An Open Source Java Framework for Biometric Web Authentication based on BioAPI

Conference Paper · September 2007

DOI: 10.1007/978-3-540-74827-4_102 · Source: DBLP

CITATIONS
5

READS
4,072

4 authors, including:

Elisardo González Agulla
University of Vigo
19 PUBLICATIONS **338** CITATIONS

SEE PROFILE

Jose Luis Alba-Castro
University of Vigo
110 PUBLICATIONS **1,230** CITATIONS

SEE PROFILE

Carmen García-Mateo
University of Vigo
292 PUBLICATIONS **1,212** CITATIONS

SEE PROFILE

An Open Source Java Framework for Biometric Web Authentication Based on BioAPI

Elisardo González Agulla, Enrique Otero Muras, José Luis Alba Castro, and Carmen García Mateo

Department of Signal Theory and Communications, University of Vigo
Lagoas Marcosende, 36310, Vigo, Spain
eli@gts.tsc.uvigo.es

Abstract. One of the major early problems that biometrics faced was the lack of interoperability between different software applications and devices developed by different vendors. The BioAPI Specification has defined an open system standard application program interface (API) which allows software applications to communicate with a broad range of biometric technologies. This paper describes the design and implementation of an open source Java framework intended to provide single sign-on web authentication based on BioAPI-compliant biometric software or devices. The open-source principle makes this system a novel and practical development tool for testing and integrating biometric modules and devices from third parties.

Keywords: Biometrics, web authentication, BioAPI, open source.

1 Introduction

The popularity of broadband internet and cell phone connections has introduced many high added-value remote services where it is important to verify the identity of the authorized user. Classical techniques for electronic person authentication have several drawbacks in terms of performing reliable and user-friendly identity recognition; this occurs particularly with remote operations, where hacker attacks add to forgotten, shared, lost or stolen passwords or cards. Automatic identity verification, based on distinctive anatomical features (e.g., face, voice, fingerprint, iris, etc.) and behavioral characteristics (e.g., online/ offline signature, keystroke dynamics, etc), is becoming an increasingly reliable standalone solution and attracting a great deal of attention as far as remotely-based applications are concerned. Nevertheless, this kind of biometric authentication implies new technological challenges that must be addressed to ensure full acceptance in many remote services [1].

One of the major early problems that biometrics faced was the lack of interoperability between different software applications and devices developed by different vendors. For this reason, the BioAPI Consortium was founded to develop a biometric Application Programming Interface (API) that would bring platform and device independence to application programmers and Biometric

B. Apolloni et al. (Eds.): KES 2007/ WIRN 2007, Part II, LNAI 4693, pp. 809–815, 2007.

Service Providers (BSPs). The BioAPI Consortium developed a specification and reference implementation for a standardized API (BioAPI 1.1) [2] that is compatible with a wide range of biometric application programs and a broad spectrum of biometric technologies [3]. The BioAPI Specification defines an open system standard application program interface (API) which allows software applications to communicate with a broad range of biometric technologies.

This paper describes an open-source Java system intended to provide single sign-on web authentication based on BioAPI-compliant biometric software or devices. Integration of multiple biometric devices and algorithms (interoperability) is quite straightforward, as we will demonstrate in the sections below. We have paid a great deal of attention to the design of a client-server architecture capable of controlling any kind of biometric acquisition device over a secure TCP/IP connection.

Otherwise, we have focused on driving user interaction with the system through an easily configurable human-machine dialogue. Thus, verification tasks are modeled and specified by an XML document which describes the sample acquisition process and the biometric verification mode.

The remainder of this paper is organized as follows. Section 2 is devoted to the description of the main design guidelines followed in the development of our system for single sign-on web authentication based on server-side biometric verification. Section 3 focuses on the integration of the BioAPI framework within our system for allowing server-side biometric authentication in web environment. Section 4 presents the overall system, described both from a structural and functional point of view. Finally, Section 5 describes our conclusions and future research lines.

2 Design Guidelines

This section describes the main design decisions taken in creating a web-based system for biometric authentication that complies with the principles of security, interoperability and usability.

