIdentity al subject.



(a) UML delle classi.



(b) Diagramma di sequenza.

Figure 1: Pattern Authenticator.

2 Android Studio

Per poter sviluppare un'applicazione per device Android è necessario utilizzare Android Studio, un IDE che permette lo sviluppo di applicazioni [2]. In ogni applicazione Android vi sono degli elementi fondamentali sempre presenti:

Activity: task che l'applicazione deve eseguire.

AndroidMainfest.xml: file contenente le informazioni essenziali per l'applicazione.

activity_"nameofActivity".xml file contenente le informazioni relative al markup dell'interfaccia dell'attività.

"Activity".class: file che contiene il codice dell'attività.

Nei seguenti sotto paragrafi verrà presentato il modo in cui gestire questi elementi per poter implementare le varie funzionalità.

2.1 Fingerprint

Per poter aggiungere utilizzare le Fingerprint API di Android è necessario configurare l'applicazione in modo adeguato [3].
Innanzi tutto è necessario modificare il file "AndroidMainfest.xml" affinché si indichi la volontà di voler utilizzare le API e le periferiche relative all'aquisizione della Fingerprint.

```
1 <uses-feature android:name="android.hardware.fingerprint"
2   android:required="true"/>
3
4   <uses-permission
5   android:name="android.permission.USE_FINGERPRINT" />
```

Per poter creare l'interfaccia utente dell'attività in questione è necessario modificare il file activity_"nameofActivity".xml associato (il seguente codice è stato utilizzato per questo progetto).

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout xmlns:
       android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      tools:context=".MainActivity">
8
9      <RelativeLayout
10         android:id="@+id/activity_main"
11         android:layout_width="match_parent"
12         android:layout_height="match_parent"
```

```
13          android:background="#CCCCCC"
14          tools:context="com.stefano.authenticator">
15
16          <TextView
17              android:layout_width="wrap_content"
18              android:layout_height="wrap_content"
19              android:layout_above="@+id/textView2"
20              android:layout_centerHorizontal="true"
21              android:layout_marginBottom="210dp"
22              android:text="@string/app_name"
23              android:textSize="35dp"
24              android:textAppearance="@style/TextAppearance.
                    AppCompat.Large" />
25
26          <TextView
27              android:layout_width="wrap_content"
28              android:layout_height="wrap_content"
29              android:layout_centerInParent="true"
30              android:id="@+id/textview"/>
31
32
33          <ImageView
34              android:id="@+id/imageView"
35              android:layout_width="155dp"
36              android:layout_height="163dp"
37              android:layout_centerInParent="true"
38              android:background="@drawable/fingerprint" />
39
40          <TextView
41              android:id="@+id/textView2"
42              android:textSize="25dp"
43              android:layout_width="match_parent"
44              android:layout_height="283dp"
45              android:layout_below="@+id/imageView"
46              android:layout_marginTop="-55dp"
47              android:gravity="center"
48              android:text="@string/instructions" />
49      </RelativeLayout>
50
51  </androidx.constraintlayout.widget.ConstraintLayout>
```

Mentre modificando il file “Activity”.class è possibile modificare il comportamento dell’attività.

```
1
```

```
2  public class MainActivity extends AppCompatActivity { //
       Nell'applicazione l'attivit  principale si occupa del
       rilevamento dell'impronta
3
4      private static final String KEY_NAME = "yourKey";
5      private Cipher cipher;
6      private KeyStore keyStore;
7      private KeyGenerator keyGenerator;
8      private TextView textView;
9      private FingerprintManager.CryptoObject cryptoObject;
10     private FingerprintManager fingerprintManager;
11     private KeyguardManager keyguardManager;
12
13     @Override
14     protected void onCreate(Bundle savedInstanceState) {
15         super.onCreate(savedInstanceState);
16         setContentView(R.layout.activity_main);
17         // Semplice controllo sulla versione di Android
               presente, le API fingerprint sono presenti da
               Android Marshmellow in poi
18         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M)
               {
19
20             keyguardManager =
21                     (KeyguardManager) getSystemService(
                           KEYGUARD_SERVICE);
22             fingerprintManager =
23                     (FingerprintManager) getSystemService(
                           FINGERPRINT_SERVICE);
24
25             textView = (TextView) findViewById(R.id.
                   textview);
26
27             //Semplice controllo sulla presenza del sensore
                    per la Fingerprint e der permessi relativi
28             if (!fingerprintManager.isHardwareDetected()) {
29                 textView.setText("Your device doesn't
                       support fingerprint authentication");
30             }
31
32             if (ActivityCompat.checkSelfPermission(this,
                   Manifest.permission.USE_FINGERPRINT) !=
                   PackageManager.PERMISSION_GRANTED) {
33                 textView.setText("Please enable the
                       fingerprint permission");}
```

