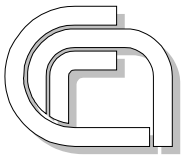


Consiglio Nazionale delle Ricerche

Certificati per Impronte Digitali Memorizzati su Smartcard

L. Bechelli, S. Bistarelli and A. Vaccarelli

{Luca.Bechelli,Stefano.Bistarelli,Anna.Vaccarelli}@iit.cnr.it

Istituto di Informatica e Telematica

28 maggio 2002

IIT TR-09/2002

Technical Report

Istituto di Informatica e Telematica

Sommario

1 Scopo

Il presente documento descrive una specifica per il formato di salvataggio, conservazione, trasmissione e trattamento di informazioni biometriche, ed in particolare, di templates di impronte digitali.

2 Definizioni

- template: l'evidenza informatica atta a rappresentare le caratteristiche distintive di una informazione biometrica (p.e. impronta) digitalizzata mediante un processo di acquisizione
- processo di acquisizione: la procedura mediante la quale l'utente fornisce ad un sistema di acquisizione una propria caratteristica biometrica
- Sistema di acquisizione: una copia di una caratteristica biometrica dell'individuo, detta *istanza attuale* o *istanza corrente*, viene acquisita in forma analogica mediante un sensore di acquisizione. Un convertitore analogico/digitale converte i segnali acquisiti e li inoltra a un elaboratore responsabile delle operazioni di confronto e di controllo del sistema.
- Caratteristica biometrica: possono essere di varia natura e sono generalmente suddivise, come riportato in figura 1, in *fisiologiche* (impronta digitale, volto, mano, retina, iride, DNA, ...) e *comportamentali* (voce, calligrafia, stile di battitura, ...).

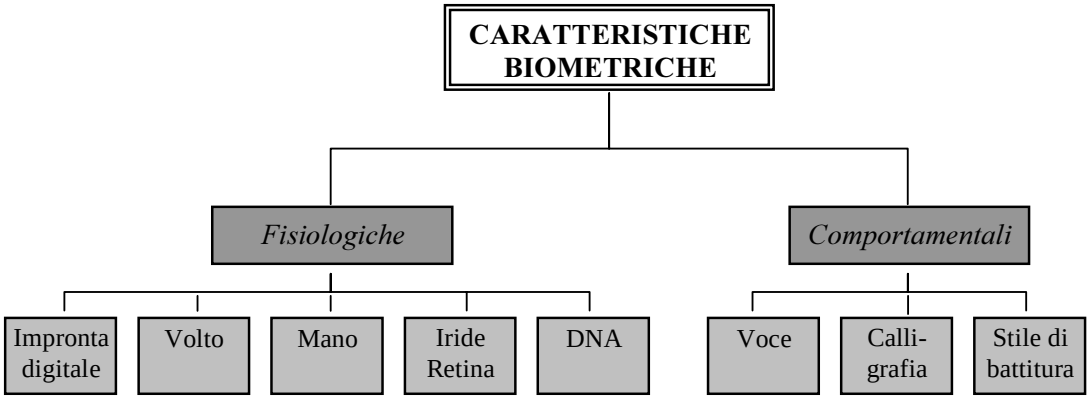


Fig. 1. Le principali caratteristiche biometriche.

3 Proprietà dei template

Il formato nel quale sono conservate le informazioni biometriche deve soddisfare i seguenti requisiti:

- integrità
- autenticità
- portabilità
- privacy

In più, il formato deve specificare:

- a quali caratteristiche dell'utente le informazioni biometriche sono associate (nel caso in cui le informazioni biometriche siano impronte digitali: a quale dito e di quale mano l'impronta fa riferimento)
- il formato di acquisizione
- la funzione di match ritenuta sufficiente per garantire la corretta identificazione
- il grado di precisione da utilizzare durante il match (il bilanciamento tra falsi positivi e falsi negativi ritenuto sufficiente a garantire che il match sia valido)
- il formato del template (se diverso dal formato di acquisizione)

Un requisito preferenziale nella scelta del formato è la possibilità di essere integrato all'interno di applicazioni che utilizzano tecnologie di autenticazione basate su certificati X.509.

3.1 Integrità, autenticità, portabilità e privatezza dei templates biometrici

Il presente capitolo individua un insieme di soluzioni che consentono di specificare un formato per il salvataggio dei templates biometrici che permette di rispettare i requisiti di integrità, autenticità, portabilità e privacy dei dati.

3.1.1 Integrità

L'integrità dei templates è necessaria per garantire che l'impronta non venga modificata (intenzionalmente o accidentalmente). Questa può essere garantita mediante applicazione di una funzione hash ai file che li contengono. L'uso di funzioni hash di tipo crittografico, in letteratura denominati secure hash algorithms, permette di verificare senza ombra di dubbio che i dati a cui l'hash fa riferimento non siano stati alterati. Ad ulteriore garanzia, l'hash può essere sottoposto a processi di cifratura asimmetrica, per garantire che non sia stato volutamente alterato assieme ai dati dell'impronta.
L'uso di meccanismi di controllo dell'integrità delle impronte biometriche ne permette la trasmissione mediante connessioni inaffidabili o basate su linee non sicure, senza dover necessariamente introdurre ulteriori controlli.

3.1.2 Autenticità

L'autenticità dei templates è necessaria per garantire:

1. la corretta associazione tra persona fisica e templates
2. la corretta associazione tra templates ed eventuali informazioni aggiuntive

L'associazione tra persona fisica e templates è necessaria in tutti i contesti dove una entità ha necessità di validare il match tra una impronta acquisita mediante *livescan* e una copia locale: tale copia, può essere ottenuta mediante una procedura di registrazione, ed in tal caso l'utente deve fornire la propria impronta ad ogni entità presso la quale deve autenticarsi.
In alternativa, il template biometrico può essere acquisito da una entità *trusted* alla quale i vari servizi di autenticazione possono rivolgersi per richiedere la o le impronte biometriche associate ad un utente: in tal caso è possibile ridurre al minimo il numero di interazioni che richiedono la presenza fisica degli utilizzatori dei servizi.
In tal caso, una entità di fiducia tanto per l'utente che per l'erogatore dei servizi che necessitano di autenticazione biometrica, deve ovviare alle procedure di acquisizione dei templates ed alla distribuzione degli stessi; in tal caso, questa entità super partes deve :

- assicurare l'integrità dei templates
- garantire l'associazione certa tra persona fisica e template
- garantire l'autenticità dei templates
- divulgare i templates mediante opportune policy riconosciute ed approvate dall'utente finale, tenendo conto delle eventuali problematiche legate alla legislazione in materia di privacy

Assicurare l'integrità dei templates, l'autenticità e l'associazione tra persona fisica e impronta biometrica acquisita si può ottenere mediante l'uso di certificati digitali nei quali il template sia inserito come estensione firmata. In un certificato digitale X.509, infatti, è possibile inserire sia un certo numero di dati anagrafici dell'utente, sia informazioni di tipo eterogeneo.

3.1.3 Portabilità

In merito al requisito di portabilità, è opportuno prevedere che i templates biometrici siano utilizzabili all'interno di applicazioni esistenti che sovrintendono ai processi di autenticazione, identificazione e autorizzazione.
Per maggiore specificità, è opportuno garantire, oltre alla fruibilità stessa dell'informazione biometrica, che tali sistemi di sicurezza siano capaci di verificare il rispetto dei requisiti sopra indicati di integrità e di autenticità, così da garantire con certezza che le impronte stesse non siano state alterate, e la associazione certa tra impronte e utenti.

Per questo motivo, il problema della portabilità può essere ricondotto alla individuazione di una tecnologia che sia utilizzabile nel maggior numero di sistemi di autenticazione, alla quale sostituire, tutte o in parte, le informazioni usate (per esempio la password) con il template biometrico corredato da caratteristiche di autenticità e integrità.
Una possibile soluzione, ad esempio, potrebbe essere quella di sostituire la password con l'impronta biometrica in tutte le procedure di autenticazione basate su username e password, nonché di apportare modificazioni al processo di validazione appoggiandolo ad un meccanismo di match tra impronta acquisita e template salvato localmente: in questo modo, l'impatto sui protocolli di comunicazione risulta minimale ma pesanti modificazioni sono apportate ai client ed ai sistemi server.
Per questo motivo, la soluzione tecnologica tradizionale, alla quale il formato dei dati biometrici dovrebbe uniformarsi, non dovrebbe basarsi esclusivamente sul mantenimento delle interfacce delle chiamate delle funzioni di autenticazione, ma dovrebbe individuare un processo di trasmissione e di validazione delle informazioni preesistente. In questo caso, l'informazione biometrica potrebbe essere un *add-on* ad una informazione di autenticazione già esistente (nel

caso in cui contribuisca ad aggiungere un ulteriore “authentication factor”) oppure avere un formato identico o compatibile a quello previsto per altri sistemi.
In tal senso, potrebbe essere possibile inserire il template biometrico in un certificato X.509 come una estensione, per poterlo poi utilizzare in tutti i contesti dove questa tecnologia è disponibile, con costi relativamente contenuti.

