Self-built grid

Ph. D. Student, Eng. Eusebiu Marcu emarcu@softwin.ro

SOFTWIN, Research and Development Department, No. 1/VII Pipera-Tunari, Nord City Tower, Voluntari - Ilfov, Romania

Summary. This paper introduces a new type of grid and gives an example of usage of this new grid for an authentication mechanism using the biometric holographic signature. We start by stating the definition of this new type of grid and some of its properties; after that, we will define a new biometric authentication architecture that uses this type of grid using Windows Communication Foundation Application (WCF Application) as the main management core for client's requests for authentication and finally, we shortly describe the biometric signature authentication process.

1 Introduction

In today's highly connected world, everyone who is on-line is exposed to the common threats. The need of highly secure systems and communications is increasing each day. But rare are the cases that software architects or software development teams are trying to create or develop secure and with a high level of performance systems, frameworks or applications.
In this paper we propose a different biometrical authentication architecture (defined in [1]) and a new grid computing architecture - we will name it *self-built grid* - that will be defined, analyzed and used for a biometric authentication mechanism. Current architectures are [1]:

- Store on Server, Match on Server
- Store on Client, Match on Client
- Store on Device, Match on Device
- Store on Token, Match on Server
- Store on Token, Match on Device
- Store on Token, Match on Token

In the end, we will discuss about some methods to create this architecture more secure and show the experimental results.

2 Grid computing

"Grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed "autonomous" resources dynamically

at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements".(definition in [2]). Even if the first definition of the grid [3] involves the computational aspects - "a computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive and inexpensive access to high-end computational capabilities" - the grid computing architecture is also related with resource sharing and management and high performance.
In the past, grid computing was only used on solving problems in different institutions, from university to medical research institutions. The cost of a grid is very high just because the cost of the computers that are nodes inside the grid is very high.
In nowadays, this type of architecture is evolving towards a more high-level architecture known as Service-Oriented Architecture (SOA) benefiting from SOA intrinsic properties and implementation that uses Web Services standards and protocols (like SOAP [4], REST [5], WCF [6], etc.). Grid computing is used today as a very elegant solution to the new problems involving the large number of requests on the current applications and services.
Of course, grid computing can be used for an e-application depending on the purpose of the application. Entering in World Wide Web World, the application must be secure and must have a high level of performance. One of the security layers is authentication and inside World Wide Web World, we will call this e-authentication. There are a number of e-authentication methods like passwords, PINs, physical crypto tokens, one time password devices and biometrics.

There are many types of scenarios where an application can use a grid. The common scenario is where a group of computers solves a problem - scientific or technical. These computers are interconnected using high-speed connections and often are in the same location.
Other scenarios can also involve grid computing concepts but the location of the computers that are part of the grid is not in one place. An example is cloud computing where the computers are connected to the Internet and are consuming some web applications or services. So, cloud computing can be seen, in some way, the evolution of grid computing.

The biometrical authentication mechanism involves specific algorithms for stating the identity of the person that uses this type of authentication. Some algorithms depend on biometric technology. In any case, these algorithms are time and resource (CPU and memory) consuming. In an e-application scenario where a large number of clients are trying to authenticate using a biometric technology is clear that the usage of a grid computing architecture is absolute necessary. The only drawbacks are the high costs of implementing a grid and maintain it throughout the application's life-cycle.

2.1 Self-built grid

In this section, we will define the concept of self-built grid and state some this type of grid properties. A self-built grid is a grid where clients are nodes inside the grid and take part in solving the grid's task. In our case, for an application that uses e-authentication, the task that must be solved by the grid is authenticating, in an acceptable time, all clients' authentication requests. This task can be divided in a

number of smaller tasks.
Figure 1 shows a self-built grid at a time moment.

