A Secure Distributed Video Surveillance System Based on Portable Devices

Pietro Albano[1], Andrea Bruno[1], Bruno Carpentieri[1], Aniello Castiglione[1], Arcangelo Castiglione[1], Francesco Palmieri[2], Raffaele Pizzolante[1]⋆, and Ilsun You[3]

[1] Dipartimento di Informatica "R.M. Capocelli"
Università degli Studi di Salerno
I-84084, Fisciano (SA), Italy
pietro.albano@gmail.com, andrea.bruno@antaresnet.org,
bc@dia.unisa.it, castiglione@ieee.org,
arccas@lug-ischia.org, rpizzolante@unisa.it

[2] Dipartimento di Ingegneria dell'Informazione
Seconda Università degli Studi di Napoli
I-81031, Aversa (CE), Italy
fpalmier@unina.it

[3] Korean Bible University
16 Danghyun 2-gil, Nowon-gu
Seoul, Republic of Korea
isyou@bible.ac.kr

Abstract. In this work a distributed video surveillance system based on a Client-Server architecture is presented. The proposed system is accessible from portable devices such as tablets, smartphones, etc. In a typical real-world scenario, for example in homeland security, it is useful to have portable devices that can receive in real-time a frame or a sequence of frames coming from a selected camera to prevent or to detect attacks (i.e. terrorist attacks, etc.). In the proposed system, a portable device knows only the address of the server (repository), and the repository sends to the portable device the list of the clients (nodes) which are connected with one or more cameras. When the portable device obtains the list of the nodes, it connects directly to a specific node and requests the images of its connected cameras. The whole system provides secure communication channel between all its components. The security of both the node-repository and the repository-portable devices communications is guaranteed by using a secure connection. The security of the node-portable devices interconnection is provided by a digital invisible watermarking algorithm that affects each image before sending it from the node to the portable devices. Each portable device can extract the watermark and verify the identity of the node.

⋆ Corresponding author: Raffaele Pizzolante, Dipartimento di Informatica, Università degli Studi di Salerno, Via Ponte Don Melillo, I-84084 Fisciano (SA), Italy, Email: rpizzolante@unisa.it, Phone: +39 089 969500, Fax: +39 089 969600

Keywords: Video Surveillance, Remote Video Surveillance, Remote Personal Security, Mobile Video Surveillance, Remote Homeland Security

1 Introduction

Video surveillance has become increasingly important in everyday use and video surveillance systems have become sophisticated and are increasingly accessible and usable by the general public. Important applications of video surveillance are the identification of individuals and objects as well as the prevention and the detection of abnormal activities. Video surveillance is also helpful in other fields such as agriculture (for the prevention of fires), etc.

With the diffusion of advanced *portable devices*, now it is possible to perform surveillance and monitoring activities using them.

Hence, in this work we propose a secure distributed system for video surveillance, based on a Client-Server paradigm, that introduces the possibility of remote connections from *portable devices* for real-time monitoring. The system architecture is based on several basic entities: a central server (or *repository*), which knows the locations of some collector *nodes*, connected with one or more *cameras*.

The *portable devices* accessing the system know only the location of the *repository*. When the connection is established, the *repository* sends to the *portable device* a list of *nodes* and, as a second step, the device connects directly with a *node* and receives from it the image frames or multimedia contents obtained by its camera(s). There is an high degree of security both in the communication among the system parts and in the video frames.

The remainder of this work is organized as follows: Section 2 discusses the distributed architecture of the proposed system. Section 3 focuses on the security aspects and describes the approaches for the engineering and development of the prototype we have developed. In Section 4 we describe the system from the point of view of the end user and in Section 5 we present our conclusions and highlight future work directions.

2 The Distributed Architecture

The main task of the *repository* is to allow the localization of the *nodes*. It interacts with them and with the *portable devices* accessing the system by maintaining a list of all the *nodes* that joined the system.

A collector *node* is connected with at least a camera and interacts with the *portable devices* interested in monitoring the areas covered by its associated camera(s). The *portable devices* need to know only the IP address of the *repository* that provides the IP addresses of the list of the available collector *nodes* and other descriptive information about them. When a *portable device* connects to a *node* it receives, at regular intervals, the images obtained by the specific camera(s) connected to the *node*.

