SOLUTIONS OF SPEEDING-UP ON-LINE DYNAMIC SIGNATURE AUTHENTICATION

Valentin Andrei, Sorin Mircea Rusu Ștefan Diaconescu,
Research & Development Department, SOFTWIN S.R.L., Sos. Pipera-Tunari, Bucharest, Romania
vandrei@softwin.ro, srusu@softwin.ro, sdiaconescu@softwin.ro

Keywords: biometrics, dynamic signature, FPGA accelerating, GPU computing, Levenshtein algorithm, parallel computation

Abstract: The article presents a study and its experimental results, over methods of speeding-up authentication of the dynamic handwritten signature, in the context of numerous requests. We will present 3 solutions that use parallel computation by using a 16 processor server, a FPGA development board and a video board, using nVidia CUDA technology.

1 INTRODUCTION

The evolution of biometry in the last few years was predictable due to the increasing amount of sensible data, like bank accounts information, new technologies documentation, etc., that needed to be kept into databases, safe from any attack attempt. The Internet held more and more valuable information so the need for high security kept increasing.

Biometry appeared relatively recently and used unique characteristics of a person in order to secure and authenticate his actions. Some of these characteristics are: the iris, the fingerprint, the signature, the voice, the face anatomy, etc. One of the most non-intrusive methods of biometric authentication is using the dynamic handwritten signature. This does not represent only the image drawn on a piece of paper, but it refers mainly to the movement of the owner’s hand. The image can be copied but the movement of the hand is almost impossible to be imitated. This type of characteristic belongs to behavioural biometrics because a person changes his way of signing across years. Due to this oneness, the offered level of security is very high.

In the last 2 years, we have focused our research towards using dynamic signature in the purpose of on-line authentication. We have built a secure web-service, capable of securing internet applications that needed authentication. The problem that rises is how we can provide a short authentication time when the server is being stressed with numerous requests. In this article we will present 3 solutions that can help us solve the problem.

2 AUTHENTICATION

In order to verify the originality of a given signature, some processing needs to be made. First of all the user needs to input 5 signatures, that will be considered templates and every new signature will be compared to these templates.

An electronic device will be used to capture the signature. This electronic device is an intelligent pen, that contains 2 MEMS accelerometers and an optical navigation system (ONS) having the ability to extract the user’s hand movement and to transmit it via USB port. The pen transmits at 1000 Hz sampling rate, the hand acceleration values on 2 axes and the data extracted by the ONS. Inside the PC, the raw signal is being stored in CSV files.

These files contain a fair amount of redundant data so in order to make the process more efficient we need to apply a compression method. For this reason we have developed a set of signature recognition algorithms (SRA) that build a set of
invariants, from the raw signals. The following picture represents the signature processing.

![Signature processing steps](image1)

Figure 1: Signature processing steps

In order to say if a signature is original, we must compute a distance between the invariants extracted from the given signature and the invariants extracted from the user’s templates. The distance is being computed by using the Levenshtein algorithm. After that, the resulted distance, is being passed to a classifier that will give the final answer over the originality of the signature. The next picture shows the dynamic signature authentication process.

![Signature authentication process](image2)

Figure 2: Signature authentication process

As we mentioned before, we have built a web-service, capable of providing biometric authentication based on the dynamic handwritten signature. The acquired signature is being sent to the web-service, where the matching against the stored templates is made. The service just sends the response, telling if the given signature is original or false. If the signature is original, the user receives access to his account. Given this context, a proportion of the authentication time belongs to the transfer operation, between the client and the web-service. However we are interested in accelerating the most time consuming operation, of the whole system. The next diagram shows the main operations that are done, in order to offer a client access based on his dynamic signature.