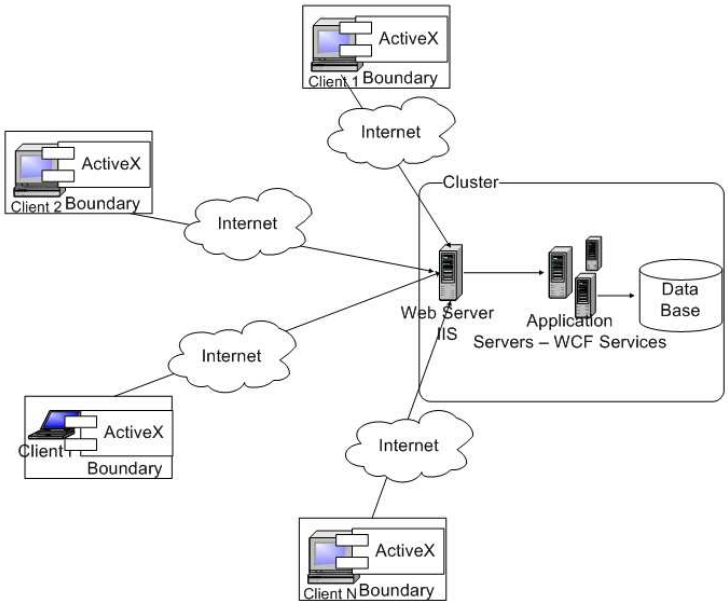

Fig. 1. Self-built grid at a time moment

As shown in the figure, each client is connected to a web server that sends the request to some application servers, grid's principal nodes - main nodes or management nodes - and each client connected to the web server is a common node inside this grid. For computation purposes, each client has an ActiveX object installed that will make all computations.

In this scenario, the self-built grid is actually a cloud application. But in an Intranet scenario, the self-built grid is classic grid.

2.2 Self-built Grid Properties

There are a number of properties for this type of grid:

- Grid's size - number of nodes - varies in time
- A new client becomes a node inside the grid - becomes a common node and increases the grid load
- A new client node - common node - takes over the task of loading the grid
- A client who is to meet the demands leaves the grid automatically - grid unloading
- If a client node leaves the grid, neither the grid nor the main job in influenced. The remained tasks are still computed

If the number of client's requests is an even number, in the

$$2 \cdot k, k > 0 \tag{1}$$

form, then all tasks (the number of tasks equals k) are distributed by main nodes to the common nodes for solving. If the number of clients is an odd number, in the

$$2 \cdot k + 1, k \geq 0 \tag{2}$$

form, then k tasks are distributed by main nodes to the common nodes for solving and one task is solved by the main node.

3 A new biometric authentication architecture

There are a number of authentication architectures (defined in [1]) including the following ones: match on server (store on server or on token), match on client (store on client), on token (store on token), match on device (store on device or token). Each of these architectures has their advantages and disadvantages. For match on server - store on server, there'll be a problem when the number of simultaneous requests made by clients is very large. For match on client - store on client, can exist security issues.
Having in mind self-built grid architecture model, we define a new biometric authentication architecture.

3.1 Cliet-Server-Client

This architecture model means that a template (under BioAPI 2.0 Standard definition [7]), of a number of samples, is stored inside a database in some format and the matching is made on client's machine. If the client makes a request and the matching with the template is made on the same machine, there may be some security issues. Knowing that the client nodes inside a self-built grid architecture can solve a task (in this case a matching between a template and a biometrical sample to match), we define the new authentication architecture model in the following manner: a client solves the task for authenticate other client (Figure 2). We can call this type of authentication, authentication in **three** points.