```
34
35              //Controllo sulla presenza di almeno una
                    fingerprint registrata
36              if (!fingerprintManager.hasEnrolledFingerprints
                    ()) {
37
38                  textView.setText("No fingerprint configured
                        . Please register at least one
                        fingerprint in your device's Settings")
                        ;
39              }
40
41              //Controllo sul lockscreen
42              if (!keyguardManager.isKeyguardSecure()) {
43                  textView.setText("Please enable lockscreen
                        security in your device's Settings");
44              } else {
45                  try {
46                      generateKey();
47                  } catch (FingerprintException e) {
48                      e.printStackTrace();
49                  }
50                  if (initCipher()) {
51                      //Istanza crittografica per la gestione
                            della fingerprint
52                      cryptoObject = new FingerprintManager.
                            CryptoObject(cipher);
53                      // alla classe helper  viene delegato
                            l'onere di interrogare il sistema
54                      Helper helper = new Helper(this);
55                      helper.startAuth(fingerprintManager,
                            cryptoObject);
56                  }
57              }
58          }
59      }
60
61
62
63
64
65
66
67
68
```

```
69      private void generateKey() throws FingerprintException
            { // generazione delle chiavi per poter gestire il
            keystore di Android
70          try {
71
72              keyStore = KeyStore.getInstance("
                    AndroidKeyStore");
73              keyGenerator = KeyGenerator.getInstance(
                    KeyProperties.KEY_ALGORITHM_AES, "
                    AndroidKeyStore");
74              keyStore.load(null);
75
76              keyGenerator.init(new
77
78                      KeyGenParameterSpec.Builder(KEY_NAME,
79                      KeyProperties.PURPOSE_ENCRYPT |
80                              KeyProperties.PURPOSE_DECRYPT)
81                      .setBlockModes(KeyProperties.
                            BLOCK_MODE_CBC)
82                      .setUserAuthenticationRequired(true)
83                      .setEncryptionPaddings(
84                              KeyProperties.
                                    ENCRYPTION_PADDING_PKCS7)
85                      .build());
86
87
88              keyGenerator.generateKey();
89
90          } catch (KeyStoreException
91                   | NoSuchAlgorithmException
92                   | NoSuchProviderException
93                   | InvalidAlgorithmParameterException
94                   | CertificateException
95                   | IOException exc) {
96              exc.printStackTrace();
97              throw new FingerprintException(exc);
98          }
99      }
100
101
102
103
104
105
106
```

```
107     public boolean initCipher() {
108         try {
109             cipher = Cipher.getInstance(
110                     KeyProperties.KEY_ALGORITHM_AES + "/"
111                             + KeyProperties.BLOCK_MODE_CBC
                                + "/"
112                             + KeyProperties.
                                ENCRYPTION_PADDING_PKCS7);
113         } catch (NoSuchAlgorithmException |
114                  NoSuchPaddingException e) {
115             throw new RuntimeException("Failed to get
                    Cipher", e);
116         }
117
118         try {
119             keyStore.load(null);
120             SecretKey key = (SecretKey) keyStore.getKey(
                    KEY_NAME,
121                     null);
122             cipher.init(Cipher.ENCRYPT_MODE, key);
123             return true;
124         } catch (KeyPermanentlyInvalidatedException e) {
125             return false;
126         } catch (KeyStoreException | CertificateException
127                  | UnrecoverableKeyException | IOException
128                  | NoSuchAlgorithmException |
                     InvalidKeyException e) {
129             throw new RuntimeException("Failed to init
                    Cipher", e);
130         }
131     }
132
133     private class FingerprintException extends Exception {
134         public FingerprintException(Exception e) {
135             super(e);
136         }
137     }
138 }
139
140
141
142
143
144
145
```