Figure 1 shows a graphical representation of the proposed system architecture.

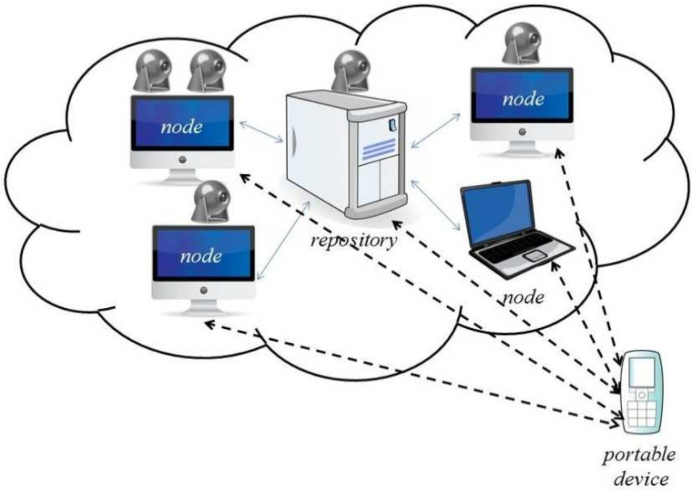


Fig. 1. The architecture of the proposed system.

In the following, we discuss the interaction among *node-repository* (see Section 2.1), *repository-portable device* (see Section 2.2) and *node-portable device* (see Section 2.3).

2.1 Interaction *node-repository*

The communication between these two entities takes place through a TCP connection over TLS/SSL. Its security is essential, since without a secure connection a fake *node* could easily enter into the system and send manipulated/tampered images to the accessing *portable devices*.

There are three types of messages exchanged between a *node* and the *repository*: *Login*, *Register*, and *Disconnect*. These messages are encoded by XML markup sequences. Figure 2 shows an example of a complete interaction between the above entities, including each of the three messages.

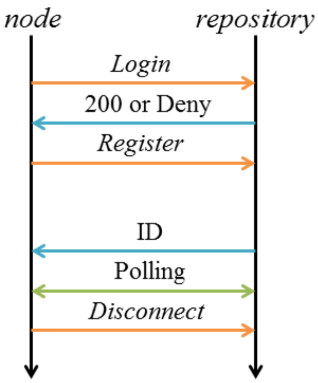


Fig. 2. An example of a complete interaction between a *node* and the *repository*.

The *Login* message is used by a *node* to authenticate itself to the *repository*. If the authentication succeeds, the *repository* answers with a return code "200", otherwise the *repository* denies the access for that *node*. Figure 3 shows an example of *Login* message from a *node* to the *repository*.

```
<?xml version="1.0" encoding="UTF-8"?>
<properties>
  <entry key="type">LOGIN</entry>
  <entry key="usr">username</entry>
  <entry key="pwd">password</entry>
</properties>
```

Fig. 3. An example of a *Login* message from a *node* to the *repository*

The *Register* message is used by a *node* to send its descriptive information to the *repository*. When the *repository* receives the information about the *node* it replies with an *ID*. The *ID* is also used by the *node* during the disconnection process. Figure 4 shows an example of *Register* message from a *node* to the *repository*.

```
<?xml version="1.0" encoding="UTF-8"?>
<properties>
  <entry key="type">REGISTER</entry>
  <entry key="descr">description</entry>
  <entry key="name">name</entry>
</properties>
```

Fig. 4. An example of a *Register* message from a *node* to the *repository*

When a *node* want to disconnect itself from the *repository*, it sends *Disconnect* message as shown in Figure 5.

```
<?xml version="1.0" encoding="UTF-8"?>
<properties>
  <entry key="type">DISCONNECT</entry>
  <entry key="id">idStr</entry>
</properties>
```

Fig. 5. An example of a *Disconnect* message from a *node* to the *repository*

2.2 Interaction *repository-portable device*

As previously, the interaction between the *repository* and the *portable devices* is based on a TCP connection over TLS/SSL. There are three new commands associated respectively to three different XML messages: *ListServer*, *Update* and *Desc*. The *ListServer* message requests the server to download the list of *nodes* currently registered. The reply of the *repository* can be either the "530" return code in case of error, or a string having this form:

$$//n\%ts\%\%\mathrm{ID}_1\%\mathrm{Name}_1\%\mathrm{IPAddress}_1\%\%\ldots\%\%\mathrm{ID}_n\%\mathrm{Name}_n\%\mathrm{IPAddress}_n\%\%//$$

where:

$\boldsymbol{n}$ is the number of the entries;
$\boldsymbol{ts}$ is a timestamp which is used for indicate when the request has been sent and is also used with *Update* command;
$\mathbf{ID}_i$ is an unique identifier associated to *node* i;
$\mathbf{Name}_i$ is the alias or the name of the *node* i;
$\mathbf{IPAddress}_i$ is the IP address of the *node* i.

The *Update* message is used by the *node* to communicate to the *repository* its intention to update the local list. The command *Update* takes as parameter the timestamp, used by the *repository* to verify if the list of *nodes* maintained by the *portable device* has been updated. The possible replies of the *repository* can be: a "400" return code if there isn't updates or, alternatively, a string structured as:

$$//n\%ts\%\%[+|\text{-}]\mathrm{ID}_1\%\mathrm{Name}_1\%\mathrm{IPAddress}_1\%\%\ldots$$
$$\%\%[+|\text{-}]\mathrm{ID}_n\%\mathrm{Name}_n\%\mathrm{IPAddress}_n\%\%//$$

where:

$\boldsymbol{n}$ is the number of the entries (n);
$\boldsymbol{ts}$ is a timestamp which is used to indicate when the request has been sent;
$+\mathbf{ID}_i$ is the *node* with the identifier IDi has been registered after the last request of update, *or*

-ID$_i$ is the *node* with the identifier IDi has been disconnected after the last request of update;

Name$_i$ is the alias or the name of the *node* i;

IPAddress$_i$ is the IP address of the *node* i.

2.3 Interaction *node-portable device*

The communication between these two entities is performed through the use of TCP sockets which adopt a FTP-like protocol for exchanging messages and data. The main difference between the FTP protocol and the proposed ad-hoc protocol is that in the former the client opens two communication channels (one for the messages and one for the data), in the latter, *portable device*, using a GPRS/UMTS connection, cannot open more than one channel due to commonly closed mobile operators policies.

In the proposed protocol the server (*node*) opens two sessions (with two different channels) and the client (the *portable device*) opens just one. The *node* communicates to the client the port of the other opened channel , resulting in two communication options. The three message used are: *Login*, *List*, and *MGet*.

The *Login* message is used by the *portable device* to authenticate. The Login operation can succeed or not. In the first case the *node* replies to the *portable device* with a message with return code "200" and the communication continues normally. On the contrary, the *portable device* receives an alert message on the display.

The *List* message is used by the *portable device* to request a snapshot of all the environments monitored by the *node*. When a *node* receives this message, it takes a snapshot from each of its camera(s) and sends it to the *portable device*.

The *MGet* message is used by the *portable device* to request the monitoring of a specific area, identified by an unique identifier. When this message is received by the *node*, it opens a data channel that is able to send the multimedia frames at regular intervals.

3 System Security

The development of secure architectures, providing controlled access, privacy protection, content confidentiality and authenticity, is one of the most challenging issues in the video surveillance area, and several solutions, based on the use of cryptography have been proposed (e.g. [1, 2, 3]). Furthermore, the existence of a surveillance system strongly depends on legal boundaries [4] that states what is allowed to be monitored, what is not, and also who is authorized to perform monitoring. In these cases the data produced by surveillance activities must be properly secured against unauthorized accesses or misuses of the collected images.

The proposed system has two important security aspects: the first one concerns the communication channel between a *node* and the *repository*, while the second one concerns the communication channel between a *node* and a *portable device*.

In the first case the secure connection between the *repository* and a *node* is guaranteed by using a SSL/TLS connection, providing privacy through symmetric cryptography and message reliability through keyed message authentication codes. Asymmetric cryptography is also used to protect each key exchange. This ensures the prevention against any type of eavesdropping and tampering.

However, some *portable devices* could not be able to implement, fully or partially, the aforementioned cryptographic mechanisms and primitives. For this reason we decided to use Digital Watermarking techniques to further guarantee security between a *node* and a *portable device*.