The workflow is the following:

- Two or more clients want to authenticate themselves against the system at the same time or in a very short interval
- Clients requests are taken by the server, the server associates an ID to that request (the ID may be a Globally Unique Identifier), a Timestamp (for timeout usage) and adds this newly created structure into a queue or a more developed structure
- The queue is verified to see if there's only one request and in this case the server will process the request and will authenticate the client and the answer will be sent to the client

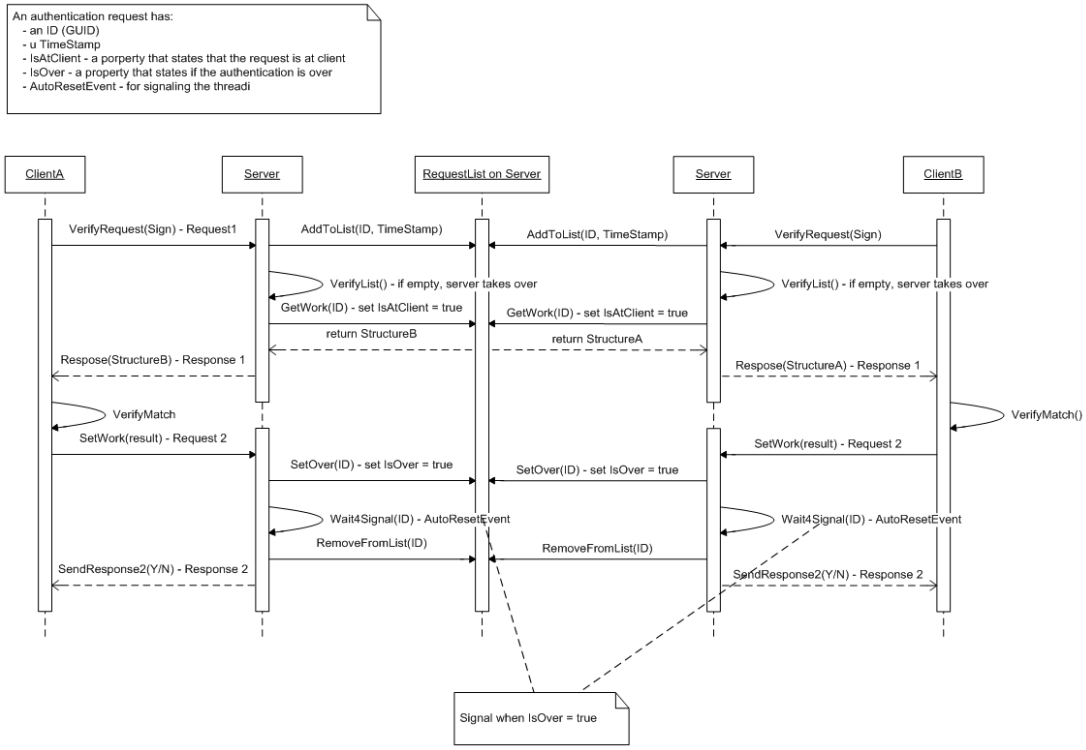

Fig. 2. Self-built grid at a time moment

- The thread that inserts the structure (Client's A Structure) in the queue removes a waiting request (the structure contains all necessary data for authentication, like clients template and sample to authenticate and set the *IsAtClient* property to true), request which belongs to other client (Client's B Structure). The server sends this client's B structure for authentication (solving the smaller task) in Client's A first response. If client A authenticates client B this not means that client B will authenticate client A. In this manner, it is obvious that client's requests are crossed-over for solving
- The authentication is made inside client's A installed ActiveX and creates a new request for the server to send the answer
- The newly created thread sets the result of the structure in a waiting queue according to the request ID - sets *IsOver* to true and signals the clients' A waiting thread to complete its second request
- Final step, the answer is sent to the client A.

In this workflow may appear some exceptions. One of this is the timeout exception. This exception can appear when either the client has some problems communicating with the server or the authentication process takes too long. In this case, if the timeout value has passed, the server automatically sends the request to other client for processing or solves himself the task. There is a limited number of re sends. If

this number is achieved, the server will solve the task.
It is clear that is this model the server becomes only a manager of client's requests and it will intervene when an exception is thrown or the number of resends is achieved.

