The Universal Biometric System

H. M. N. Dilum Bandara, S. M. Ravindra P. De Silva, and P. W. H. Dasun Weerasinghe
Department of Computer Science and Engineering,
University of Moratuwa,
Sri Lanka.
dilumb@cse.mrt.ac.lk

Abstract

The Universal Biometric System is a biometric enabled third-party authentication system. It tries to address some of the issues relating to integration of biometrics in an enterprise level network. We present a system that hides all the complexity of biometrics and provides biometric technology, vendor, and platform independent authentication. It also introduces two novel ideas; many-to-many mapping to reduce the total cost of ownership while device-hierarchy enforces in-depth security. Biometric vendor, technology, and platform independence is achieved by implementing the system on top of the BioAPI (the Defacto standard for biometrics). However our system sets its sights far beyond the BioAPI. It is designed to provide a simple development environment that does not require complex data structures, pointers, and memory management inherent to the BioAPI. This is a Proof of Concept effort.

Keywords- Biometric, BioAPI, biometric service provider, device hierarchy, networking, security.

1.0 Introduction

[1]Biometrics is an open-ended set of technologies based on the measurement of some unique physical characteristics of human beings (or even animals) for the purpose of identifying an individual or verifying identity. Simply saying "*your body is your password*". Biometrics is today's prime technology when it comes to access control, especially in medium to large-scale organisations. At present, technology is matured enough and has proven it is the current best when tight security is the main concern. It is convenient to use, publicly accepted (up to a certain extent), and more importantly affordable. Users have all forms of biometrics technologies (Fingerprint, Iris, Retina, Facial, etc.) to choose, based on the required level of security and budget constrains.

With such a value proposition, it is disheartening to see the low level of deployment of biometrics. The reason is the difficulty of integration with in a networked environment at a low cost. Thereby this paper describes a concept and its implementation to integrate biometric technology at low cost with less effort while enforcing in-depth security [1].

2.0 Biometric integration

The complexity of integration is the major factor that is holding the market for biometrics and its large-scale deployment. Most of the time it is too hard, too costly and some times it is impractical as well. It does not easily fit into today's complex enterprise level networks. There are a very few solutions that meet up this challenge, even those solutions are either limited to a specific biometric technology, vendor, or platform.

If any organization is moving into biometrics for access control, according to the current practice it will require to install same type of device from the same vendor, all over the organization. This raises three concerns. Firstly, it is required to have a dedicated device for each and every doorstep and host (PC/Server). Secondly, organizations have to stick with the same type of devices regardless of required level of security. Thirdly, organizations are in a dilemma when they want to scale up (i.e., problems related to integration, tight dependence on a particular vendor, inability to go forward with the latest technological developments due to backward compatibility issues, etc.).

From the application developers point of view they need to master a specific Software Development Kit (SDK) provided by its device vendor. In most cases this requires thorough knowledge of C/C++ or even Assembly

Last updated 31/07/2008

programming, which is very inconvenient for a typical developer.

3.0 Previous work

Based on our literature survey it seems that the Universal Biometric System (referred as ***Universal BioSys***) is unmatched. There are several security solutions that are related and worth mentioning here.

The *Independent Security Server* provides a mean to identify a person based on any biometric characteristics [3]. The developers provide set of Biometric Service Provider (BSP) libraries for all the popular products therefore it is compatible with all major biometric scanners. Hence users have to rely on the service provider to ensure compatibility with whatever the future biometric device they purchase.

The *WhoIsIt* biometric server for e-Commerce is an application server hosted in the Internet where a client system sends a biometric template to be verified. On success any secret (payload) that is stored for that particular user can be retrieved. *WhoIsIt* biometric server needs to be aware of the underlying technologies and the users are restricted to the vendors in commercial agreement with them [4]. Even in these systems it is obvious that the problems of vendor, platform and technology dependence are still present up to a certain level.