3.1.4 Privacy degli utenti

Nell’indicare i requisiti che il formato dei dati biometrici dovrà rispettare, è necessario individuare anche l’impatto che la tecnologia adottata potrebbe avere sugli utenti finali.
Le informazioni biometriche, pur essendo caratteristiche individuali facilmente acquisibili, sono *proprietà* degli utenti, i quali potrebbero manifestare perplessità o richiedere speciali politiche di gestione delle informazioni biometriche stesse.
In particolare, il formato definito nel presente documento, deve poter essere associato ad una policy che consenta agli utenti di manifestare esplicitamente le autorizzazioni alla diffusione, al trattamento ed alla eliminazione dell’informazione stessa, in modo tale da non ledere il possesso sui dati stessi.
Per questo motivo, l’informazione biometrica deve essere utilizzata solo quando strettamente necessario, e quindi **deve essere separata fisicamente dalle informazioni identificative che l’utente utilizza per autenticare se stesso in altri contesti**.
Allo stesso modo, deve essere mantenuta una relazione univoca tra i dati identificativi (come ad esempio, le informazioni anagrafiche) e l’impronta dell’utente, in modo che possa essere ricondotta al possessore. In questo modo, l’utente ha il potere di decidere *se*, *come* e *quando* diffondere le proprie informazioni biometriche.
Soluzioni possibili possona essere:

- quella di dotare gli utenti di due certificati digitali, dei quali uno solo contenente l’impronta, così da non obbligare l’utente a diffondere l’impronta ogni qualvolta utilizzi la propria coppia di chiavi di cifratura e firma.
- Quella di prevedere la cifratura del template con una shared key condivisa tra l’utente ed un servizio di autenticazione, o l’uso di funzioni hash. Nel primo caso (shared key), l’utente deve necessariamente registrarsi presso un servizio remoto o locale, fornendo la propria impronta biometrica e accordando una chiave condivisa che il sistema potrà utilizzare per accedere al template in chiaro. Nel secondo caso (hash del template), l’impronta biometrica potrebbe essere conservata su un servizio di Directory remoto, mentre un hash dello stesso potrebbe essere conservato dall’utente per garantirne l’autenticità.

Entrambe le soluzioni, tuttavia, restringono il campo di utilizzo dei templates, e quindi ledono gli altri requisiti sopra indicati. La soluzione che adotteremo sarà quindi di avere un certificato di attributo come contenitore dell’impronta, strettamente correlato con un certificato associato all’utente stesso.

4 Certificati di attributo

Il presente paragrafo descrive il formato dei certificati di attributo conformi allo standard X.509 v3.
Il certificato di attributo non contiene una chiave pubblica (lo scopo non è infatti garantire l’associazione utente-coppia di chiavi asimmetriche) ma garantisce il legame tra l’identità digitale e informazioni particolari che possono essere associate all’utente (il ruolo, l’appartenenza ad un gruppo, associazione o corporazione, le proprietà di autenticazione in un

particolare sistema informativo, informazioni biometriche, …). Tipicamente questo tipo di informazioni sono invarianti nel tempo o, al contrario, volubili al punto tale da non essere garantite per il periodo di validità di un certificato personale.

4.1 La sintassi ASN.1

Abstract Syntax Notation 1 è una notazione standard per la descrizione formale di dati strutturati che possono essere trasmessi attraverso piattaforme e supporti di comunicazioni diversi, indipendentemente dal linguaggio con il quale sono implementate le applicazioni.
Standardizzato originariamente per la specifica dei protocolli di comunicazione OSI, ASN.1 è oggi utilizzata per la definizione delle strutture dati e delle informazioni negli standard internazionali (Internet Standard, ITU e ISO).
La sintassi ASN.1 è stata oggetto di standardizzazione ad opera del CCITT (*International Telegraph and Telephone Consultative Committee*, adesso chiamato ITU-T, *International Telecommunication Union - Telecommunication Standardization Sector*) sotto il nome di X.409 Recommedation. ISO, successivamente, ha adottato questa notazione suddividendola in due standard: Abstract Syntax Notation (ASN) (parte 1) e Basic Encoding Rules (BER) (parte 2), che definisce il formato per il salvataggio e la trasmissione di informazioni strutturate in formato binario.
Ad oggi, lo standard di riferimento per la sintassi ASN.1 è descritto in ISO 8824 e 8825, a frutto di una collaborazione tra ISO e CCITT.

4.1.1 Utilizzo degli Object ID

Gli `Object ID` (`OID`) costituiscono gli identificatori degli oggetti rappresentati in sintassi ASN.1
Un OID è una lista univoca di numeri interi, separata mediante il carattere “.”, dove ciascun elemento della lista è definito come “object identifier component”.
La specifica del formato BER prevede, infatti, che la traduzione delle strutture ASN.1 in informazioni binarie sia applicata mediante conversione del nome delle strutture dati nei relativi OID: l’interprete ASN.1 deve quindi essere a conoscenza dei codici identificativi per descrivere in modo human readable i dati contenuti.
Un OID deve essere costituito da una lista di almeno 2 elementi, ricavata da una gerarchia logica ad albero di elementi il cui path (l’OID completo) identifica universalmente un oggetto.
Per poter garantire l’univocità di un Object ID è necessario un ente preposto all’assegnazione degli identificatori, e questo ruolo è demandato alla ICANN (Internet Corporation for Assigned Names and Number).
La composizione degli elementi di una lista OID è tale da consentire di ricavare una descrizione più o meno dettagliata delle caratteristiche di un oggetto.
Per esempio, l’OID che identifica la funzione di cifratura `DES-EDE3-CBC`, ovvero triple DES utilizzato in modalità Encryption-Decryption-Encryption in Code Block Chaining mode è identificato mediante OID `1.2.840.113549.3.7`:

- `1` - ISO assigned OIDs : root del directory IANA degli OID ASN.1
- `1.2` - ISO member body : indica che l’OID è stato assegnato a seguito della pubblicazione di una norma ISO
- `1.2.840` - USA : identificatore ISO della nazionalità di appartenenza

- `1.2.840.113549` - RSADSI : algoritmi e strutture dati definiti in RFC 1423 che concernono indicazioni e recommendation della RSA Data Security Inc.
- `1.2.840.113549.3` - Encryption Algorithms : algoritmi di cifratura

Nel medesimo path `1.2.840.113549.3` sono definiti anche gli algoritmi:

- `1.2.840.113549.3.2 - RC2-CBC`
- `1.2.840.113549.3.4 - RC4 Encryption Algorithm`
- `1.2.840.113549.3.7 - DES-EDE3-CBC`
- `1.2.840.113549.3.9 - RC5 CBC Pad`
- `1.2.840.113549.3.10 - id-desCDMF`

4.2 Descrizione della struttura ASN.1

Il certificato di attributo è così definito, in sintassi ASN.1:

```
AttributeCertificate ::= SEQUENCE {
   acinfo               AttributeCertificateInfo,
   signatureAlgorithm   AlgorithmIdentifier,
   signatureValue       BIT STRING
 }
```

dove:

```
AttributeCertificateInfo ::= SEQUENCE {
   version                 AttCertVersion DEFAULT v1,
   owner                   Owner,
   issuer                  AttCertIssuer,
   signature               AlgorithmIdentifier,
   serialNumber            CertificateSerialNumber,
   attrCertValidityPeriod  AttCertValidityPeriod,
   attributes              SEQUENCE OF Attribute,
   issuerUniqueID          UniqueIdentifier OPTIONAL,
   extensions              Extensions OPTIONAL
 }
```

La `SEQUENCE` ASN.1 `AttributeCertificateInfo` contiene i dati del certificato di attributo che si desidera legare all'utente.
In particolare,

- `owner`: questo campo può indicare l'identità dell'utente al quale il certificato si riferisce, oppure l'identificativo univoco di un suo certificato digitale.
- `issuer`: la Attribute Authority che ha emesso il certificato di attributo, garantendo per l'autenticità delle informazioni ivi contenute
- `signature`: l'algoritmo utilizzato per la firma dei dati
- `serialNumber`: il numero seriale del certificato di attributo
- `attrCertValidityPeriod`: l'intervallo di tempo nel quale il certificato ha valore. Questo campo indica la data di inizio validità e di scadenza. Nel caso

in cui le informazioni non siano più valevoli durante questo periodo, il certificato dovrà essere revocato

- `extensions`: le estensioni sono un meccanismo per inserire informazioni aggiuntive nel certificato senza alterarne la struttura ASN.1 .
- `attributes`: consente di inserire un elenco di chiavi-valori che indicano attributi associati al titolare del certificato.

I campi `dell'AttributeCertificate` denominati "`signatureAlgorthm`" e "`signatureValue`" specificano rispettivamente l'algoritmo e il valore binario della firma apposta dalla Attribute Authority sul certificato. In questo modo, l'Attribute Authority garantisce l'associazione utente - template biometrico.

4.2.1 Estensioni

Le estensioni costituiscono un meccanismo per estendere il contenuto del certificato con strutture ASN.1
Le estensioni sono definite in ITU-T X.509v3 e spossono essere standard o proprietarie. A ciascuna di esse può essere definito l'attributo `critical` che specifica l'obbligo di interpretarne il contenuto da parte di un sistema di elaborazione.
La definizione ASN.1 delle estensioni X.509v3 è la seguente:

```
Extensions  ::=  SEQUENCE SIZE (1..MAX) OF Extension

 Extension  ::=  SEQUENCE  {
   extnID      OBJECT IDENTIFIER,
   critical    BOOLEAN DEFAULT FALSE,
   extnValue   OCTET STRING  }
```

Il campo `extnID` contiene l'Object ID associato all'estensione, il cui contenuto è specificato nel campo `extnValue`, che può assumere qualunque valore consentito dalla sintassi ASN.1.
La sequenza di `Extensions` non è definita superiormente, e per questo motivo può essere vuota oppure può contenere un numero arbitrario di `Extension`.