3.2 And More Performance

According to [7], a "biometric sample" is "information obtained from a biometric sensor, either directly or after further processing". A "biometric template" is a "biometric sample or combination of biometric samples that is suitable for storage as a reference for future comparison". A "processed biometric sample" is a "biometric sample suitable for comparison". Therefore, a "biometric sample" is created when enrollment action takes place and is used in a matching operation.
Let us assume that there is no way that a "biometric template" can be transformed again into the same "biometric sample" that was captured by the biometric device. For enrollment and authentication operations, a biometric sample must be processed. To increase the system performance we can "move" the biometric sample processing operation on client side and consequently, decreasing server computation work. Adopting this model, the system works only with processed samples and increase data security. In Figure 2 we show an example of how a template can be created.
In short, the client opens the acquisition application (which hosts the ActiveX object), the application loads a BioAPI Engine (BioAPI Service Provider - BSP), an user-interface appears in front of the client, the client give a number of samples, the application sends these samples to the engine for processing and the engine creates a biometric template or a processed sample.

3.3 Advantages

The proposed architecture has the following advantages:

- The server will be unloaded by all clients' authentication tasks. For an authentication, to maximum time will be the sum of : Client's A connection time, Client's B connection time, Number of resends * (timeout time + new client connection time) - task time if the server computes the task;

  $$t_{total_{max}} = t_{connectionA} + t_{connectionB} + n_{resends} \cdot (t_{timeout} + t_{connectionC}) + t_{server} \tag{3}$$

  where

  $$t_{connectionC} \tag{4}$$

  is the connection time of new client C that gets client's A task. The connection time is different from one client to other and includes the request and response time between that client and the server.
  The minimum time is given by:

  $$t_{total_{max}} = t_{connectionA} + t_{connectionB} + t_{effective} \tag{5}$$

- The server will be unloaded by all clients' enrollment/process operation.

- Usage of "processed template" format increases the data security - the original data cannot be rebuilt using the sample features. Even if someone succeeds capturing a template, the attacker cannot do anything with it because he or she doesn't know how template is it.
- When a client computes other ones task, there is no information what so ever about the client that wants the task solved

3.4 Disadvantages

The proposed architecture has the following disadvantages:

- The presence of the authentication engine on the client machine makes possible the observation of operations made by the engine. A possible countermeasure is the online downloading of the engine.
- There is a vulnerable point before the template is created - there is a possibility that an attacker can break into the engine and save the biometric samples.

3.5 Notes about security

Even if this type of grid is used as a clasic grid inside a Intranet network or it is used as a cloud application, security is important. There are many types of attacks that can be launched against a system that is using this architecture. The following table contains some types of attacks.

Table 1. Types of attacks

Attack	Countermeasure
man in the middle	encrypting the network traffic
spying on application process	encrypting data on client, derypting on server
spoofing	using SSL certificates for authentication

One of the most secure and reliable authentication methods is certificate based authentication. To create a certificate based authentication, a Public Key Infrastructure (PKI) must be implemented using a PKI software from specific vendor. Once the PKI software is installed, a Certificate Authority (CA) exists as a part of the PKI software. The CA can issue or revoke certificates for a number of clients. A web server can specifically request for the client's certificate. If the client does not have a certificate issued by the CA, the connection is closed (the client was not authenticated). If the certificate exits, the network traffic in encrypted using Secure Socket Layer (SSL).

4 Experimental results

In order to test a system that uses this architecture, a two types of applications were created: a WCF application as the management core and a console application

simulating the client. The console application was installed on a number of 20 computers and, at a given time, a authentication request was sent (all computers were syncronized). The following image shows a state of the authentication mechanism.