The BioAPI specification [2] could be considered as the Defacto industry standard framework for biometrics. It defines how application developers and device vendors communicate with each other through a standard framework. The BioAPI is designed to overcome the issues of technology, platform, and vendor dependencies. A reference implementation of the BioAPI is also freely available [2]. BioAPI is two fold; the application developer needs to comply with Application Program Interface (API) while the device manufacturer needs to comply with the Service Provider Interface (SPI). That is how platform, technology, and vendor independence are achieved.

However, these issues are not as simple as they sound. The developers need to have some basic idea of biometric technology and need master C/C++. The BioAPI consists of complex data structures, pointers (in average with level three indirection) and gruesome memory management. These required skills are far beyond the skills of an average developer.

4.0 Related work

The Universal Biometric System is a Proof of Concept that is intended to overcome the above-mentioned issues relating to biometric technology, vendor and platform dependence. The BioAPI is the core of the Universal BioSys, however it sets its sites far beyond BioAPI. It introduces a simple development platform hiding the complexities of the BioAPI. It also introduces two novel concepts.

4.1 many-to-many mapping

The Universal BioSys offers seamless *many-to-many* (*m-to-n*) mapping between biometric devices and hosts. This is one of its novel concepts. Current standard practice is to install a dedicated device for each and every host. So if an organization has 50 machines it has to have 50 devices. This is one of the major reasons that bar the heavy deployment of biometrics.

The Universal BioSys will map m number of devices into n number of hosts where m is much less than n ($m \ll n$) or m could even be 1 ($m=1$). However, in practice an organization may has to install several devices due to physical boundaries such as rooms, floors, or buildings mainly for better user convenience and fault tolerance. BioSys can share biometric devices among multiple hosts.

Consider a software development company that markets a proprietary product. It is essential that no one else other than the development team has access to its source code. If the development room has 50 PCs it needs 50 devices and few more at doorsteps. Based on our many-to-many mapping, the organization may need only five to ten biometric devices. It allows grouping of several hosts together and sharing them with one or more devices. It could even share all the installed devices among whole the hosts.

4.2 The user's view point

Many-to-many mapping approach share devices among each other through a network. Fig. 1 illustrates user authentication steps.

Step 1: User informs the application that he/she needs access and that request will be sent to the server.

Step 2: The BioSys server informs the user a device where user can submit his/her biometric credentials (this decision is given based on user's location, nearest device, and its availability).

Step 3: User submits his/her credentials and it is sent to the server where it get processed.

Step 4: Server carries out identify or verify functions and makes sure that user is either authenticated (according to the predefined policies) or rejected.

The reader may ask; how to handle a situation where some one else uses that machine while the user is still coming back (after submitting credentials). This is similar to a situation where a user moving away from a computer while already logged in. There is no real solution to this

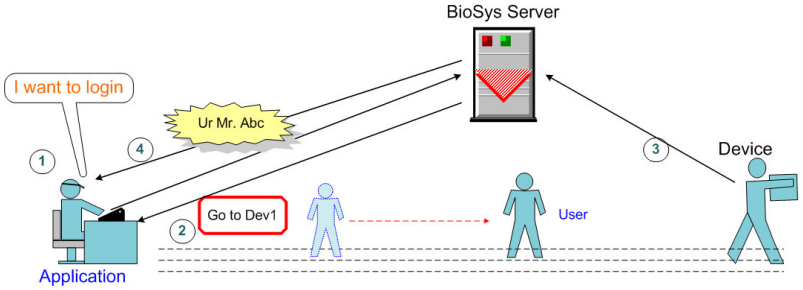

Fig. 1: Authentication steps involved with Universal BioSys.

problem (even without the BioSys) and this is a compromise between how much the owners are willing to pay and the level of security they expect. Possible solution would be not to place devices far away from the host or to use double authentication (i.e., first with biometric then possibly by means of a password).

4.3 Device-Hierarchy

Organizations may install multiple devices (either same or different biometric technologies) on different locations based on the required level of security while having a balance with Total Cost of Ownership (TCO). These devices automatically create different security levels which can be represented as a hierarchy.