Therefore, each *node*, before sending an image modifies it with an invisible digital watermark. On the other hand, when the *portable device* receives the image, it extracts the watermark in order to verify the authenticity of each image.

3.1 Digital Watermarking to Improve the Security of the Proposed System

Digital Watermarking is one of the techniques generally used to insert hidden data into digital contents. When a signal is protected by a robust digital watermark, then the associated hidden information will be also carried in its copies. Watermarking is also used to prevent unauthorized copy of digital media.

There are different embedding methods: such as Spread-Spectrum [5, 6, 7], or Amplitude Modulation [8].

In the first one, the signal affected by digital watermark is obtained by an additive modification. Also in the case of amplitude modulation the marked signal is obtained by an additive modification, like in Spread-Spectrum embedding, but the watermark is only embedded in the spatial domain.

Before sending the images, the *node* embeds in them a digital invisible watermark. When the *portable device* receives the image it extracts the watermark in order to verify the trueness of each image.

The watermarking algorithm used in this work has been proposed in [9], and it is based on a modified version of the one proposed in *Langelaar et al.* [10].

It takes as input the source image, the watermark string, a seed and a threshold T.

The watermark string is converted in a bit matrix where each character is converted in a 5x8 sub-matrix of bits (an example is reported in Figure 6). The resulting bit string is obtained by reading the bit matrix line-by-line from left-top corner. The *seed* represents an ID (such as numeric PIN) that is used to embed the watermark, and in second instance to extract it from the watermarked image. The threshold T is a real number that indicates the robustness of the watermark that will be embedded.

The algorithm used for embedding a digital invisible watermark in an image is sketched as follows:

1. The image is converted from the **RGB** domain to the **YUV** domain.
2. The watermark string is converted to a matrix of bits. Each character is converted in a 5x8 matrix of bits (see the example in Figure 6).
   The resulting matrix will be embedded in the original image line-by-line from left-top corner.
3. A block B of 8x8 pixels is pseudo-randomly selected from the image to embed one bit of the watermark string.
4. A fixed binary pseudo-random pattern of the same size of B is generated.
5. The I_0, I_1 and D quantities are calculated from B. I_0 and I_1 are obtained by calculating the averages of the luminance values in B, respectively where the random pattern is 0 and where the random pattern is 1. D is the difference $I_1 - I_0$.
6. B' is a reduced quality block obtained by applying the quantization and the 8x8 DCT transform.
7. The I'_0, I'_1 and D' quantities are calculated from B'. I'_0 and I'_1 are obtained by calculating the averages of the luminance values in B', respectively where the random pattern is 0 and where the random pattern is 1. D' is the difference $I'_1 - I'_0$.
8. If the bit to embed has value 1 the go to step 11.
9. In order to embed the **bit with value** 0, the binary pseudo-random pattern is subtracted from the block B, if D and D' are greater than the threshold T. The steps 6-8, and 10 are repeated iteratively until both differences are less or equal than $-T$. Go to step 12.
10. In order to embed the **bit with value** 1, the binary pseudo-random pattern is added from the block B, if D and D' are less or equal than the threshold T. The steps 6-8, and 11 are repeated iteratively until both differences are greater than T.
11. The steps from 4 to 11 are applied to all pseudo-randomly selected blocks until all bits of the watermark string are embedded.
12. The image in **YUV** domain is converted back to the **RGB** domain.

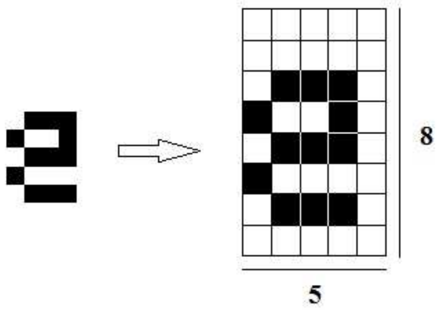


Fig. 6. Example of conversion from the character 'e' to the matrix of bits composed by 5x8 (40 bits). The white cells are represented by the value 0 and the black cells are represented by the value 1.

Figure 7(a) and Figure 7(b) show respectively the original “Lena” image and the “Lena” image affected by a digital invisible watermark of 200 bits (the string “SeCAM”) with the previously described watermarking algorithm.