```
146 //classe helper che si occupa di interrogare il sistema
147 public class Helper extends FingerprintManager.
        AuthenticationCallback {
148
149     private CancellationSignal cancellationSignal;
150     private Context context;
151
152     public Helper(Context mContext) {
153         context = mContext;
154     }
155
156     // metodo che interroga in modo vero e proprio il
            sistema operativo
157     public void startAuth(FingerprintManager manager,
            FingerprintManager.CryptoObject cryptoObject) {
158
159         cancellationSignal = new CancellationSignal();
160         if (ActivityCompat.checkSelfPermission(context,
                Manifest.permission.USE_FINGERPRINT) !=
                PackageManager.PERMISSION_GRANTED) {
161             return;
162         }
163         manager.authenticate(cryptoObject,
                cancellationSignal, 0, this, null);
164     }
165     @Override
166     //caso di errore durante l'autenticazione
167     public void onAuthenticationError(int errMsgId,
            CharSequence errString) {
168
169         Toast.makeText(context, "Authentication error\n" +
                errString, Toast.LENGTH_LONG).show();
170     }
171     @Override
172     //caso di fallimento durante l'autenticazione
173     public void onAuthenticationFailed() {
174         Toast.makeText(context, "Authentication failed",
                Toast.LENGTH_LONG).show();
175     }
176     @Override
177     public void onAuthenticationHelp(int helpMsgId,
            CharSequence helpString) {
178         Toast.makeText(context, "Authentication help\n" +
                helpString, Toast.LENGTH_LONG).show();
179     }
```

```
180     @Override
181     //Autenticazione avvenuta
182     public void onAuthenticationSucceeded(
183             FingerprintManager.AuthenticationResult result)
                    {
184
185         Toast.makeText(context, "Success!", Toast.
                LENGTH_LONG).show();
186         SystemClock.sleep(1000);
187
188         Intent intent = new Intent(context.
                getApplicationContext(), NfcTransmit.class); //
                 viene richiamata l'attivita che si occupa
                della trasmissione NFC
189         context.startActivity(intent);
190     }
191 }
```

2.2 Android Beam

Anche per poter utilizzare le API NFC di Android è necessario configurare l'applicazione in modo adeguato, in particolar modo la fnzione Beam [4].

```
1   <uses-permission android:name="android.permission.NFC"/>
2   <uses-feature android:name="android.hardware.nfc" />
```

Essendo stata implementata come una seconda attività vanno definiti anche interfaccia utente e codice.

```
1
2 <?xml version="1.0" encoding="utf-8"?>
3 <androidx.constraintlayout.widget.ConstraintLayout
4     xmlns:android="http://schemas.android.com/apk/res/
          android"
5     xmlns:app="http://schemas.android.com/apk/res-auto"
6     xmlns:tools="http://schemas.android.com/tools"
7     android:layout_width="match_parent"
8     android:layout_height="match_parent"
9     tools:context="com.stefano.authenticator.NfcTransmit">
10
11     <RelativeLayout xmlns:android="http://schemas.android.
          com/apk/res/android"
12         xmlns:app="http://schemas.android.com/apk/res-auto"
13         xmlns:tools="http://schemas.android.com/tools"
```