Fig. 2 illustrates a hypothetical organization that has installed different devices based on their security requirement. They have installed card reader at the entrance and placed more secure fingerprint and iris scan systems at places like Server room, Engineering and R&D. Such an arrangement would result in different security levels which could be mapped into a logical hierarchy (Fig 3). Knowledge of this hierarchy (referred as *Device-Hierarchy*) could be used to gain in-depth security.

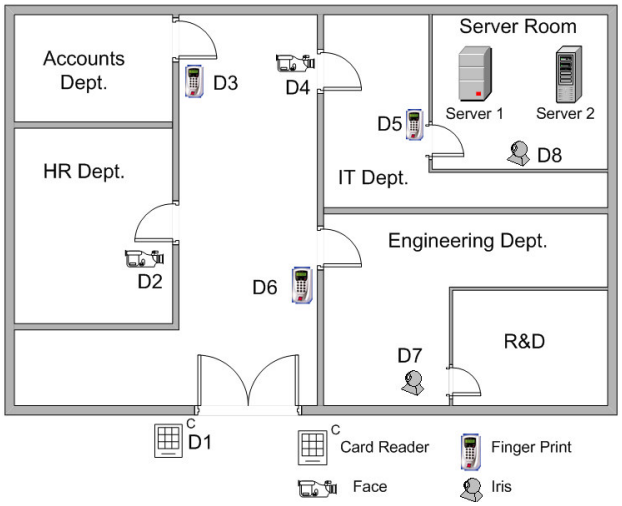

Fig. 2: Device arrangement of an organisation.

The Device-Hierarchy is automatically created when multiple devices are installed with different access levels. It is already there but the problem is no one sees it; therefore no one makes use of this hierarchy to gain tighter security.

According to the *defence in-depth* approach an organization should have security from its doorstep to the server room. Today all these security precautions are there with different access levels. However, these security measures are independent therefore it is possible to bypass one or more layers. Device-Hierarchy tries to integrate all these levels together and do not allow any layer to be bypassed. While doing so it enforces tighter security.

When multiple levels of security measures exist, a user needs to get authenticated through several devices in a specific order. Consider an example where the system administrator is going to the server room starting from the main entrance. First he/she has to get authenticated using the card reader at the entrance. Then he/she is required to use the facial recognition system at the IT department. Next if the administrator needs to go into the server room he/she has to get authenticated through the fingerprint device as well. The path followed by the system administrator can be represented by a specific branch in a tree which represents the Device-Hierarchy (Fig. 3). The branch includes device *D1*, *D4*, and *D5*.

From the system administrator's point of view he/she does not need to remember any of these devices or specific paths. It occurs naturally when users move around the organization.

This sort of path tracking and enforcement will make

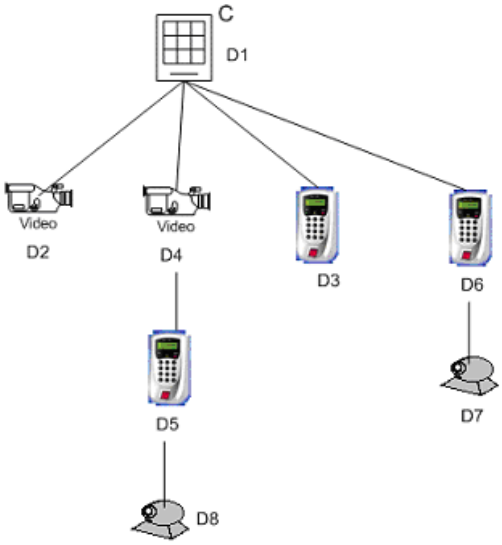

Fig. 3: Device hierarchy for the organization given in Fig. 2.