4.2.2 Attributi

Gli attributi di un certificato X.509 sono definiti come una lista non limitata di chiavi-valori :

```
AttributeTypeAndValue ::= SEQUENCE {
   type     AttributeType,
   value    AttributeValue }


 AttributeType ::= OBJECT IDENTIFIER


 AttributeValue ::= ANY   -- Directory string type --
```

```
DirectoryString ::= CHOICE {
   teletexString           TeletexString (SIZE (1..maxSize),
   printableString         PrintableString (SIZE (1..maxSize)),
   universalString         UniversalString (SIZE (1..maxSize)),
   bmpString               BMPString (SIZE(1..maxSIZE))
}
```

quindi, è possibile associare ad un attributo un valore rappresentato in uno dei formati definiti dal tipo `DirectoryString`, oppure è possibile includere un elemento di struttura ASN.1, come una `SEQUENCE`, che consente di definire una struttura dati di profondità variabile.

4.3 Attributi specifici per autenticazione

Il certificato di attributo, come per altro specificato al paragrafo §4.2, può avere associati un numero non limitato di attributi ed estensioni, sia che esse siano specificate in X.509, sia definite in altri standard.
La possibilità di definire nuove estensioni e attributi per scopi specifici, consente di ampliare le caratteristiche ed il numero di informazioni del certificato di attributo senza alterarne la struttura e quindi senza che sia necessario emettere nuove versioni dello standard X.509.
Nel caso di utilizzo di certificati di attributo per scopi di autenticazione, è possibile includere nello stesso certificato le informazioni necessarie al supporto di meccanismi di autenticazione con le relative policy di gestione.

4.3.1 L'attributo AuthenticationInfo

ECMA- 219, "Authentication and Privilege Attribute Security Application with related key destruction functions", definisce un attributo (`authenticationInfo`) che può essere utilizzato per rappresentare le informazioni di autenticazione, i formati di salvataggio, gli algoritmi di elaborazione e di verifica.

La struttura di questo attributo descritta in sintassi ASN.1 :

```
authenticationInfo ATTRIBUTE::={ WITH SYNTAX
AuthenticationInfo, ID id- at- TBD}

AuthenticationInfo::= SEQUENCE{
authenticationMethod AuthenticationMethod, -- defined in ECMA. 219
exchangeAI  SEQUENCE SIZE (1...MAX) OF AuthMparm, -- the data, as defined in
ECMA. 219
}
```

dove `authenticationMethod` indica quindi il sistema di autenticazione utilizzato:

```
SelectedAuthenticationMethods {
      iso(1) identified-organisation(3) icd-ecma(0012) standard(0)
      apa(219) modules(1) selectedAuthenticationMethods(5) }

DEFINITIONS ::=
      BEGIN
      -- exports everything
```

```
      IMPORTS
      id-am
      FROM UsefulDefinitions {
            iso(1) identified-organisation(3) icd-ecma(0012) standard(0)
            apa(219)
            modules(1) usefulDefinitions(1) } ;

-- End of imports
-- Object Identifiers for the authentication methods
password ::= { id-am 0 }
encryptedPassword ::= { id-am 1 }
encryptedRepPwd ::= { id-am 2 }
hashedPassword ::= { id-am 3 }
hashedRepPwd ::= { id-am 4 }
passwordAsKey ::= { id-am 5 }
passiveToken ::= { id-am 6 }
activeToken ::= { id-am 7 }
voicePrint ::= { id-am 8 }
signature ::= { id-am 9 }
fingerprint ::= { id-am 10 }
retinaPattern ::= { id-am 11 }
userCertificate ::= { id-am 12 }
zeroKnowledgeMeth1 ::= { id-am 13 }
location ::= { id-am 14 }
passwordBits ::= { id-am 15 }
symmetricCrypto ::= { id-am 16 }
asymmetricCrypto ::= { id-am 17 }
applicationName ::= { id-am 18 }


END -- of selectedAuthenticationMethods
```

`exchangeAI` contiene i dati necessari per l'applicazione del `method`, ovvero fornisce le informazioni di autenticazione stesse, politiche per il loro trattamento e ulteriori informazioni dipendenti dal contesto:

```
AuthMparm ::= CHOICE{
printableValue [0] PrintableString,
integerValue [1] INTEGER,
octetValue [2] OCTET STRING,
bitStringValue [3] BIT STRING,
otherValue [4] ANY} -- defined by authenticationMethod
```

In particolare, a seconda del `method` di autenticazione definito, ECMA 2.19 fornisce alcune implementazioni specifiche per il contenuto di `exchangeAI` :

```
-- Exchange AI Constructs
PasswordAI ::= PrintableString

EncPasswordAI ::= SEQUENCE {
      encPassword [0] BIT STRING,
      encAlgId [1] AlgorithmIdentifier OPTIONAL,
      keyIdentifier [2] INTEGER OPTIONAL}

EncRepPwdAI ::= SEQUENCE {
```

```
      encPassword [0] BIT STRING,
      encAlgId [1] AlgorithmIdentifier OPTIONAL,
      keyIdentifier [2] INTEGER OPTIONAL,
      uniqueNumber [3] UniqueNumber} -- defined in 4.1

HashedPasswordAI ::= SEQUENCE {
      hashedPassword [0] BIT STRING,
      hashParameter [1] OCTET STRING OPTIONAL,
      hashAlgId [2] AlgorithmIdentifier OPTIONAL}

HashedRepPwdAI ::= SEQUENCE {
      hashedPassword [0] BIT STRING,
      hashParameter [1] OCTET STRING OPTIONAL,
      uniqueNumber [2] UniqueNumber,-- defined in 4.1
      hashAlgId [3] AlgorithmIdentifier OPTIONAL}

DeviceAI ::= BIT STRING

CharacteristicAI ::= BIT STRING

DirectoryAI ::= SEQUENCE {
      certificate [0] Certificate,
      token [1] Token}

ZeroKnowledgeAI ::= CHOICE {
      challenge [0] BIT STRING,
      response [1] BIT STRING}

LocationAI ::= BIT STRING -- some form of address

ApplicationNameAI ::= PrintableString
```

4.4 Ipotesi per la realizzazione di una Attribute Authority

La Attribute Authority è un servizio di certificazione finalizzato all'emissione di certificati di attributo X.509 agli utenti finali.
Generalmente, la AA è parte integrante di una Certification Authority, soprattutto se questa ha titolarità sulle informazioni di attributo che devono essere associate all'utente finale, ma in taluni casi può trattarsi di una entità separata, e comunque ritenuta *fidata*, che ha l'autorità di assegnare specifiche informazioni all'utente, quali il ruolo, l'appartenenza ad un ordine o ad una associazione o altre (come informazioni di autenticazione).
La struttura di una Attribute Authority corrisponde a quella di una PKI standard, ed è costituita almeno da:

- Attribute Authority: il sistema sul quale è installata la coppia di chiavi ed il certificato con i quali sono firmati tutti i certificati di attributo generati
- Registration Authority: i sistemi verso i quali l'utente si identifica e richiede il certificato di attributo
- Certificate Server: i sistemi di pubblicazione del certificato della AA, che opzionalmente possono ospitare un directory server per la consultazione dei certificati di attributo emessi e la lista dei certificati revocati

Le principali differenze tra una Attribute Auhtority ed una Certification Authority sono costituite dalle procedure di identificazionee registrazione degli utenti e di enrolment dei certificati.

4.4.1 Identificazione utenti e richiesta certificato

La procedura di identificazione, come avviene per le Certification Authorities, è descritta nei Certificate Practice Statement della Attribute Authority, e può variare in base al livello di sicurezza che suddetta autorità vuole garantire.
In particolare, la caratteristica principale che distingue la Attribute Authority, è la possibilità di sostituire le procedure che necessitano della presenza fisica degli utenti con altre completamente automatiche basate su flussi elettronici, senza compromettere il livello di sicurezza dell'identificazione.
L'utilizzo di certificati di attributo, infatti, è vincolato nella maggior parte dei casi dal possesso di un certificato digitale di autenticazione o di sottoscrizione emessi da una Autorità di Certificazione, la quale ha già provveduto a identificare gli utenti mediante una apposita procedura.
In questo caso, l'utente finale non ha necessità di presentarsi fisicamente allo sportello della Attribute Authority, ma può firmare digitalmente un documento di richiesta nel quale sono indicati tutti i dati che l'Autorità ritiene necessari per l'emissione del certificato.
In tal caso, tuttavia, è opportuno che la CA e la AA siano in accordo in merito al livello di sicurezza delle procedure di identificazione, ed in particolare la Attribute Authority deve considerare sufficientemente fidata la Certification Authority da non ritenere necessaria la ripetizione della procedura di identificazione degli utenti.
Il livello di fiducia tra due Autorità di Certificazione o di Attributo può essere stabilito mediante una comparazione dei Certificate Practice Statement, la quale dovrà stabilire l'equipollenza tra:

- procedure di certificazione
- sistemi di protezione della chiave privata della CA
- livello di sicurezza degli strumenti crittografici e di gestione utilizzati
- valutazioni e certificazioni di qualità e sicurezza
- sistemi di protezione fisica dei locali che ospitano la documentazione cartacea ed elettronica ed i sistemi di elaborazione
- livelli di servizio
- esperienza, nel settore, del personale impiegato

ed altri eventuali misure di protezione e di tipo procedurale.
In alcuni casi, tuttavia, non è possibile evitare che l'utente sia coinvolto fisicamente nelle procedure di identificazione e richiesta certificato: nei casi in cui l'Attribute Authority debba autenticare, certificare o garantire per alcune informazioni che non sono in suo possesso o che non possono essere trasmesse per via telematica senza perdita di validità (come ad esempio un attestato, un diploma o altro) è infatti necessario che l'utente consegni queste informazioni *brevi manu* fornendo prove ulteriori della associazione tra la sua persona fisica o giuridica e queste informazioni.
Analogamente, nel caso in cui le informazioni che devono essere incluse nel certificato di attributo siano di tipo biometrico, e, nello specifico, finalizzate all'autenticazione, è opportuno che la Attribute Authority si faccia carico delle procedure di acquisizione dei template richiedendo quindi la presenza fisica dell'utente. In questo modo la responsabilità per la veridicità dei dati biometrici per autenticazione può essere attribuita alla Attreibute Authority

stessa, con maggiore garanzia per i gestori di sistemi di servizi che richiedono accessi protetti e autenticati.