Any biometric recognition procedure can be divided into a number of stages:

1. Acquisition of a biometric sample (device-dependent processing).
2. Extraction of a biometric template (signal processing: pre-processing, feature extraction and user-template creation).
3. Biometric template matching (pattern recognition processing).

The third of these stages usually requires prior training of user models and thresholds set up in an enrolment process whenever a new user is added to the system.

When dealing with biometric authentication in client-server architectures, the three processes described above will provide different configurations depending on where the processes are executed. Although it is obvious that the acquisition process must be executed on the client side, there are three options remaining for extracting biometric templates and matching user reference templates: both

these processes are performed in the client machine (pull model); the biometric template is extracted on the client side, sent to the server and matched there against the user reference templates (push model); or both processes are performed in the server (a variant of the push model).

The system we have developed is based on the third of the above configurations. The client-side biometric application is in charge of multi-biometric sample acquisition, encryption and secure transaction. On the server side, there is a centralized authentication server with a biometric authentication module that is in charge of extracting the biometric template, matching, and checking access privileges.

The main guidelines for the construction of our system for biometric authentication have been the following:

- Maximized portability
  - of the client application, by using Java as programming language. Concretely, the biometric client application uses the Java Web Start technology, a Java-based environment for running desktop applications.
  - of the verification server, by following the standardized Java Enterprise Edition (EE) specification in web application development, permitting deployment in any Java EE compliant server.
- Maximized interoperability
  - with different biometric software or devices, by using a Java Native Interface wrapper for the BioAPI framework that is compatible with the BioAPI 1.1 reference implementations on Linux and Win32 operating systems.
  - with different software platforms, by using web services as middleware, permitting the development of alternative client applications that comply with our system.
- Use of free open source software: e.g. JBoss Application Server, MySQL database...
- Maximized security in managing biometric data, trying to comply with the Core Security Requirements of the ANSI X9.84 standard for Biometric Information Management and Security [4]. For this purpose SSL connections are used, and local disk writing of user samples is avoided, for instance.
- Maximized code reusability and maintainability, by using object-oriented software and design patterns.
- Multilingual support both in the web and in the client desktop application.

3 Integrating the BioAPI Framework into Our Architecture for Server-Side Biometric Verification

In order to allow server-side biometric verification based on BioAPI-compliant modules, the server contains a BioAPI instance running in a Linux platform and allowing third party biometric modules compliant with BioAPI 1.1 and compiled in C/C++ to be integrated in our system as Linux shared objects.

All these biometric modules can be controlled through this server-side BioAPI. BioAPI-compliant modules are also called Biometric Service Providers or BSPs following BioAPI nomenclature.

Moreover, if a user has a BioAPI installed locally in the client machine (Linux or Win32 platforms), the corresponding local BSPs will be also driven by our client application. Nevertheless, for a complete integration of any BioAPI 1.1 compliant BSP within our authentication system by using server-side verification for more security, compiled versions of the BSP libraries for both Linux and Win32 platforms are required.

Each BSP involved in the verification or enrolment process is properly indicated in an XML document that contains a high-level description of the protocol for the current session of verification or enrolment. This document is securely transferred from server to client when starting the session.

The low-level processing is composed of the primitives `BioAPI_Capture`, `BioAPI_CreateTemplate`, `BioAPI_Process` and `BioAPI_VerifyMatch`. In the more secure configuration, all these low-level primitives except `BioAPI_Capture` are executed as calls to a BSP in the server through the remote BioAPI. The `BioAPI_Capture` primitive is executed as a call to the BioAPI and BSP installed in the client machine and the resulting Biometric Identification Record (BIR) is sent back to the server (see Figure 1).