```
14          android:id="@+id/activity_main"
15          android:layout_width="match_parent"
16          android:layout_height="match_parent"
17          android:gravity="center"
18          android:background="#add8e6"
19          tools:context="com.stefano.authenticator">
20      <TextView
21          android:id="@+id/tv_out_label"
22          android:textSize="25dp"
23          android:gravity="center"
24          android:layout_width="wrap_content"
25          android:layout_height="wrap_content"
26          android:layout_marginEnd="8dp"
27          android:layout_marginLeft="16dp"
28          android:layout_marginRight="8dp"
29          android:layout_marginStart="16dp"
30          android:layout_marginTop="64dp"
31          android:text="@string/instructions2"
32          android:textAppearance="@style/TextAppearance.
                AppCompat.Medium"
33          app:layout_constraintEnd_toEndOf="parent"
34          app:layout_constraintStart_toStartOf="parent"
35          app:layout_constraintTop_toTopOf="parent"/>
36
37      </RelativeLayout>
38
39
40  </androidx.constraintlayout.widget.ConstraintLayout>
```

Il codice relativo a questa attività per la gestione di Android Beam funziona correttamente se e solo se su entrambi i dispositivi vi è un'applicazione in grado di saper gestire il file che si vuol condividere (in questo caso è necessario che il ricevitore abbia un applicazione in grado di gestire tag di testo, un esempio di ricevitore è presente in Appendice).

```
1  public class NfcTransmit extends AppCompatActivity
       implements OutcomingNfcManager.NfcActivity {
2
3      private NfcAdapter nfcAdapter;
4      private OutcomingNfcManager outcomingNfccallback;
5
6      @Override
7      protected void onCreate(Bundle savedInstanceState) {
8          super.onCreate(savedInstanceState);
9          setContentView(R.layout.activity_nfc_transmitter);
10         //controllo sulla presenza e lo stato dell NFC
11         if (!isNfcSupported()) {
12             Toast.makeText(this, "Nfc is not supported on
                   this device", Toast.LENGTH_SHORT).show();
13             finish();
14         }
15         if (!nfcAdapter.isEnabled()) {
16             Toast.makeText(this, "NFC disabled on this
                   device. Turn on to proceed", Toast.
                   LENGTH_SHORT).show();
17         }
18
19         this.outcomingNfccallback = new OutcomingNfcManager
               (this);
20         this.nfcAdapter.setOnNdefPushCompleteCallback(
               outcomingNfccallback, this);
21         this.nfcAdapter.setNdefPushMessageCallback(
               outcomingNfccallback, this);
22     }
23
24
25     @Override
26     protected void onNewIntent(Intent intent) {
27         setIntent(intent);
28     }
29
30     private boolean isNfcSupported() {
31         this.nfcAdapter = NfcAdapter.getDefaultAdapter(this
               );
32         return this.nfcAdapter != null;
33     }
34
35
36
```

```
37      @Override
38      public String getOutcomingMessage() {
39
40          String deviceId = Settings.System.getString(
                getContentResolver(),
41                  Settings.System.ANDROID_ID);
42
43          return deviceId+encryptedTimestamp();
44      }
45
46      public String encryptedTimestamp(){
47          String encrypted="";
48          try
49          {
50              String key = "ProGettoDiIngegn";
51              Date d = new Date();
52              String date = DateFormat.format("yyyy-MM-dd hh:
                    mm", d.getTime()).toString();
53              Key aesKey = new SecretKeySpec(key.getBytes(),
                    "AES");
54              Cipher cipher = Cipher.getInstance("AES");
55              // encrypt the text
56              cipher.init(Cipher.ENCRYPT_MODE, aesKey);
57              encrypted = cipher.doFinal(date.getBytes()).
                    toString();
58          }
59          catch(Exception e)
60          {
61              e.printStackTrace();
62          }
63          return encrypted;
64
65      }
66
67
68      @Override
69      public void signalResult() {
70          runOnUiThread(() -> Toast.makeText(NfcTransmit.this
                , R.string.message_beaming_complete, Toast.
                LENGTH_SHORT).show());
71          System.exit(0);
72      }
73  }
74
75
```