4.4.2 Enrollment dei certificati

Le procedure di enrolment dei certificati di attributo, dal punto di vista tecnologico, possono essere descritte prendendo in considerazione due contesti diversi:

- La Attribute Authority e la Certification Authority costituiscono una unica Public Key Infrastructure
- La Attribute Authority è una terza parte rispetto all'utente ed alla Certification Authority presso la quale questi si è rivolto per ottenere il proprio certificato

Nel primo caso, ovvero qualora la Attribute Authority e la CA costituiscano una unica PKI, entrambe possono condividere le procedure di identificazione e richiesta certificato, che possono essere quindi svolte contestualmente.
Questa soluzione impone che il certificato di attributo sia associato al certificato di autenticazione/sottoscrizione, condividendone i tempi di inizio validità e scadenza, nonché l'eventualità della revoca: in tal caso, per ciascun certificato di autenticazione/sottoscrizione emesso o rinnovato, deve essere emesso il rispettivo certificato di attributo.
In questa situazione, l'utente che intende ottenere un certificato di autenticazione/sottoscrizione ed un certificato di attributo, deve fornire le proprie credenziali di identificazione assieme alle informazioni che desidera siano incluse nel proprio certificato di attributo.
Una volta ottenuta l'approvazione formale della richiesta di certificazione da parte della CA/AA, egli dovrà sottomettere le medesime informazioni mediante utilizzo di un messaggio PKCS#10 `CertificationRequest`:

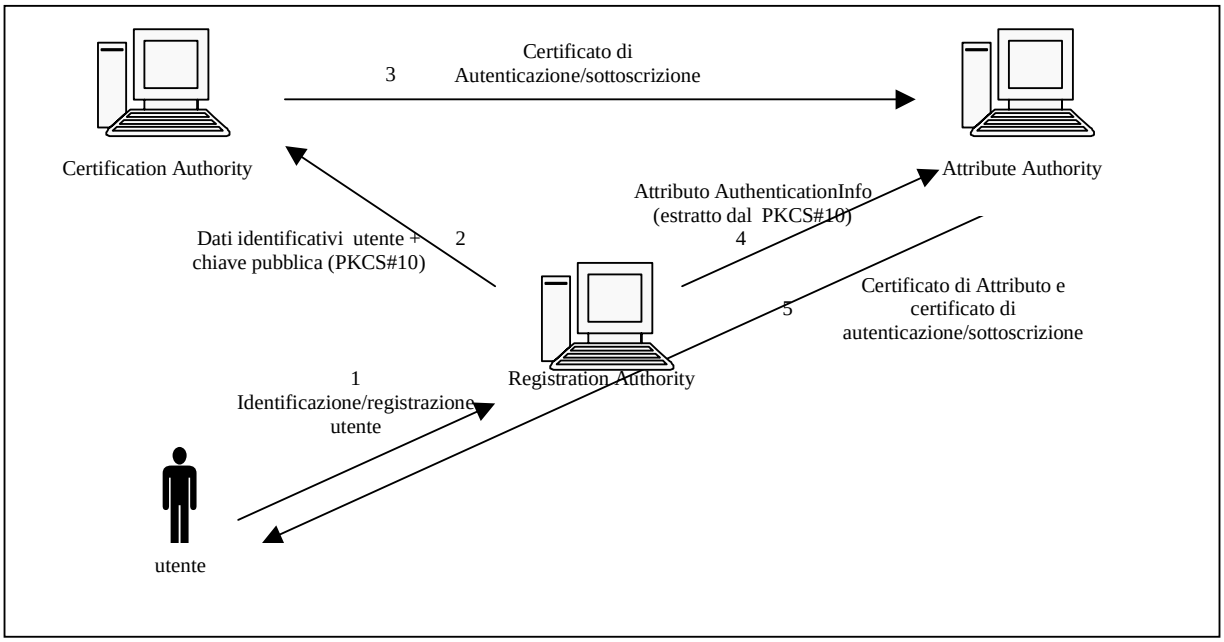
```
CertificationRequest ::= SEQUENCE
{
CertificationRequestInfo CertificationRequestInfo, signatureAlgorithm
SignatureAlgorithmIdentifier,
signature Signature
}
```

L'oggetto `CertificationRequestInfo` è così strutturato:

```
CertificationRequestInfo ::= SEQUENCE {
version Version, -- INTEGER numero di versione
subject Name, -- distinguished name dell'entità che fa richiesta del
certificato
subjectPublicKeyInfo SubjectPublicKeyInfo, -- informazioni riguardanti la
chiave pubblica
attributes [0] IMPLICIT Attributes – informazioni addizionali sul
richiedente.
}
```

Il messaggio `CertificationRequest` è costituito dai dati del richiedente (le stesse informazioni che la CA dovrà utilizzare per generare il Distinguished Name dell'utente stesso), dalla sua chiave pubblica e da una serie di attributi opzionali, la cui presenza è stabilita dai Certificate

Practice Statement della CA/AA. L'intero messaggio di richiesta è firmato con la chiave privata dell'utente, al fine di fornire la *prova di possesso* della coppia di chiavi asimmetriche, necessaria per evitare che un'entità malevola possa richiedere un certificato fornendo la chiave pubblica di un altro utente.
Questo messaggio, nel caso in cui la medesima richiesta PKCS#10 sia utilizzata dalla Attribute Authority della PKI, può contenere un numero non limitato di informazioni addizionali (`attributes`), nei quali possono essere incluse le informazioni che devono essere incluse nel certificato di attributo.
Nel caso in cui la Attribute Authority sia demandata all'emissione di certificati di attributo per autenticazione, il messaggi `CertificationRequest` o potrebbe contenere una istanza dell'attributo `AuthenticationInfo`, come descritto in §4.3.1.



Nel caso in cui la Attribute Authority sia separata completamente dalla Certification Authority, oppure siano previste due procedure diverse per l'emissione dei certificati di autenticazione/sottoscrizione e di attributo, non sono attualmente disponibili standard per lo scambio di messaggi durante la procedura di enrolment tra l'utente e la Attribute Authority, cosa che può essere motivata con le seguenti considerazioni:

- La Attribute Authority può essere un servizio di un ente o di una struttura che ha titolo per assegnare agli utenti le informazioni che essi richiedono siano incluse in un certificato di attributo, e che ha a disposizione in appositi archivi: è questo il caso delle associazioni di categoria, delle corporazioni, degli albi professionali
- La richiesta di certificati attributo può essere fatta, alla Attribute Authority, da una entità che ha titolo per assegnare agli utenti le informazioni che devono essere inserite nel certificato di attributo, in vece degli utenti stessi, i quali forniranno i propri certificati di autenticazione/sottoscrizione in una successiva fase, direttamente alla Attribute Authority
- Le procedure di richiesta di certificato alla Attribute Authority, in taluni casi, possono essere inviate per via telematica mediante semplici documenti firmati digitalmente dal

richiedente con il proprio certificato di autenticazione/sottoscrizione, con le dovute misure cautelative descritte nei Certificate Practice Statement della Attribute Authority come descritto in §4.4.1.

4.5 Informazioni per l'autenticazione

Con particolare riferimento a ECMA- 219, "Authentication and Privilege Attribute Security Application with related key destruction functions", è possibile specificare nell'attributo `authenticationInfo` di un certificato digitale, le informazioni necessarie per associare all'utente le informazioni biometriche che può utilizzare per i servizi di autenticazione.

In particolare al campo `exchangeAI` è possibile sostituire un nuovo gruppo di informazioni specifiche per i sistemici autenticazione biometrica:

```
biometricInfo BiometricInfo OPTIONAL
```

La particolare istanza dell'attributo AutheticationInfo assume la seguente struttura:

```
AuthenticationInfo::= SEQUENCE{
authenticationMethod AuthenticationMethod, -- defined in ECMA. 219
biometricInfo BiometricInfo OPTIONAL }
```

nei casi in cui `authenticationMethod` assuma uno dei seguenti valori:

```
voicePrint ::= { id-am 8 }
signature ::= { id-am 9 }
fingerprint ::= { id-am 10 }
retinaPattern ::= { id-am 11 }
```

La specifica "*R22 User Authentication Techniques Using Public Key Certificates Part 2: Authentication Information Including Biometrics*" pubblicata dalla NSA, specifica in modo dettagliato il contenuto della `SEQUENCE` ASN.1 definita `BiometricInfo`:

```
BiometricInfo::= SEQUENCE{
processingInfo ProcessingInfo OPTIONAL,
matchingInfo MatchingInfo}
```

la quale descrive i meccanismi di processo dell'informazione biometrica e l'algoritmo di match da utilizzare per la verifica in fase di autenticazione

Descriviamo adesso le singole strutture ASN.1 e le possibili applicazioni nel caso di utilizzo con informazioni biometriche.