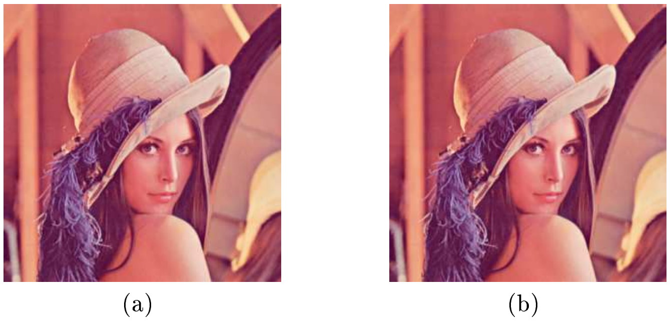

(a) (b)

Fig. 7. (a) The original “Lena” image; (b) The “Lena” image affected by a digital invisible watermark.

Analogously, the algorithm for extracting the digital invisible watermark is reported below:

1. The image is converted from the **RGB** domain to the **YUV** domain.
2. A block B of 8x8 pixels is pseudo-randomly selected from the image to read one bit of the watermark string.
3. A fixed binary pseudo-random pattern of the same size of B is generated.
4. I_0, I_1 and D are calculated from B. I_0 and I_1 are then obtained by calculating the averages of the luminance values in B, respectively where the random sequence is 0 and where the random sequence is 1. D is the difference $I_1 - I_0$.
5. If $D > 0$ then the embedded bit has value 1 else the embedded bit has value 0.

4 The End User Interface

From the end user point of view, there are three main system components:

- The *repository* GUI (described in Section 4.1)
- The *node* GUI (described in Section 4.2)
- The *portable device* GUI (described in Section 4.3)

4.1 The *repository* GUI

The *repository* GUI is very intuitive and simple, it is subdivided in two parts: the first one provides a panel to configure its starting options (Username and Password), the second one provides a panel which allows the *repository* to operate as a *node*. An example is shown in Figure 8.

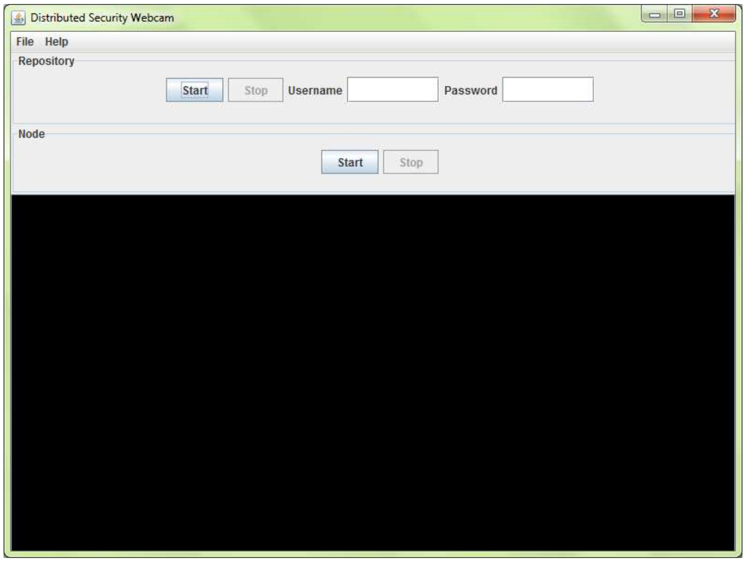
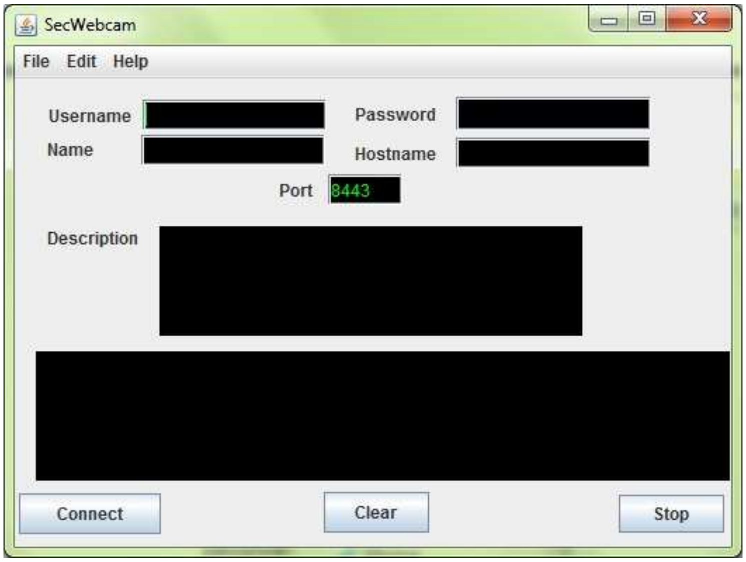


Fig. 8. The *repository* GUI.