![On-line signature authentication](image3)

Figure 3: On-line signature authentication

We have measured the time consumed by each operation, in order to find the component whose function needs optimizations. The following table shows percentages of the authentication time, consumed by every system block.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Time (% of total)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signature acquisition</td>
<td>&lt; 1%</td>
</tr>
<tr>
<td>Transfer time</td>
<td>&lt; 2%</td>
</tr>
<tr>
<td>Invariants computation</td>
<td>&lt; 5%</td>
</tr>
<tr>
<td>Distances computation</td>
<td>&gt; 92%</td>
</tr>
</tbody>
</table>

As we can see from the previous table, the Levenshtein algorithm is the most time consuming task of the authentication process. Therefore we need to find a solution in order to compute the distances as fast as we can, to provide the client a reasonable authentication time.

The Levenshtein algorithm is based on the following formula:

\[
D[i, j] = \text{Minimum} (D[i-1, j] + \text{Deletion Cost}, D[i, j+1] + \text{Insertion Cost}, D[i-1, j-1] + \text{Substitution Cost})
\]

(1)

It involves the usage of a matrix of N x M size where N and M are the lengths of the strings being compared. The parallelization possibilities of a single algorithm instance are poor. However parallelizing is possible by using systolic arrays but
increases the implementation difficulty. If the length of the strings is large, then the parallelization of the algorithm worths being implemented. If this length is small, but the requests are numerous, then it is advantageous if several instances of the algorithm are being computed in parallel.

We have chosen to distribute distance computing tasks into several processing units. We have studied the possibility of using the following:

- 16 processor powerful server;
- FPGA development board;
- Video card using nVidia CUDA technology.

3 ACCELERATING

As said before, we need to start several instances of the Levenshtein algorithm in multiple processing units. We focused our work only to studying how we can speed-up the distance computing block, so this is the reason we have built a block that generates strings of symbols, to be passed as input data to the processing units. We have measured the time it took for the whole block of data to be processed and we compared the 3 solutions. The following figure gives an overview over the testing procedure.

![Testing procedure diagram](image)

The strings generator generates a queue of pairs of strings. Each pair will serve as input for an instance of Levenshtein algorithm. This queue will be a symbol matrix of \((2 \times N) \times M\) cells where \(N\) represents the number of distances that need to be computed, and \(M\) represents the length of the strings being compared. In the real system we will provide a mechanism of queuing for the authentication requests. The request queue will be very similar to the matrix generated by the strings generator so the estimations we will make based on the results the 3 solutions give, will be close to truth.

3.1 Accelerating the authentication process using a 16 processor server

The goal is to use the server hardware capabilities at full power, in order to achieve the best time. We have done this by starting a number of threads in our main program and by assigning them a high priority among the tasks of the operating system.

First of all we needed to discover the optimum thread number in order to minimize the overhead. For that we have computed 2048 distances between strings of 750 symbols on 32 bits each. We have distributed these tasks to a variable number of threads and measured the computing time. If the number of threads is too small, then the computing power of the server is not used at maximum, which will result in a time increase. If the number of threads is too high, then the generated overhead will also result in a time increase. The average computing time for a variable number of threads is synthesized in the following table.

<table>
<thead>
<tr>
<th>Number of threads</th>
<th>Duration (milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>2281</td>
</tr>
<tr>
<td>14</td>
<td>1328</td>
</tr>
<tr>
<td>16</td>
<td>1412</td>
</tr>
<tr>
<td>32</td>
<td>1515</td>
</tr>
</tbody>
</table>

This behaviour was predictable because the server has 16 processors and the optimum thread number should be near 16. The resulted number of threads that would give best results is 14. In this case 14 of the system's processors will be used at full power and the rest of 2 processors will be used for the vital tasks of the operating system.

For the system to offer a short response time, the processor frequency must be as high as possible, and also a key element is the RAM frequency. If the frequency is too low, then the reading and writing from and into the memory will require a high amount of time, seriously slowing down the process. However, caching mechanism should be also available on each processor but all the generated data can’t fit into the cache memory so that is why the RAM frequency is an important element. The on-line authentication system, using the first solution, is drawn in the following image:
The system will host a web-service that receives all the client's requests, and all the distance computations are done inside the server. The web-service can be hosted on a different station. The disadvantage of such a system is that a powerful server is very expensive and its purpose is for general usage, rather than to be used exclusively for authentication purposes.