4.5.1 ProcessingInfo

Questa struttura ASN.1 è così specificata:

```
ProcessingInfo::=SEQUENCE {
processingID OBJECT IDENTIFIER, -- Registered by implementation
```

```
processingParms AuthMparm OPTIONAL, -- Defined in ECMA.219
processingVersion Version} -- Defined inX.509
```

`PorcessingInfo` descrive l'algoritmo di acquisizione e salvataggio delle informazioni biometriche in un template.
`ProcessingID` contiene l'Object ID assegnato dalla IANA all'algoritmo utilizzato per trasformare l'input in un template. L'indicazione dell'algoritmo consente ad una entità diversa dalla Attribute Authority di *ripetere* il processo di acquisizione per applicare, ad esempio, una funzione di match, oppure per poter interpretare correttamente il formato del template stesso.
`ProcessingParms` permette di associare un numero variabile di parametri opzionali all'algoritmo di acquisizione, qualora necessari. Nel caso di impronte biometriche, potrebbero essere indicate quali dita di quale mano possono essere utilizzate per l'acquisizione, parametri inerenti il livello di dettaglio della acquisizione, oppure il numero di template necessari per garantire una collezione di informazioni sufficienti per permettere la creazione di un template consistente e completo.

L'importanza della presenza, nel certificato, delle informazioni relative alla modalità di acquisizione dei templates, permette ai sistemi di autenticazione di stabilire delle policy sulla qualità delle informazioni di autenticazione. Come avviene per i certificati digitali, che sono spesso suddivisi in classi di sicurezza, sulla base delle policy di identificazione degli utenti utilizzate dalle CA, anche per i dati biometrici potrebbe essere possibile definire il grado di sicurezza di un template sulla base della modalità con la quale è stato acquisito.
I sistemi di autenticazione, quindi, potrebbero vincolare gli accessi ai soli utenti che posseggono un certificato di attributo nel quale l'impronta è stata digitalizzata ed elaborata secondo specifici algoritmi e procedure.

Le informazioni di processo, infine, devono essere rispettate dai sistemi di autenticazione per quanto riguarda l'acquisizione dell'impronta live, per ottenere così il medesimo formato del template con gli stessi requisiti di sicurezza e procedurali.

4.5.1.1 ProcessingParms

Una possibile indicazione di parametri contenuti nel campo `ProcessingParms` potrebbe essere la seguente:

- `resolution`: indicazione della risoluzione da utilizzare nel processo di acquisizione
- `collection`: numero di templates necessari per considerare esaustivo il numero di informazioni ottenute su una specifica impronta
- `templates`: numero di templates che devono essere raccolti (ovvero il numero di impronte digitali che devono essere acquisite)

La sintassi ASN.1 specificata:

```
ProcessingParms::=SEQUENCE SIZE (1...MAX) OF{
      ProcessInfoPolicy SEQUENCE OF{
            ID          OBJECT IDENTIFIER,
            Resolution        OCTET VALUE,
            Collection        INTEGER OPTIONAL,
            Templates         INTEGER  Default 1
```

```
      }
}
```

L'elemento `ProcessInfoPolicy` descrive una politica applicabile a un generico template, identificata da un ID univoco.
A ciascun elemento `ProcessInfoParms` è possibile associare un numero arbitrario di politiche di processo, in modo tale da rendere possibile l'uso di procedure specifiche a seconda del contesto di utilizzo.

4.5.2 MatchingInfo

Il match è la funzione mediante la quale due templates biometrici sono confrontati nella ricerca di similitudini.
Il risultato della funzione è, tipicamente, uno *score* che indica il grado di similitudine tra i due.
La struttura ASN.1 `MatchingInfo` è così rappresentata:

```
MatchingInfo::=SEQUENCE{
matchingID OBJECT IDENTIFIER, -- Defined by implementation
matchingParm AuthMparm OPTIONAL, -- Defined in ECMA.219
matchingVersion Version} -- Defined in X.509
```

Il campo `matchingID` contiene un Object ID assegnato dalla IANA alla funzione di match, mentre nel campo `matchingParm` è possibile specificare un certo numero di parametri opzionali, le cui caratteristiche sono strettamente dipendenti dalla funzione stessa.

In particolare, possibili parametri nel campo `matchingParm`, potrebbero essere l'indicazione dell'Object ID del formato del template (o dei templates) biometrici compatibili con la funzione di match, o lo *score* minimo ritenuto sufficiente perché il match abbia successo.

4.5.2.1 MatchingParm

Definiamo adesso una specifica ASN.1 per il campo `MatchingParm` secondo quanto indicato al paragrafo precedente:

```
MatchingInfoParms::= SEQUENCE{
MatchingPolicy SEQUENCE OF{
      PolicyID    OBJECT IDENTIFIER,
      ProcessingID      SEQUENCE SIZE (1…MAX) OF OBJECT IDENTIFIER,
      MatchingScore     CHOICE {
            SCORE       INTEGER (SIZE (1..100)),
            SCORES      SEQUENCE SIZE (1…ProcessingID.SIZE) OF INTEGER (SIZE
(1..100))},
      }
}
```

`MatchingInfoParms` consente di specificare un numero arbitrario di politiche di processo per le funzioni di match.
Ciascuna policy ha un proprio identificativo unico, e può specificare quali formati di acquisizione ritenere idonei alla sua applicazione.

Per determinare in modo inequivocabile un livello adeguato di sicurezza nella valutazione dello *score*, è possibile, infine, specificare un valore in percentuale valido per ciascun formato di acquisizione, o una indicazione generale per tutti i formati supportati.

5 Sequenza completa dell'attributo AuthenticationInfo per le impronte biometriche

Segue una rappresentazione completa della struttura ASN.1 dell'attributo `AuthenticationInfo` per la conservazione di informazioni biometriche:

```
AuthenticationInfo::= SEQUENCE{
AuthenticationMethod::= fingerprint{ id-am 10 },

biometricInfo SEQUENCE{
        processingInfo SEQUENCE {
                processingID OBJECT IDENTIFIER
                processingParms SEQUENCE SIZE (1...MAX) OF{
                        ProcessInfoPolicy SEQUENCE OF{
                        ID              OBJECT IDENTIFIER,
                        Resolution              OCTET STRING,
                        Collection              INTEGER OPTIONAL,
                        Templates               INTEGER  Default 1
                }

                processingVersion 1
        },
        matchingInfo SEQUENCE{
                matchingID OBJECT IDENTIFIER,
                matchingParm SEQUENCE{
                        MatchingPolicy SEQUENCE OF{
                        PolicyID        OBJECT IDENTIFIER,
                        ProcessingID    SEQUENCE SIZE (1…MAX) OF OBJECT IDENTIFIER,
                        MatchingScore   CHOICE {
                                SCORE           INTEGER (SIZE (1..100)),
                                SCORES  SEQUENCE SIZE (1…ProcessingID.SIZE) OF INTEGER (SIZE
(1..100))},
                        },
                matchingVersion 1
        }

}}
```

6 X.509 v3: informazioni biometriche per scopi generici

Nel caso in cui le informazioni biometriche non siano utilizzate specificatamente come supporto ai sistemi di autenticazione, ma è ritenuta opportuna la loro indicazione nel certificato utente o di attributo, è possibile utilizzare l'estensione V3 standard `BiometricInfo`, definita in PKIX Qualified Certificate Profile.
Questa estensione non è stata prevista per l'inserimento, nel certificato, delle informazioni biometriche, ma offre un meccanismo standard per associare informazioni biometriche al suo possessore, indicando una URI (`sourceDataUri`) dalla quale è possibile acquisirla.

Questa soluzione può risultare utile in altri contesti dove le informazioni biometriche non siano strettamente utilizzate per scopi di autenticazione.
Un ulteriore ambito, potrebbe essere l'uso di questo attributo in accordo con il nuovo standard XMLDSIG, la firma digitale XML, che utilizza le URI come riferimenti assoluti o relativi dei nodi delle strutture dei documenti. In tal caso, potrebbe essere possibile definire un formato XML che preveda, in un documento firmato, la presenza del certificato utente e l'informazione biometrica referenziata tramite l'URI `sourceDataUri`.

```
id-pkix  OBJECT IDENTIFIER  ::=
             { iso(1) identified-organization(3) dod(6) internet(1)
                     security(5) mechanisms(5) pkix(7) }


BiometricSyntax ::= SEQUENCE OF BiometricData
```

`BiometricData` contiene le informazioni relative ad ogni template biometrico associato all'utente.
L'informazione, reperibile ad una apposita URI `sourceDataUri`, è associata in modo incontrovertibile all'utente mediante un hash della stessa (`biometricDataHash`). L'hash è poi firmato assieme alle altre informazioni contenute nel certificato.

```
BiometricData ::= SEQUENCE {
   typeOfBiometricData TypeOfBiometricData,
   hashAlgorithm       AlgorithmIdentifier,
   biometricDataHash   OCTET STRING,
   sourceDataUri       IA5String OPTIONAL }


TypeOfBiometricData ::= CHOICE {
   predefinedBiometricType PredefinedBiometricType,
   biometricDataOid        OBJECT IDENTIFIER }
```

Ai fini delle modalità di acquisizione e di match dei dati biometrici, è possibile specificare il formato dei dati stessi, come reperibili nella URI:

```
 PredefinedBiometricType ::= INTEGER {
  picture(0), handwritten-signature(1)}
  (picture|handwritten-signature)
```

In questo modo, non essendo l'informazione biometrica contenuta nel certificato utente, non vi sono problematiche relative alla privacy degli utenti. Ogni eventuale limitazione di accesso alla vera e propria impronta biometrica, può essere gestita sul servizio di pubblicazione delle impronte al quale la URI `sourceDataUri` fa riferimento.
Inoltre, può essere prevista la pubblicazione dell'impronta come immagine digitalizzata ad alta risoluzione, per facilitare la conversione nel formato nativo di ogni meccanismo di trasformazione in template.
Questa soluzione privilegia quei contesti ove i dati biometrici sono accessibili mediante sistemi eterogenei e con applicazioni di match diverse.