```
76  //parametri e gestione di Android Beam
77  public class OutcomingNfcManager implements NfcAdapter.
        CreateNdefMessageCallback,
78          NfcAdapter.OnNdefPushCompleteCallback {
79
80      public static final String MIME_TEXT_PLAIN = "text/
            plain";
81      private NfcActivity activity;
82
83      public OutcomingNfcManager(NfcActivity activity) {
84          this.activity = activity;
85      }
86
87      @Override
88      public NdefMessage createNdefMessage(NfcEvent event) {
89
90          String outString = activity.getOutcomingMessage();
91          byte[] outBytes = outString.getBytes();
92          NdefRecord outRecord = NdefRecord.createMime(
                MIME_TEXT_PLAIN, outBytes);
93
94          return new NdefMessage(outRecord);
95      }
96
97      @Override
98      public void onNdefPushComplete(NfcEvent event) {
99          activity.signalResult();
100     }
101
102     public interface NfcActivity {
103         String getOutcomingMessage();
104
105         void signalResult();
106     }
107 }
```

3 WorkFlow

Una volta installata e avviata l'applicazione, l'utente vedrà in esecuzione la "MainActivity", il cui scopo è quello di chiedere a quest'ultimo di autenticarsi al dispositivo mediante Fingerprint. In questa fase il dispositivo genera tutte le istanze degli oggetti necessari per interrogare il sistema operativo e rimane in attesa di un input da parte dell'utente. Non appena l'utente utilizza la propria impronta per autenticarsi, il sistema preleva il campione e lo compara con i campioni salvati all'interno del sistema, se il match non va a buon fine, l'applicazione si limita a riportare la causa dell'errore (impronta non rilevata correttamente, autenticazione fallita, ecc...), altrimenti viene avviata l'attività "NfcTransmit" la quale fa le veci della "Proof_of_Identity". In questa fase dell'esecuzione, dopo aver confermato l'identità del soggetto mediante Fingerprint, l'interfaccia utente si limita ad invitare l'utente ad avvicinare il dispositivo al ricevitore e, una volta rilevata la presenza del ricevitore NFC, Android Beam provvede allo scambio del tag tra i dispositivi.

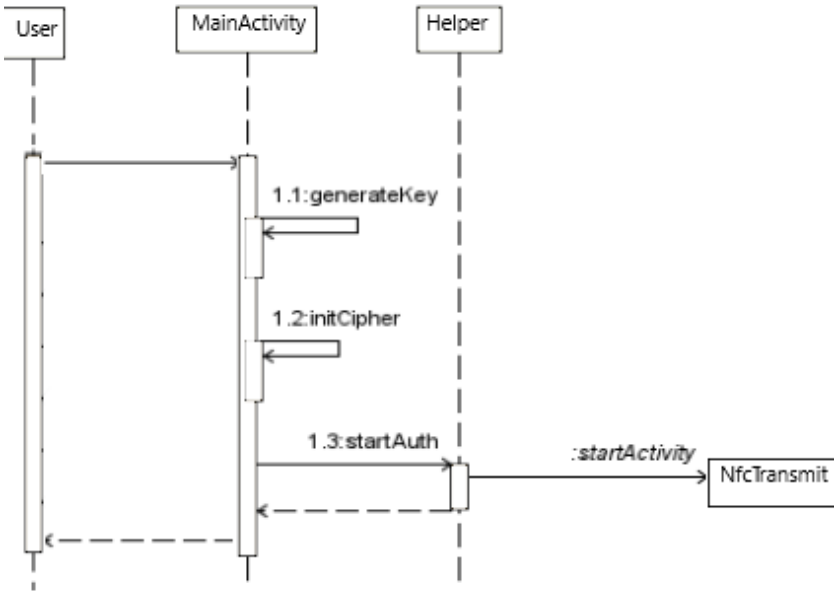


Figure 2: Diagramma di sequenza

Per poter autenticare l'utente mediante possesso, è necessario che il dispositivo ed il ricevitore siano sincronizzati computazionalmente. L'approccio utilizzato in questo progetto struttura il tag in modo che nella prima parte vi sia l'ID del dispositivo, mentre la nella seconda la cifratura del timestamp mediante un segreto associato al suddetto ID.

3.1 Considerazioni di sicurezza

Da un punto di vista della sicurezza, la soluzione proposta risulta essere alquanto robusta. Lo scopo della prima parte del messaggio (deviceId) è quello di consentire al ricevitore di prelevare la chiave associata allo smartphone che sta tentando di immettere le credenziali. La seconda parte invece, ha lo scopo prevenire gli attacchi di replica da parte di un eventuale attaccante. L'utilizzo di un timestamp infatti, certifica la "freshness" del tag generato.

4 Considerazioni finali

In conclusione, l'applicazione proposta (battezzata con il nome di "Autenticator"), risulta essere un'alternativa valida ed immediata alle classiche metodologia di autenticazione presso un luogo d'interesse. Le periferiche utilizzate sono presenti sulla quasi totalità dei dispositivi presenti sul mercato e, d'altro canto, è sempre più comune da parte delle aziende fornirli ai propri dipendenti. Risulta essere chiaro quindi come una soluzione simile abbia un impatto minimo a fronte delle spese necessarie per l'utilizzo dei classici meccanismi per l'autenticazione senza inficiare la sicurezza.

Figure 3: Logo Autenticator.

A Alternative ad Android Beam

Nella fase di sviluppo di questo progetto sono state valutate due alternative all'utilizzo di Android Beam.

Tag Writer: utilizzare lo smartphone come scrittore di tag esterni [7].
Le API Android infatti permettono di scrivere informazioni su tag di terze parti, questo approccio però avrebbe inficiato notevolmente l'utilizzabilità del sistema, in quanto avrebbe costretto l'utente a possedere un tag, scrivere le informazioni mediante l'applicazione e successivamente porlo sul ricevitore NFC.

HCE: utilizzare lo smartphone come una smartcard [6].
Uno degli approcci esistenti per l'autenticazione mediante possesso prevede l'utilizzo di smartcard. Emulare tale sistema utilizzando lo smartphone non avrebbe inficiato l'utilizzabilità del sistema, tuttavia, sebbene sia possibile effettuare tale operazione, l'implementazione di tale sistema risulta essere complessa, e poco scalabile, riducendo di molto la flessibilità dell'implementazione.

(a) Tag NFC. (b) Smartcard.

Figure 4: Esempio tag e smartcard.

B Esempio di Ricevitore

Segue l'esempio di un possobole ricevitore [5].