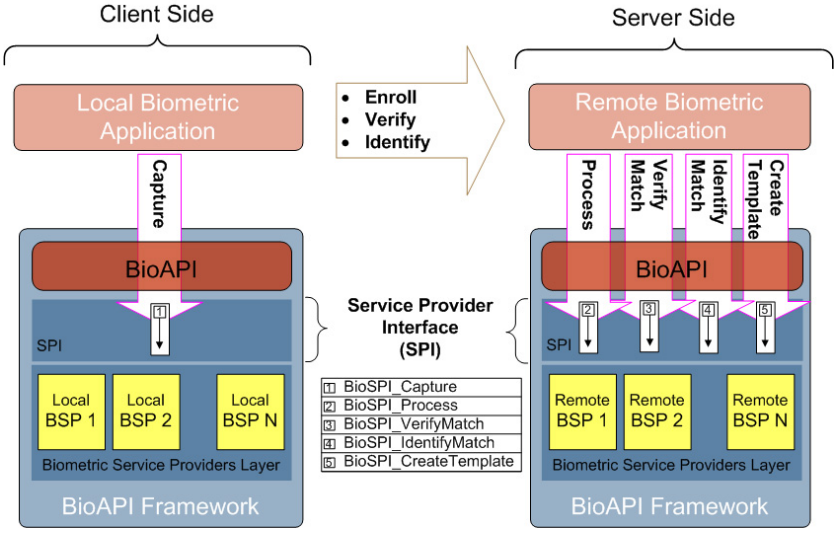

Fig. 1. Building-blocks of the BioAPI model in client/server environment

Java management of the different BSPs through local and remote BioAPI instances is performed using a Java wrapper that manages the calls to native code based on the Java Native Interface technology. Concretely, an open source Java Native Interface wrapper for the BioAPI framework on Linux/Unix has

been used [5]. To integrate into our system, this Java wrapper has been extended to include Windows support and access to low-level BioAPI primitives.

With the advent of the standard BioAPI 2.0, the definition of an additional layer of components below the BSPs (called Biometric Function Providers or BFPs) will allow our system to relax some of the interoperability conditions for including new biometric software for server-side verification. Only the corresponding implementation (whether Windows or Linux) of the BFP in charge of the capture process will be necessary on the client side (called Sensor BFPs), and only the Linux implementation of the BFPs in charge of the remaining processes will be necessary on the server side (archive, processing-algorithm and matching-algorithm BFPs). Thus, the server can perform verification without forcing the client to have a duplicate of the complete BSP.

Otherwise, we use a web service as middleware for the communication between the clients and the server. Web services with SOAP (Simple Object Access Protocol) allow interoperability between heterogeneous clients, permitting the development of new client applications that are fully compatible with our system and that do not need to be Java-based, merely compliant with the agreed interface. This functionality is particularly suitable for client terminals without a Java Virtual Machine.

4 The Overall Framework

As mentioned above, an XML document specifies the biometric sample acquisition protocol in an enrolment or verification session. This document represents the man-machine dialogue. Due to the inexistence of a universally agreed markup language for biometric acquisition we have developed an extension to the VoiceXML standard suited to our necessities. Our system can be easily translated into any other future markup language.

Figure 2 depicts a block diagram and functionality description for the system. Starting from a client verification or enrolment request, the successive actions and functionalities are explained as follows (see diagram numbering):

1. The biometric client application obtains, from the server, the XML dialogue that contains the enrolment or verification process description.
2. The client application interprets the protocol contained in the XML dialogue, prompts the corresponding information to the user, acquires the biometric sample, and performs an enrolment or verification.
3. Each time a biometric sample is required, the sample is captured from the corresponding BioAPI-compliant module. For this purpose, the client application calls the `BioAPI_Capture` primitive using a Java Native Interface wrapper for the BioAPI framework.
4. The result of the acquisition process is sent to the server bound to an enrolment or verification request.
5. The enrolment or verification process is executed in the server as a sequence of BioAPI calls.