7 Installazione delle impronte biometriche su smartcard

Le impronte biometriche, sia che esse siano contenute in certificati di attributo, sia in qualità di informazioni rappresentate in un formato riconducibile a sequenza di byte, possono essere conservate in smartcard a microprocessore per operazioni di autenticazione verso sistemi esterni o verso la smartcard stessa, in sostituzione del PIN utente.
Le problematiche principali ed i vincoli che una applicazione, che necessita di utilizzare un hardware security module, deve affrontare, sono i seguenti:

- limitazione dello spazio di memoria: tipicamente, gli hardware security modules hanno capacità di memorizzazione limitata a poche migliaia di bytes
- limitazione della capacità di calcolo: la smartcard non offre prestazioni adeguate per l'esecuzione di calcoli complessi
- formato del file system della smartcard: le informazioni, all'interno della smartcard, devono essere conformi ad uno standard specifico onde consentirne la portabilità in ambienti eterogenei
- set di comandi del sistema operativo della carta: al fine di accedere al file system della smartcard, è opportuno scegliere carte a microprocessore che presentino una interfaccia standard e indipendente dal sistema operativo della carta stessa. In tal modo è inoltre possibile utilizzare la smartcard per richiedere l'esecuzione di applicazioni *on board* o impostare funzioni di protezione dei dati in modo standard.

A tal fine, la soluzione individuata prevede l'uso di smartcard con una quantità di memoria di salvataggio pari almeno a 16Kbytes e con un set di comandi conforme alla specifica PKCS#11, la quale permette di consultare ed usufruire del file system della smartcard in modo indipendente da come questo è implementato, e sfruttando le funzionalità crittografiche e di protezione dei dati che lo standard impone.

7.1 Privatezza e caratteristiche di accesso dell'informazione biometrica

La salvaguardia della privatezza delle informazioni biometriche all'interno di token crittografici PKCS#11 può essere ottenuta mediante una adeguata combinazione degli attributi PKCS#11 del file contenente il template.
In PKCS#11, un file all'interno della smartcard è costituito da un elenco di attributi rappresentanti un oggetto di una specifica classe. Le classi ("CLASS") definite dallo standard sono:

- chiavi private (simmetriche o asimmetriche)
- chiavi pubbliche
- certificati digitali("X509Certificate")
- dati generici ("DATA")

Per ciascun oggetto, in generale, è possibile associare un identificativo univoco, l'ownership, il livello di accesso (informazione pubblica o privata) e la possibilità di eliminare o sostituire l'oggetto stesso.
Alcuni oggetti, come le chiavi private generate internamente alla carta, contengono informazioni che sono accessibili soltanto dalle applicazioni interne alla carta stessa.

Nel caso in cui l'informazione biometrica sia rappresentata come un semplice gruppo di bytes, l'oggetto da utilizzare deve avere classe "DATA", poichè si tratta di una generica informazione non strutturata.
Nel caso in cui l'impronta biometrica sia contenuta all'interno di un certificato digitale di attributo, il file avrà come attributo CLASS il valore "X509Certificate".
In ogni caso, per quanto riguarda la protezione dell'accesso al template, questa può variare a seconda dei requisiti imposti dalle politiche della Attribute Auhtority o dell'entità che sovrintende alle procedure di enrollment e di autenticazione.
Possono essere distinti diversi livelli di accesso:

- Accesso pubblico in lettura e scrittura: il template è conservato in un file dichiarato come pubblico e salvato nell'area non protetta del file system, senza che vi sia necessità di alcuna procedura di autenticazione sulla smartcard
- Accesso pubblico in lettura: il template è conservato in un file dichiarato come pubblico con divieto di modificazione, cancellazione o sostituzione se non a seguito di un accesso autenticato sulla smartcard da parte del possessore della carta
- Accesso pubblico in lettura con limitazione di scrittura dell'utente: il template è conservato in file dichiarato come pubblico ed è modificabile soltanto mediante accesso autenticato alla smartcard del Card Operator, ovvero dell'entità che ha emesso la smartcard
- Accesso limitato: l'impronta biometrica è contenuta in un file privato della carta, ovvero è accessibile solo da parte dell'utente a seguito di un logon autenticato (inserimento PIN)

La scelta di una delle soluzioni sopra descritte deve essere valutata anche tenendo conto di una analisi della conformità del prodotto di smartcard utilizzato con lo standard PKCS#11, poiché in molte implementazioni alcune delle funzioni non sono implementate o, addirittura, possono non essere pienamente conformi alla specifica per andare incontro a requisiti di mercato.

7.2 Interazione con smartcard mediante tecnologia ActiveX

È prevista l'integrazione tra la tecnologia offerta dai sistemi di identificazione biometrica di Centro Biometrika con API Java di firma digitale, che offrono la possibilità di utilizzo delle smartcard PKCS#11 per il salvataggio e la conservazione di informazioni eterogenee sia pubbliche che private, nonché l'uso di algoritmi di crittografia asimmetrica per firma e crittografia dei dati.

Il progetto prevede una prima fase in cui sia possibile integrare le due librerie per fornire una soluzione denominata "Template on Card" dove le impronte biometriche saranno conservate in modo sicuro all'interno della smartcard per fornire servizi "dual factor authentication", ovvero basato sul possesso (della smartcard) e sull'apposizione dell'impronta: in quest'implementazione, il template con il quale fare il confronto con l'impronta "live" è prelevato dalla carta, mentre l'operazione di match è realizzata sul sistema locale.

Gli strumenti attualmente disponibili non sono interoperabili, poiché le librerie di firma digitale sono sviluppate e compilate in linguaggio Java™, mentre le API fornite da Centro Biometrika sono realizzate in Microsoft Visual Basic® e C++.

Il primo passo per l'interazione tra dispositivi biometrici e smartcard è quindi quello di consentire una integrazione il più possibile completa e stabile tra il mondo Java e quello Visual Basic/C++. Al fine di consentire questo, due soluzioni sono applicabili:

- individuare strumenti e soluzioni per consentire alle API Java di interagire con le Librerie C++/VB
- individuare strumenti e soluzioni per permettere alle librerie C++/VB di utilizzare oggetti Java

Entrambe le soluzioni sono valide, ad alto livello, ai fini del progetto di integrazione per l'uso di tecnologie biometriche con sistemi di smartcard, e tuttavia sono presenti fattori tecnologici, di costo e di progetto che possono indirizzare la scelta di una tra le due opportunità.

7.2.1 Analisi delle soluzioni disponibili

7.2.1.1 Accesso a librerie native C/C++ da classi Java

Grazie alla tecnologia JNI, Java Native Interface, è possibile richiamare a piacere una DLL (Dinamic Link Library), un DSO (Dinamic Shared Object) o qualsiasi altra libreria nativa sviluppata in linguaggio C/C++ su una specifica piattaforma.
Questa soluzione prevede lo sviluppo di headers C/C++ e di uno strato di interfaccia costituito da classi Java, compatibili con i requisiti della tecnologia JNI, e che permettano di mappare completamente ciò che si vuole rendere visibile delle API C/C++ (descritto dagli headers) alla Java Virtual Machine.
L'accesso alle librerie native avviene "by name", ovvero è possibile caricare qualsiasi implementazione nativa che risponda alle specifiche definite negli headers, semplicemente indicando il nome della libreria stessa ed eventualmente il path sul file system.
Il vantaggio principale di questa soluzione è, innegabilmente, la portabilità, garantita in parte dalla portabilità del codice Java, e completata dal fatto che le librerie devono essere *native sul sistema in uso*, ovvero equivalenti, ad esempio, alle DLL di un sistema MS Windows o ai DSO del sistema operativo Linux.
Naturalmente, questa soluzione è applicabile esclusivamente a oggetti nativi sul sistema in uso, e quindi non per linguaggi ad alto livello come Visual Basic e con tutti i linguaggi interpretati, mentre attualmente non è chiaro che cosa sia possibile fare in termini di integrazione con la piattaforma MS denominata *.NET*, presto a disposizione sui sistemi desktop basati su sistema operativo MS Windows, e comunque presto(?) disponibile su piattaforme Unix e free clones.
In questa soluzione, il linguaggio che dovrà essere utilizzato per lo sviluppo di applicazioni, front-end o script di vario genere è Java, e sarà quindi possibile accedere ai sistemi biometrici da applet (purchè firmate digitalmente), servlet, eseguibili stand-alone, e quindi coprire le esigenze di tutte le possibili necessità a partire dal desktop utente fino al sistema server.

Le piattaforme supportate, come precedentemente detto, sono in pratica tutte quelle dove è possibile installare una Virtual Machine Java.

7.2.1.2 Accesso alle classi java mediante parser JavaBeans/ActiveX

Questa soluzione prevede la costruzione, mediante opportuna configurazione delle impostazioni del sistema Windows, di uno strato di interfaccia che consenta di accedere alle classi java come se si trattassero di oggetti ActiveX, e quindi garantendo la possibilità di invocare metodi e funzioni da qualunque applicazione compatibile con la tecnologia VBA, ActiveX, COM, DCOM nonché mediante tutti i linguaggi di programmazione della suite MS Visual Studio.
La soluzione proposta prevede che le classi java siano rese compatibili con la tecnologia JavaBeans™, ovvero che siano conformi alla specifica SUN per la realizzazione di oggetti java portabili per strumenti di sviluppo RAD.
I componenti JavaBeans sono entità autodescrittive (nelle proprietà e nei metodi) mediante un meccanismo di *introspezione*, sono graficamente editabili attraverso un sistema di customizzazzione, nonché reattivi agli eventi attraverso un modello di link.