sure that users are not allowed to access any resources unless he/she has entered whatever the place according to the accepted route (i.e., branch in the tree). Consider a case where an unauthorized person is being able to get into the server room through the roof of the organization (or by any other means) and trying to access one of the servers in the server room. If the unauthorized person is able to provide a valid username password combination or is able to forge the biometric device attached to the server there is nothing stopping him/her from accessing the server. According to the concept of Device-Hierarchy the unauthorized user has violated the hierarchy (i.e., not gone through the accepted path). He/she has directly used device *D8* without getting authenticated through devices *D1*, *D4*, and *D5*. In this case the Universal BioSys will not provide any access to the server because the Device-Hierarchy is being violated, although the submitted credentials (to device *D8*) are valid. Enforcement of such a policy would result in-depth security from organizations doorstep to the servers.

Tracking employees (specially the IT support staff) in a large organization could be a real problem. Being aware of the Device-Hierarchy will enable the possibility of tracking users as well. Based on the last device used for authentication, a probable location within the organization can be identified. All this is possible because the administrator can configure the Universal BioSys with a map of the organization's floor plan (something similar to Fig. 2). Several maps can be used if the organization spans several building, multiple floors, or if it is too dense to put everything in a single map.

Whatever technologies come and go passwords will remain so many years to come, although it is easily forgettable by users or guessable by others. Therefore BioSys should also support password-based authentication. Although having all those features the Universal BioSys will not be completed if standard practices of network management and administration are not combined with security. Its design highly encourages such security precautions.

5.0 Design and implementation

The Universal BioSys consist of several components that are interconnected to each other (Fig. 4). The BioSys Server is the central point of communication and it performs all the administrative, management, policy enforcement and image processing tasks [6].

The Universal BioSys makes use of the BioAPI and it enables plug & play biometric components. The BioAPI was wrapped by adding another layer in-between the BioSys Server and the BioAPI. This in-between layer (referred as the *BioAPI Wrapper*) was essential since the BioAPI reference implementation is written in C/C++. It was not directly accessible through Microsoft .Net C# as the BioAPI deals with multiple levels of indirection of pointers and union data structures. Therefore it was