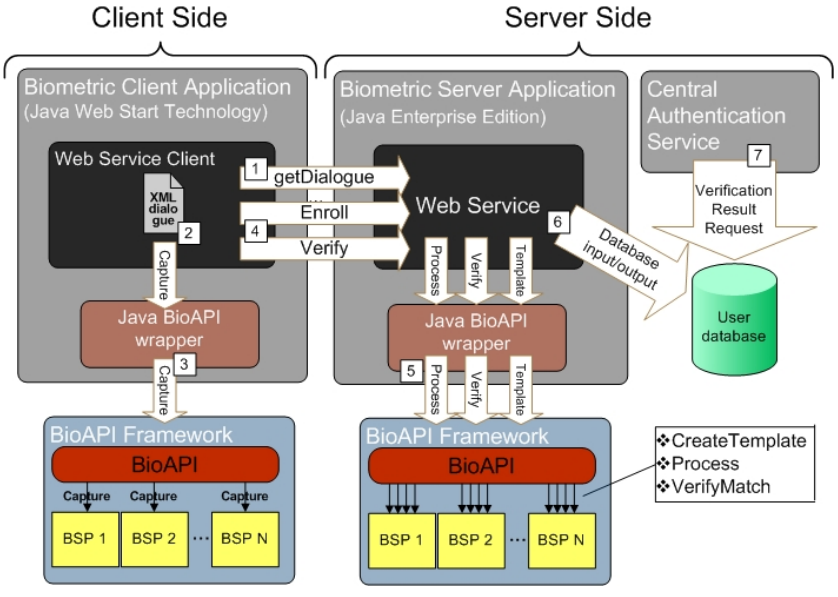

Fig. 2. Building-blocks and functionality description of the overall architecture

6. The verification result or the enrolment template is stored in the server database.
7. The database with the biometric verification results will be available to finally authenticate users for the web through the Centralized Authentication Service (CAS).

The main objective of the system was to offer biometric authentication capabilities to other web-based applications. For this purpose, we have integrated our system within a widely accepted Java-based open-source authentication solution known as Central Authentication Service (CAS) [6] [7]. CAS is a single sign-on solution originally developed at Yale University and later placed under the auspices of the Java Architectures Special Interest Group. CAS has quickly become the most popular single sign-on solution for universities. The main idea behind the integration of biometric verification functionality within the Central Authentication Service was to take advantage of the infrastructure provided by CAS to offer single sign-on web authentication, while improving security beyond basic mechanisms based on login and password, by adding biometrics. Thus, we make possible that any application prepared to use CAS for authenticating its users can also use our biometric system for this purpose, supporting any BioAPI-compliant biometric software or device in order to perform the authentication. The open-source e-learning platforms Moodle, ILIAS, or Claroline are well known examples of web applications that are yet capable of relying the authentication task on CAS. We had used Moodle and ILIAS to demonstrate the usability of our biometric extension of CAS within a common web application.

5 Conclusion and Future Work

In this paper we have presented and explained the design and implementation of an open source Java framework intended to provide single sign-on web authentication based on BioAPI-compliant biometric software or devices.

In order to control any BioAPI-compliant device from our Java architecture we have used an open-source Java Native Interface wrapper for the BioAPI framework that is compatible with both BioAPI 1.0 reference implementations of Linux and Windows platforms.

Moreover, diverse open-source solutions have been carefully integrated into our system attending to flexibility, portability, and interoperability issues.

Current version of our open-source biometric authentication system is available under LGPL license on [8]. Regarding future lines of work, we are analyzing the viability of porting our biometric system for web authentication to mobile devices.

Acknowledgments. This project has been partially supported by Spanish MEC under the project PRESA TEC2005-07212, Xunta de Galicia with the projects PGIDIT05TIC32202PR and PGIDIT05PXIC32202PM, and the European NoE BioSecure.

References

1. Jain, A., Bolle, R., Pankanti, S.: Introduction to Biometrics. In: Biometrics. Personal Identification in Networked Society, Kluwer Academic Publishers, Dordrecht (2002)
2. BioAPI Consortium (ANSI/INCITS 358-2002), `http://www.bioapi.org`
3. Yuan, X., Hui, S.C., Leung, M.H.K., Gao, Y.: Towards a BioAPI compliant face verification system. Computer Standards & Interfaces 26, 289–299 (2004)
4. ANSI X9.84-2003, Biometric information management and security for the financial services industry. American National Standards Institute, New-York (USA) (2003)
5. JBioAPI, a library of tools for accessing BioAPI-compliant biometric service providers in Java: `http://code.google.com/p/jbioapi/`
6. JA-SIG (Java Architectures Special Interest Group) Central Authentication Service (CAS): `http://www.ja-sig.org/products/cas/`
7. Aubry, P., Mathieu, V., Marchal, J.: ESUP-Portail: open source Single Sign-On with CAS (Central Authentication Service). In: Proceedings of EUNIS04 - IT Innovation in a Changing World, Bled (Slovenia) (July 2004)
8. Biometrics for Web Authentication: `http://sourceforge.net/projects/biowebauth/`