ActiveX è una tecnologia Microsoft che ha introdotto modifiche sostanziali al component object model dei sistemi *win16* e *win32*.
ActiveX è il meccanismo mediante il quale componenti diversi, implementati mediante tecnologie e su piattaforme differenti, possono comunicare tra loro: naturalmente, per poter essere platform indipendent, la tecnologia ActiveX deve essere supportata su tutti i sistemi interoperanti. Alla base di questa tecnologia vi è COM (Component Object Model[1]), DCOM (Distributed Component Object Model) e la più obsoleta e radicata OLE (Object Linking and Embedding). Se, ad oggi, ActiveX stenta ad estendersi su piattaforme diverse da quelle Microsoft, nei sistemi Windows è ormai radicata e consolidata, tanto da essere implementata mediante software sviluppati in Visual Basic, C++ e Java.

La comunicazione mediante ActiveX e JavaBeans può essere effettuata in due modi, ovvero mediante due strumenti messi a disposizione dai rispettivi fornitori delle due tecnologie: Sun e Microsoft.

7.2.1.2.1 JavaBeans bridge for ActiveX

Questo tool, sviluppato da SUN, permette di richiamare classi 100% Pure Java come dei normali ActiveX, mendiante un processo di generazione automatica. Il bridge è costituito da una applicazione, GUI Packager, che, preso in ingresso un **J**ava **Ar**chive, genera una serie di interfacce java necessarie per l'integrazione con l'ambiente OLE di Microsoft, nonché i file da aggiungere al registro di configurazione di Windows per consentire la registrazione dell'oggetto nel sistema perché sia visibile da tutte le applicazioni .
In questo modo, le classi Java saranno visibili dal sistema come un qualunque altro ActiveX, e quindi riutilizzabili come qualunque oggetto OLE in applicazioni compatibili (in pratica, tutte le applicazioni compatibili Microsoft Windows che consentono l'importazione di oggetti eterogenei mediante operazioni di Cut&Paste). Con un minimo di interazione con i file di registro del sistema, è possibile aggiungere ulteriori features, come l'associazione di file al controllo ActiveX, l'esecuzione automatica di operazione all'interno di browser (in special modo MS IExplorer), nonché la piena integrazione con la suite Microsoft Office e le applicazioni VBA (per esempio, per la generazione di macro e script).

[1] COM è l'antagonista di CORBA per casa Microsoft. Come CORBA, è un sistema platform indipendent che, mediante la costruzione di un Interface Definition Layer, consente di mettere in comunicazione oggetti sviluppati in linguaggi diversi. COM è evoluto in DCOM, che costituisce il primo passo per l'evoluzione alle piattaforme .NET, nonché il core dei sistemi operativi Windows 2000 e successivi.

Questa soluzione gode del vantaggio di essere particolarmente semplice e di avere un minimo impatto (praticamente nullo) sul codice Java già sviluppato.
Il vincolo principale risiede nel fatto che il componente Java deve rispettare le specifiche JavaBeans™.

7.2.1.2.2 Microsoft Software Developer's Kit for Java

Il tool di Microsoft, *javareg*, consente sostanzialmente di registrare classi java come componenti ActiveX, indipendentemente dal fatto che rispettino o no la specifica JavaBeans™.
Come il software SUN, *javareg* non richiede alcuna modifica al codice Java preesistente, e introduce un ulteriore risparmio non effettuando la costruzione di classi di bridging da integrare nella distribuzione, ma genera tutto il codice eseguibile necessario a run-time in modo automatico.
Questo vantaggio si paga essenzialmente nel fatto che è necessario l'uso di una Virtual Machine Java prodotta dalla stessa Microsoft, e non è possibile utilizzare quella fornita da SUN che offre maggiore compatibilità e pieno rispetto delle specifiche del linguaggio.

Entrambe le soluzioni, sono evidentemente vincolate all'uso con sistemi Microsoft. Su piattaforme Macintosh è attualmente presente il supporto per componenti ActiveX, ma i vincoli principali sono dovuti ai produttori di smartcard, la maggior parte dei quali non rilasciano librerie compatibili con sistemi Apple, mentre su sistemi *nix attualmente non sono disponibili piattaforme COM capaci di garantire il funzionamento di ActiveX.

La costruzione di un bridge ActiveX è avvantaggiata dalla semplicità d'uso e la rapidità con la quale è possibile effettuare l'integrazione, specialmente pensando allo sviluppo di prototipi, nonché il fatto che la generazione dei front-end e delle applicazioni può essere fatta mediate linguaggi di programmazione ad alto livello, RAD e comunque di grande diffusione e reperibilità su sistemi MS Windows.
La tecnologia JNI a suo vantaggio ha la maggiore portabilità su piattaforme diverse da quelle Microsoft: in tal senso è possibile pensare all'uso su sistemi *nix e sulla piattaforma Mac OS X, la quale differisce dalle versioni precedenti perché è, di fatto, un vero e proprio sistema Unix.
Viceversa, l'uso di tecnologia JNI implica il dover scrivere una interfaccia di riferimento per il codice C/C++ (gli headers) e lo sviluppo delle chiamate alle procedure native. Anche se, di fatto, l'attività è comunque minimale, è pur sempre una scelta maggiormente impegnativa in termini di tempi e di investimento di know how.

7.2.2 Analisi tecnica e soluzione di implementazione

7.2.2.1 Sun's JavaBeans Bridge for ActiveX

Il vincolo principale di questa soluzione risiede nel fatto che il componente java deve rispettare le specifiche JavaBeans.

Il tool **Packager** fornito da Sun mette a disposizione sia un'interfaccia grafica sia una a linea di comando. Il JavaBeans su cui vogliamo effettuare il bridging deve essere inserito in un file Jar insieme all'eventuale codice di supporto che serve al JavaBeans per essere eseguito.

L'interfaccia grafica del Packager guida l'utente attraverso il processo di bridging del JavaBeans. Prima l'utente seleziona il file Jar contenente il JavaBeans, poi il nome del controllo ActiveX da creare e la directory di destinazione per i file creati.
Infine l'utente clicca sul bottone Start Generation ed il Packager crea il type library file (tlb) ed il registry file (reg). Questi due file permettono così di accedere al JavaBeans come se fosse un componente ActiveX.

Il file di registro contiene informazioni circa il controllo ActiveX, inclusa la locazione del suo tlb e del file Jar contenente il codice. Le informazioni contenute nel file reg sono inserite nel registry del SO in modo da permettere alle applicazioni compatibili con la tecnologia ActiveX, COM, DCOM ecc. di farne uso.

Il file type library contiene informazioni riguardo all'interfaccia COM fornita dal controllo ActiveX. Esso descrive le proprietà, gli eventi e i metodi del componente JavaBeans sul quale è stato effettuato il bridging.

Quando è stato effettuato il bridging di un JavaBeans, se vuole essere utilizzato su una qualsiasi macchina, questa dovrà essere dotata di:

- JDK oppure JRE della Sun.
- Il Jar file contenente il JavaBeans.
- Il file di registro ed il file tlb generati dal Packager.

Notare che il file di registro contiene delle informazioni riguardanti le directory. Potrebbe essere necessario modificare questi valori per poter utilizzare il controllo ActiveX. Questo deve essere fatto prima di caricare il file nel registro di sistema.

I vantaggi principali di questo tool sono:

- La maggiore compatibilità con il codice Java sviluppato con il JDK di Sun.
- La generazione dei file reg e tlb. Questi permettono di esportare facilmente su altre macchine i Javabeans sui quali si è effettuato il bridging.
- Un'interfaccia grafica.

Lo svantaggio è quello di far rispettare alla classe Java le specifiche JavaBeans e la necessità di "impacchettare" le classi in file Jar.

7.2.2.2 Microsoft's Software Developer's Kit (SDK) for Java

Il tool messo a disposizione da Microsoft all'interno del SDK è **javareg**.
Anche se effettua le solite operazioni del Packager di Sun, il suo approccio è piuttosto differente.

In aggiunta alla registrazione di JavaBeans come controlli ActiveX, esso fornisce la possibilità di registrare qualunque tipo di classe Java come oggetti standard COM o componenti ActiveX indipendentemente dal fatto che rispettino o no la specifica JavaBeans.

Javareg non ha una GUI ma viene utilizzato solo da linea di comando, specificando i parametri neccessari per registare il JavaBeans o la classe java. Questo tool a differenza di quello della Sun non genera un file reg, ma aggiorna direttamente il registro di sistema, limitando così l'accesso alle informazioni che vengono inserite nel registro.

Javareg per registrare controlli ActiveX genera, come il tool di Sun, un file tlb che contiene le dichiarazioni per l'interfaccia pubblica del JavaBeans.

L'uso del controllo ActiveX è lo stesso sia dopo aver effettuato il bridging con il tool di Sun sia con quello di Microsoft. Il JavaBean appare come un normale controllo ActiveX. Quando è caricato in un'applicazione, la Microsoft Java VM viene caricata e crea un'istanza del JavaBean.

Quando è stato effettuato il bridging di un JavaBeans o di una classe Java con javareg, per poterlo utilizzare su una qualsiasi macchina, questa dovrà essere dotata di:

- Una recente versione della Microsoft Java VM
- Il javareg.exe dal Microsoft Java SDK
- I file class da registare

Javareg dovrà quindi essere eseguito sulla macchina nella quale vogliamo utilizzare il componente java come controllo ActiveX.
Per far questo in maniera più rapida potrebbero essere creati degli script nei quali viene richiamato javareg con gli opportuni parametri.