Therefore we need a solution cheaper than a powerful server, and designed specifically to accelerate the authentication process.

3.2 Accelerating the authentication process using a FPGA development board

FPGA development boards have become more and more used lately, especially where computational power is needed, in systems with an increased computational complexity. Systems composed of several FPGA boards are used for example in biotechnology, performing sequence alignments, proteins matching, docking, etc.

The acronym FPGA, stands for Field Programmable Gate Array, which is a chip, that contains a matrix of elementary electronic circuits, that can be combined to build a complex function. A digital circuit systems designer can implement inside a FPGA board, a processing unit that can perform one specific algorithm. He will design the blocks of the system, will write their code in a hardware description language and will synthesize the system on the chip, using a synthesizer like XilinX for example.

One key element for a system working on a FPGA is the achieved frequency. If it is high, then the system will be efficient. To increase the frequency, the delay from a combinational circuit’s input to its output should be as small as possible. A combinational circuit does not work by using a clock input and it performs its function asynchronously.

One feature the FPGA board should have is the amount of block-RAM or if possible it should have external RAM blocks. The block-RAM is the RAM cells located inside the FPGA chip, and its access frequency is the system’s working frequency. To use external memory chips, frequency adapters are needed and not always these blocks can be used at full speed.

The number of elementary circuits inside the FPGA is also crucial. If the chip has a large number of gates, then we can implement several processing units inside a single chip that perform in parallel. The following picture, describes the general architecture of a system composed of several Levenshtein processing units, implemented on a FPGA chip.

The system can compute multiple Levenshtein distances in parallel, on a single chip. If we need more speed, we can build a custom hardware device composed of several FPGA boards that work independently, controlled by a processor. However this solution raises implementing problems so it’s preferable to just use FPGA-s connected to a single computer that will distribute processing tasks to them.

The following table presents the time needed for a number of comparisons, depending on the system’s frequency. We assumed that our FPGA board will host 10 processing units. We calculated the time needed to compute 2050 distances between strings of 750 symbols, 32 bits each. We intend to make a comparison between the 3 proposed solutions so the number of comparisons should be the same for all systems. Of course we can approximate 2050 with 2048.
Table 3: Computing duration using a FPGA board

<table>
<thead>
<tr>
<th>System's Frequency</th>
<th>Used BUS</th>
<th>Duration (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50 MHz</td>
<td>USB 2.0</td>
<td>22700</td>
</tr>
<tr>
<td>100 MHz</td>
<td>USB 2.0</td>
<td>11448</td>
</tr>
<tr>
<td>150 MHz</td>
<td>USB 2.0</td>
<td>7700</td>
</tr>
<tr>
<td>200 MHz</td>
<td>USB 2.0</td>
<td>5800</td>
</tr>
<tr>
<td>300 MHz</td>
<td>USB 2.0</td>
<td>4500</td>
</tr>
<tr>
<td>300 MHz</td>
<td>PCI Express</td>
<td>4200</td>
</tr>
</tbody>
</table>

A frequency of 200 MHz is achievable even by using a low cost FPGA, such as a Spartan 3. More preferment boards such as Virtex 5 can achieve even a higher frequency.

We can also see that the used BUS does not influence considerably the response time. An on-line authentication system using FPGA boards to perform Levenshtein distances is described in the following figure. Besides the Levenshtein processors, a BUS controller must be implemented inside the FPGA, to synchronize the system with the computer BUS and to realize the data transfer and also a software driver needs to be made in order to command the system.