```
1  public class ReceiverActivity extends AppCompatActivity {
2
3      public static final String MIME_TEXT_PLAIN = "text/
           plain";
4      private TextView tvIncomingMessage;
5      private NfcAdapter nfcAdapter;
6      Map<String, String> map = new HashMap<String, String>()
           ;
7
8
9      @Override
10     protected void onCreate(@Nullable Bundle
           savedInstanceState) {
11         super.onCreate(savedInstanceState);
12         setContentView(R.layout.activity_main);
13
14         if (!isNfcSupported()) {
15             Toast.makeText(this, "Nfc is not supported on
                   this device", Toast.LENGTH_SHORT).show();
16             finish();
17         }
18         if (!nfcAdapter.isEnabled()) {
19             Toast.makeText(this, "NFC disabled on this
                   device. Turn on to proceed", Toast.
                   LENGTH_SHORT).show();
20         }
21
22         initViews();
23     }
24
25     private boolean isNfcSupported() {
26         this.nfcAdapter = NfcAdapter.getDefaultAdapter(this
               );
27         return this.nfcAdapter != null;
28     }
29
30     private void initViews() {
31         this.tvIncomingMessage = findViewById(R.id.
               tv_in_message);
32     }
33
34
```

```
35      @Override
36      protected void onNewIntent(Intent intent) {
37
38          receiveMessageFromDevice(intent);
39      }
40
41      @Override
42      protected void onResume() {
43          super.onResume();
44
45          enableForegroundDispatch(this, this.nfcAdapter);
46          receiveMessageFromDevice(getIntent());
47      }
48
49      @Override
50      protected void onPause() {
51          super.onPause();
52          disableForegroundDispatch(this, this.nfcAdapter);
53      }
54
55      public String decryptTimestamp(String key, String
            encrypted){
56          String decrypted="";
57          try
58          {
59              Key aesKey = new SecretKeySpec(key.getBytes(),
                    "AES");
60              Cipher cipher = Cipher.getInstance("AES");
61
62              cipher.init(Cipher.DECRYPT_MODE, aesKey);
63              decrypted = new String(cipher.doFinal(encrypted
                    .getBytes()));
64          }
65          catch(Exception e)
66          {
67              e.printStackTrace();
68          }
69          return decrypted;
70      }
71
72
73
74
75
76
```