I vantaggi principale di questo tool sono:

- La possibilità di registare classi java che non adottano la specifica Javabeans.
- La velocità con cui viene registrata una classe java (non c'è bisogno di inserire la classe in archivi compressi e di creare file manifest)

Gli svantaggi principali sono:

- La non creazione del file reg.
- L'utilizzo della Microsoft Java VM, meno aggiornata e meno compatibile rispetto alle specifiche del linguaggio (per es. non è possibile registrare come controllo ActiveX una classe che faccia uso delle classi Swing)
- Mancanza di una GUI.

7.2.3 Integrazione tra le API di Centro Biometrika e le librerie di firma digitale

Lo strumento scelto per effettuare il lavoro di interfaccia ActiveX-java è quello di Sun.
Il motivo principale che ha portato a questa scelta è stato evitare futuri problemi di compatibilità che potrebbero nascere durante l'integrazione.

Dato che tutte le librerie java di firma digitale sono state implementate con il JDK 1.3 di Sun, se venisse adottata la soluzione Microsoft, il codice scritto per il JDK 1.3 sarebbe eseguito dalla Microsoft Java VM.
Nella versione fornita dal Microsoft SDK for Java 4.0 le classi implementate corrispondono a quelle che si trovano nel JDK 1.1.4 di Sun.

Nel caso siano state utilizzate particolari classi, create nelle ultime versioni del JDK, potrebbero crearsi evidenti problemi di integrazione.

7.2.4 Salvataggio, conservazione e accesso alle informazioni biometriche su smartcard PKCS#11 mediante applicazione Java

In questo paragrafo sono proposte alcune interfacce Java per l'implementazione di una API con funzionalità di accesso alla smartcard e trattamento di informazioni biometriche salvate sul file system della carta stessa.
La modalità di protezione delle informazioni prevista, è di tipo *accesso pubblico in lettura*, che prevede che il template sia conservato in un file dichiarato come pubblico con divieto di modificazione, cancellazione o sostituzione se non a seguito di un accesso autenticato sulla smartcard da parte del possessore della carta.

7.2.4.1 FingerprintData

it.iat.biometrika.pkcs11
Interface FingerprintData

---

public interface **FingerprintData**

Questa classe implementa un oggetto FingerPrint Data, che consente il trattamento di impronte biometriche e dei relativi dati identificativi indipendentemente dal formato con il quale sono rappresentate.

Method Summary	
byte[]	**getData**() restituisce l'impronta biometrica
byte[]	**getID**() restituisce i dati identificativi dell'impronta biometrica
void	**setData**(byte[] data) Imposta i dati dell'impronta biometrica
void	**setID**(byte[] id) Imposta i dati identificativi dell'impronta biometrica, quali, ad esempio, nome e cognome dell'utente.

Method Detail

setData

public void setData(byte[] data)

Imposta i dati dell'impronta biometrica
Parameters:
data - l'impronta, rappresentata come un array di bytes

---

setID

public void setID(byte[] id)

Imposta i dati identificativi dell'impronta biometrica, quali, ad esempio, nome e cognome dell'utente.
Parameters:
id -

---

getData

public byte[] getData()

throws FingerprintException

restituisce l'impronta biometrica
Returns:

Throws:
FingerprintException -

---

getID

public byte[] getID()

throws FingerprintException

restituisce i dati identificativi dell'impronta biometrica
Returns:

Throws:
FingerprintException -

7.2.4.2 FingerprintIO

it.iat.biometrika.pkcs11
Interface FingerprintIO

---

public interface **FingerprintIO**
Interfaccia per l'implementazione delle comunicazioni con il repository delle impronte biometriche. Questa classe definisce i metodi con i quali è possibile richiedere ad una smartcard PKCS#11 il salvataggio, la consultazione ed il trattamento di dati sulla smartcard

---

Method Summary	
void	**close**() Chiude la sessione dicomunicazione con la smartcard
void	**deleteFingerprint**(byte[] id, java.lang.String pin) Cancella una impronta dalla smartcard
FingerprintData	**getFingerprintData**(byte[] id) restituisce i dati di una impronta contenuta nella smartcard
java.util.Vector	**getIDs**() Restituisce un elenco degli ID delle impronte contenute nella smartcard.
void	**init**(java.lang.String pkcs11NativeLibrary) Inizializza la comunicazione con la smartcard, inviando il nome della libreria nativa da utilizzare per la comunicazione PKCS#11.
void	**putFingerprintData**(FingerprintData data, java.lang.String pin) Inserisce i dati di una impronta biometrica all'interno della smartcard
void	**renameID**(byte[] id, byte[] newID, java.lang.String pin) Modifica i dati identificativi di una impronta

Method Detail

init

public void init(java.lang.String pkcs11NativeLibrary)
throws RepositoryException
Inizializza la comunicazione con la smartcard, inviando il nome della libreria nativa da utilizzare per la comunicazione PKCS#11.
Parameters:
pkcs11NativeLibrary - il nome della libreria nativa fornita dal produttore della smartcard
Throws:
RepositoryException - errore di accesso alla smartcard

---

close

public void close()
throws RepositoryException
Chiude la sessione dicomunicazione con la smartcard
Throws:
RepositoryException - errore durante lòa chiusura della sessione PKCS#11

---

putFingerprintData

public void putFingerprintData(FingerprintData data,
java.lang.String pin)
throws FingerprintException,
RepositoryException
Inserisce i dati di una impronta biometrica all'interno della smartcard

Parameters:

data -

pin - il pin di accesso alla smartcard

Throws:

FingerprintException - in caso di errore nei dati di input

RepositoryException - errore durante l'accesso ed il salvataggio sulla smartcard

---

getFingerprintData

public FingerprintData getFingerprintData(byte[] id)

throws FingerprintException,

RepositoryException

restituisce i dati di una impronta contenuta nella smartcard

Parameters:

id - l'id dell'impronta

Returns:

Throws:

FingerprintException - in caso di errore nei dati letti dalla smartcard

RepositoryException - errore durante l'accesso ed il salvataggio sulla smartcard

---

getIDs

public java.util.Vector getIDs()

throws FingerprintException,

RepositoryException

Restituisce un elenco degli ID delle impronte contenute nella smartcard.

Returns:

Throws:

FingerprintException - in caso di errore nei dati letti dalla smartcard

RepositoryException - errore durante l'accesso ed il salvataggio sulla smartcard

---

deleteFingerprint

public void deleteFingerprint(byte[] id,

java.lang.String pin)

throws FingerprintException,

RepositoryException

Cancella una impronta dalla smartcard

Parameters:

id - l'identificativo dell'impronta das eliminare

pin - il pin di accesso alla smartcard

Throws:

FingerprintException - in caso di errore nei dati contenuti nella smartcard

RepositoryException - errore durante l'accesso ed il salvataggio sulla smartcard

---

renameID

public void renameID(byte[] id,

byte[] newID,

java.lang.String pin)

throws FingerprintException,

RepositoryException

Modifica i dati identificativi di una impronta

Parameters:

id - i dati identificativi dell'impronta attualmente impostati

newID - i dati identificativi dell'impronta da sostituire ai vecchi

pin - il pin di accesso alla smartcard

Throws:

FingerprintException - in caso di errore nei dati di input

RepositoryException - errore durante l'accesso ed il salvataggio sulla smartcard

7.2.4.3 FingerprintException

it.iat.biometrika.exceptions
Class FingerprintException

java.lang.Object
  |
  +--java.lang.Throwable
      |
      +--java.lang.Exception
          |
          +--it.iat.biometrika.exceptions.FingerprintException

All Implemented Interfaces:
java.io.Serializable

---

public class **FingerprintException**
extends java.lang.Exception
See Also:
Serialized Form

Constructor Summary
FingerprintException() Creates new FingerprintException without detail message.
FingerprintException(java.lang.String msg) Constructs an FingerprintException with the specified detail message.

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

FingerprintException

public FingerprintException()

Creates new FingerprintException without detail message.

FingerprintException

public FingerprintException(java.lang.String msg)

Constructs an FingerprintException with the specified detail message.

Parameters:

msg - the detail message.

7.2.4.4 RepositoryException

it.iat.biometrika.exceptions
Class RepositoryException

java.lang.Object
  |
  +--java.lang.Throwable
      |
      +--java.lang.Exception
          |
          +--it.iat.biometrika.exceptions.RepositoryException

All Implemented Interfaces:
java.io.Serializable

---

public class **RepositoryException**
extends java.lang.Exception
See Also:
Serialized Form

Constructor Summary
RepositoryException() Creates new RepositoryException without detail message.
RepositoryException(java.lang.String msg) Constructs an RepositoryException with the specified detail message.

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

RepositoryException
public RepositoryException()

Creates new RepositoryException without detail message.

RepositoryException
public RepositoryException(java.lang.String msg)

Constructs an RepositoryException with the specified detail message.

Parameters:

msg - the detail message.

---

8 Bibliografia

[1] ECMA 2.19 Authentication and Privilege Attribute Security Application with related key distribution functions

[2] ISO 8824:1993 Information Processing Systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1)

[3] ISO 8825-1:1993 Information Processing Systems - Open Systems Interconnection - Specification of Encoding Rules for Abstract Syntax Notation One (ASN.1) - Part 1 - Basic Encoding Rules

[4] ISO/IEC 7816-4, “Identification Cards - Integrated circuit(s) cards with contacts - Part 4: Interindustry commands for interchange.” 1 September 1995.

[5] RSA Public Key Cryptography Standard 11: Cryptographic Token Interface Standard

[6] RFC 2459 Internet X.509 Public Key Infrastructure Cert and CRL Profile

[7] RFC 2510 Certificate Management Protocols – CMS

[8] R22 User Authentication Techniques Using Public Key Certificates Part 2: Authentication Information Including Biometrics – NSA

[9] Public Key Cryptography Standards - RSA Technologies