Figure 7: Authentication system using multiple FPGA boards

3.3 Accelerating the authentication process using CUDA enabled video cards

A solution used in the last few years for high computing processes is using graphic cards computing. nVidia producer launched the CUDA architecture and also a software development kit, that allows programmers to use the graphic card’s capabilities at full power. For example, a developer familiarized with the C language, can easily write C code for CUDA, respecting a number of conventions and that code will be computed by the graphics card. CUDA is very suitable for algorithms that can be parallelized, matrix multiplications, etc. In the proposed solution we will use the graphic card by launching multiple Levenshtein distances computing instances in parallel.

The CUDA architecture is described in the following image. Each graphics card chip is composed of a number of multiprocessors each containing a number of 8 streaming processors. Fast shared on-chip memory is available to use and also the board offers a high amount of RAM memory connected externally, called device memory.

Figure 8: nVidia CUDA architecture

The device memory is the slowest memory available. The streaming processors can read and write it but at great time cost. The shared memory is the best solution to use when time is critical. Each streaming processor also has a number of fast registers that can also be used for optimisations.

We used the same evaluating method as for the dedicated server solution, by generating a matrix of input strings. We have copied the matrix into the device memory and then launched threads on each streaming processor of each multiprocessor. The operations needed to perform the comparisons are mentioned below:

- Start Timer
- Copy strings matrix into device memory
- Fill the shared memory of each processor
- Launch Levenshtein algorithm on several threads
- Copy distances into device memory
- Copy distances from device memory to system RAM
- Stop Timer

The results we have obtained by using this solution are presented in the following table. We have used 3 CUDA enabled graphic cards of
different computing capabilities. We initiated the comparison of 2048 pairs of strings, 750 symbols of 32 bits each.

Table 4: Computing duration using nVidia CUDA enabled video cards

<table>
<thead>
<tr>
<th>Video Card</th>
<th>Capabilities</th>
<th>Duration (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>nVidia Quadro NV 135 M</td>
<td>2 Multiprocessors</td>
<td>23625</td>
</tr>
<tr>
<td>nVidia GeForce 9500 GT</td>
<td>4 Multiprocessors</td>
<td>7266</td>
</tr>
<tr>
<td>nVidia GeForce GTX275</td>
<td>30 Multiprocessors</td>
<td>1390</td>
</tr>
</tbody>
</table>

An on-line authentication system, using several graphics cards to process distances between invariants strings, will have the architecture presented below.

Figure 9: Authentication system using multiple nVidia CUDA enabled video cards

The communication through the PCI Express BUS is transparent to the developer because it is realised by the provided CUDA driver. The implementing problems are similar to ones that rise when implementing the server solution. The main advantage of this solution is that the video cards are relatively cheap and their applicability area is rather large.

4 CONCLUSIONS

We have presented 3 solutions of accelerating on-line authentication, by using dynamic handwritten signature. We have presented the signature processing that is made, and we have shown 3 methods of speeding-up the most computational blocks. The following table synthesizes the results we have obtained.

We have considered a system using a 16 processor powerful server, one using 3 USB 2.0 FPGA boards with 10 processors each, working at 300 MHz and also a system using a single nVidia GeForce GTX275 video card.

Table 5: Comparison between the proposed acceleration solutions

<table>
<thead>
<tr>
<th>Solution</th>
<th>Price</th>
<th>Duration (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 Processor Server</td>
<td>~8000 USD</td>
<td>1328</td>
</tr>
<tr>
<td>3 FPGA Boards on a PC</td>
<td>~2000 USD</td>
<td>1400</td>
</tr>
<tr>
<td>1 nVidia GeForce GTX275 video card</td>
<td>~300 USD</td>
<td>1390</td>
</tr>
</tbody>
</table>

As we can see from the table above, the solution that should be used is obvious. Motherboards build using nVidia SLI technology allow up to 3 video cards on one single system so the speed achieved can be highly improved.

Given this context, our on-line system of authenticating users using their dynamic signature, can respond to a number of around 100 requests per second which makes it a high security feature needed to be considered.

REFERENCES