```
77      private void openTheDoor(){
78          Toast.makeText(ReceiverActivity.this, "Ok!", Toast.
                LENGTH_LONG).show();
79      }
80      private void notopenTheDoor(){
81          Toast.makeText(ReceiverActivity.this, "Intruder!!",
                Toast.LENGTH_LONG).show();
82      }
83
84      private void receiveMessageFromDevice(Intent intent) {
85          String action = intent.getAction();
86
87          if (NfcAdapter.ACTION_NDEF_DISCOVERED.equals(action
                )) {
88              Parcelable[] parcelables = intent.
                    getParcelableArrayExtra(NfcAdapter.
                    EXTRA_NDEF_MESSAGES);
89
90              NdefMessage inNdefMessage = (NdefMessage)
                    parcelables[0];
91              NdefRecord[] inNdefRecords = inNdefMessage.
                    getRecords();
92              NdefRecord ndefRecord_0 = inNdefRecords[0];
93
94              String inMessage = new String(ndefRecord_0.
                    getPayload());
95              this.tvIncomingMessage.setText(inMessage);
96
97              if(map.get(inMessage.substring(0,16))!=null) {
98                  Date d = new Date();
99                  String inTime = decryptTimestamp(map.get(
                        inMessage.substring(0,16)), inMessage.
                        substring(17));
100
101                 if(inTime.equals(DateFormat.format("yyyy-MM
                        -dd hh:mm", d.getTime()).toString())){
102                     openTheDoor();
103                 }
104                 notopenTheDoor();
105             }
106
107         }
108     }
109
110
```

```
111     public void enableForegroundDispatch(AppCompatActivity
            activity, NfcAdapter adapter) {
112
113         final Intent intent = new Intent(activity.
                getApplicationContext(), activity.getClass());
114         intent.setFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP);
115
116
117         final PendingIntent pendingIntent = PendingIntent.
                getActivity(activity.getApplicationContext(),
                0, intent, 0);
118
119         IntentFilter[] filters = new IntentFilter[1];
120         String[][] techList = new String[][]{};
121
122         filters[0] = new IntentFilter();
123         filters[0].addAction(NfcAdapter.
                ACTION_NDEF_DISCOVERED);
124         filters[0].addCategory(Intent.CATEGORY_DEFAULT);
125         try {
126             filters[0].addDataType(MIME_TEXT_PLAIN);
127         } catch (IntentFilter.MalformedMimeTypeException ex
                ) {
128             throw new RuntimeException("Check your MIME
                    type");
129         }
130
131         adapter.enableForegroundDispatch(activity,
                pendingIntent, filters, techList);
132     }
133
134     public void disableForegroundDispatch(final
            AppCompatActivity activity, NfcAdapter adapter) {
135         adapter.disableForegroundDispatch(activity);
136     }
137 }
```

References

[1] Eduardo Fernandez-Buglioni *Security Patterns in Practice*. John Wiley & Sons, 25 giu 2013.

[2] Developer: guide,
https://developer.android.com/guide

[3] How to add fingerprint authentication to your Android app,
https://www.androidauthority.com/how-to-add-fingerprint-authentication-to-your-android-app-747304/

[4] Sending files to another device with NFC,
https://developer.android.com/training/beam-files/send-files.html

[5] Receiving Files from Another Device with NFC,
https://developer.android.com/training/beam-files/receive-files.html

[6] Host-based card emulation overview,
https://developer.android.com/guide/topics/connectivity/nfc/hce

[7] NFC basics,
https://developer.android.com/guide/topics/connectivity/nfc/nfc