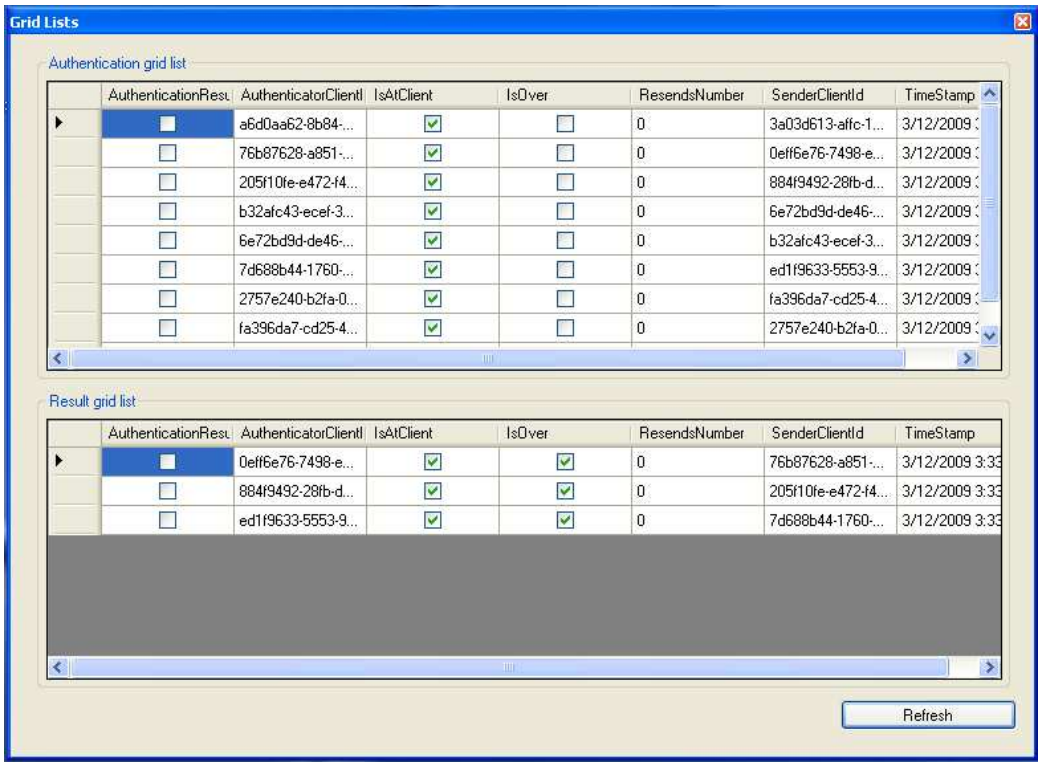

Fig. 3. Self-built grid at work

Before this system, we had a classic "Store on Server, Match on Server" arhitecture. The grid method and the classic method are compared below (on X-axis is the number of nodes, on Y-axis is the time, in seconds).

The test was made using the same signature for all tests. Therefore we get an almost "perfect" linear variation.

5 Conclusions

This paper's new definition of a grid tries to create the base point of a new type of distributed computing. Of course, this type of grid isn't widely applicable. The architect that will use this type of architecture must obey the following rules:

- The grid's main task can be divided in a number of smaller tasks and solving a smaller task will decrease grid's main task complexity

Fig. 4. Self-built grid vs. Classic method

- A smaller task can be solved by a common node

The new authentication model is derived from match on server - store on server model, the only difference is that the second client solves the task (as it was a part of the server). So, from client's point of view, the server is a multi-machine system.

References

1. International Committee for Information Technology Standards, 2007, Study Report on Biometrics in E-Authentication, 30 March 2007, INCITS M1/07-0185
2. Grid Computing, Grid Computing Info Centre http://www.gridcomputing.com/gridfaq.html
3. I. Foster, C. Kesselman, The grid: Blueprint of a New Computing Architecture, Morgan Kauffman Publishers, San Francisco, CA, 1999
4. Simple Object Access Protocol, World Wide Web Consortium, SOAP Version 1.2, http://www.w3.org/TR/2007/REC-soap12-part1-20070427
5. Representational State Transfer, www.w3.org/2005/Talks/1115-hh-k-ecows
6. Windows Communication Foundation, Microsoft Co., http://msdn.microsoft.com/en-us/netframework/aa663324.aspx, 2007
7. BioAPI Consortium, Information technology - Biometric application interface - BioAPI Specification, ISO/IEC 19784-1:2005