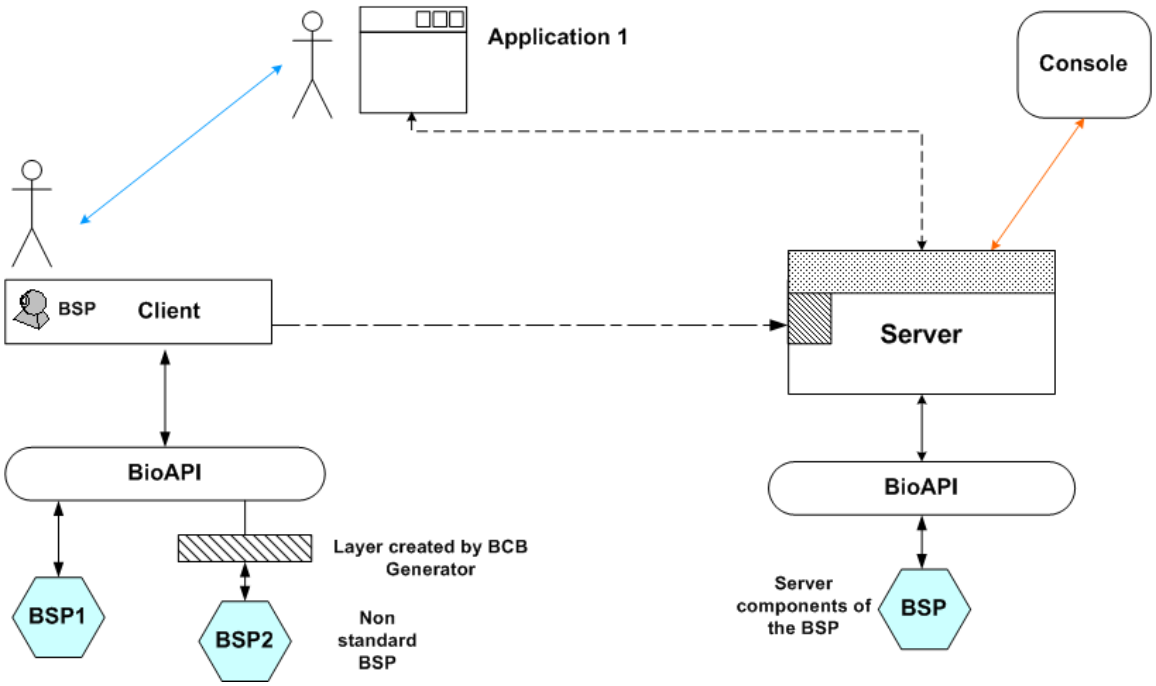

Fig. 4: Components of BioSys and their intercommunication.

essential to have such a layer. By doing so it did made the task of the BioSys development much easier. This makes it even simple for the application developers to work with the BioSys rather than with the BioAPI.

Considerable interest and effort was put onto make development as simple as possible. The decision was to make use of web services [5, 7-8]. Web services allow high level programming module with platform independence, centralised control with sufficient scalability. Use of web services (referred as the BioSys Service) makes it suitable for an enterprise level networked environment. It only requires applications (i.e. clients) to support SOAP (Simple Object Access Protocol) and HTTP messaging. Therefore any programming language that supports SOAP and HTTP can be used to develop applications that make use of the BioSys. Use of web services allows the BioSys to extend beyond a LAN or an intranet into the Internet. This is useful in cases where remote users and mobile users want to get authenticated using biometrics.

The Console (Fig. 4) is a separate management station, which can either reside on the same machine as the server or in a different one. It is the place where all the policies are defined and monitoring is done. All the user information, management policies, user biometric records, and events are stored in a centralised database and could be extended into a distributed database system if required.

Use of web services allows applications and Console to be of any platform however the server is to be limited to a specific platform. Web service was developed using Microsoft .Net C# and currently works only with Microsoft IIS (Internet Information Systems).

Microsoft .Net C# was selected not just because it supports Web services but there were two other concerns. Firstly, it was not possible to use other languages (other than C++) to access complex data structures and pointers that the BioAPI extensively requires. Even support of C# is limited up to a certain level. The authors could have used C++, but writing web services in C++ is tedious and nothing much will be gained since this is a research prototype. The second concern was the performance offered by the compiler.

It is unrealistic to ask a biometric device to support web services and send whatever it captures to the server for the processing. It is highly encouraged that image processing to be carried out in the server because it is secure doing it at the server. It will also reduce the processing overhead of the device (most devices do have limited processing power). Devices make use of socket communication to communicate with the server. System should also support RS232/485 protocols if it to be commercially successful.

6.0 Future directions

The developed system only supports some of the very basic network and security practices. It should be redesigned addressing security from bottom-up to manifest high-level of security. This is not because the current design is bad but because security should never be a separate layer and should always be an integral part of the whole system. These things were left out in the prototype due to resource limitations.

Current biometric infrastructure of an organization consists of lots of non-BioAPI compliant devices. If these devices can be transferred to become BioAPI compliant it would reduce a lot of reinvestment. BCB Generator (BioAPI Compliant BSP Generator) is such an approach where it tries to automate the processes which could transform non-BioAPI device to a BioAPI compatible device.

The Universal BioSys can be easily extended to the Internet with enhanced security because it is already uses web services. As with many biometric systems, the Universal BioSys can also be used as a time and attendance system. Necessary data is already available within the system and it is just a matter of organizing them.

The BioSys could also extend into a ticketing system like Kerberos or integrate with Domain management system such as Microsoft Active Directory [9]. To enhance security certain vendors uses a combination of biometric technologies (example: combined recognition systems with face, lip movement and voice). The Universal BioSys could support such mechanisms as well.

7.0 Conclusion

As the Universal BioSys is a Proof of Concept there are no measurable results. It has proven that biometric integration can be made seamless and effortless while enforcing management and security practices. It offers biometric technology, platform, and vendor independent third-party authentication. The system also reduces the TCO considerably and enforces tighter security with two unique features. The solution would be more suitable for medium to large-scale organizations with stringent security requirements that require installation of many biometric devices. Authors work could be used as a framework in developing next generation biometric based third-party authentication systems.

8.0 Case Study

Consider a hypothetical software company that is developing high quality software products and involved in top class research. Assume that its floor arrangement is

something similar to Fig. 2. Suppose they would like to have a balance between tighter security and TCO. If the organization is concerned in physical access to the building and servers they may install different type of devices at different locations as indicated in Fig 2. This sort of device placement will produce multiple access levels which can be mapped into a hierarch of devices.

Altogether they may install 8 devices (*D1* to *D8*) and device *D8* will be shared by the 2 servers. If they had ten servers on the server room they still need only one device which can be shared by all the servers. Therefore the total investment is less than having a dedicated device for each and every server. It is also possible to incorporate all the hosts with the BioSys, if necessary in future. If the organization has 100 computers which are used for development and R&D it does not need 100 biometric devices. With the BioSys approach in minimum it will need only 2 devices: one for the IT department and another device for the R&D. Preferably they may need to install few more devices for user convince and fault tolerance. Because they have already invested in the BioSys it will reduce cost from 90% (10 devices) to 98% (2 devices).

When using the BioSys for the first time the administrator has to draw a floor arrangement similar to Fig. 2. Then based on that he/she can indicate where different doors, rooms, departments, hosts, servers, applications, etc. are located. Then it is required to install biometric devices and register them with the system as well. Thereafter the Device-Hierarch has to be defined using a graphical interface similar to Fig 3. Finally users have to be enrolled to use applications. Enrolment involves; a particular user, an application, and a device class. Users are enrolled based on a particular class of devices rather than to an individual device (i.e. there are device classes such as fingerprint, facial, iris, etc. and if a user is enrolled to fingerprint class he/she can use any fingerprint device). This is referred as single point of enrolment where a user is enrolled once for all the devices belong to the same device class.

When things are configured correctly system will identify potential users and allow them access based on predefined policies. However, if someone tries to bypass the Device-Hierarchy either purposefully or by accident, it will not allow any access although the given credentials are valid.

Acknowledgement

The authors thank Dr. Chathura de Silva who was the project supervisor as well as the final year project coordinator. Authors would also like to thank Dr. Sanath Jayasena, the Head of Department of Computer Science and Engineering and other staff members. Finally authors would further like to thank their peers.

References

[1] The Universal BioSys web site. Available: www.cse.mrt.ac.lk/~dilumb/Documents/BioSys/

[2] The BioAPI Consortium, "BioAPI Specification Version 1.1", 16 Mar. 2004, Available: http://www.bioapi.org

[3] The Independent Security Server, Info Data, Inc., Available: http://www.infodatany.com/independentsecurity server.htm

[4] The WhoIsIt biometric server for E-commerce, Available: http://www.qvbiometrics.com/E_Metrics_server .htm

[5] De Silva S. M. R. P, Weerasinghe P. W. H. D, and Bandara H. M.N. D, "Universal BioSys - A literature review", Available: www.cse.mrt.ac.lk/~dilumb/Documents/BioSys/

[6] De Silva S. M. R. P, Weerasinghe P. W. H. D, Bandara H. M.N. D, "Universal BioSys", final project report, Available: www.cse.mrt.ac.lk/~dilumb/Documents/BioSys/

[7] Lakshmi Ananthamurthy, "Introduction to Web Service" Available: http://www.developer.com/services/article.php Heather Kreger, "Web Services Conceptual Architecture (WSCA 1.0)", IBM Software Group, May 2001, Available: http://www-306.ibm.com/software/solutions/webservices/pdf /WSCA.pdf

[8] "Active Directory Overview", 30/06/1999, Available: http://www.microsoft.com/windows2000/server/evaluation/ features/dirlist.asp