4.2 The *node* GUI

The *node* GUI takes as input from the user: the Username and the Password, obtained during the registration process, the Name (or the alias) of the *node*, the Hostname (that is the IP address of the *repository*) and the Description of the *node*. Figure 9 shows an example of the *node* GUI.



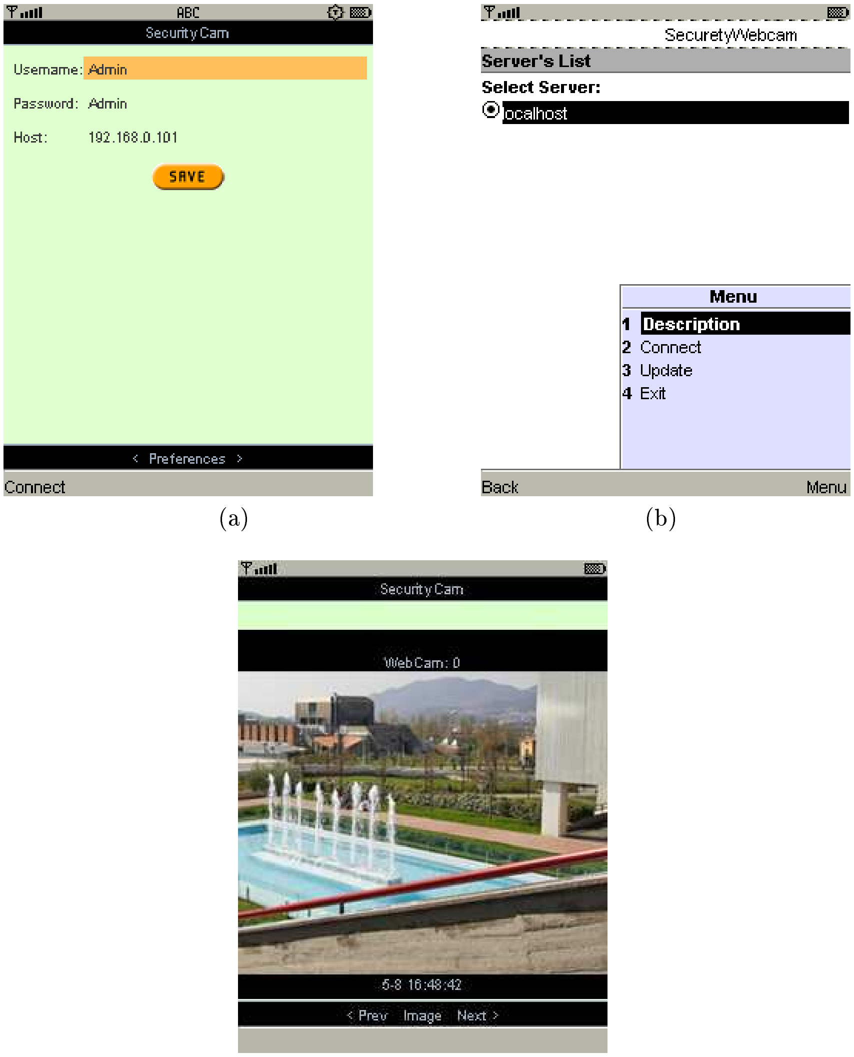
Port 8443

Fig. 9. The *node* GUI.

4.3 The *portable device* GUI

As in the case of the *repository* GUI, also the one of *portable device* is subdivided in two parts. In the first step (Figure 10(a)) the end user must insert the information associated to the connection with the *repository*. In details, the required information are: Username, Password and Host respectively used for authentication and connection.

In the second step (Figure 10(b)) the application shows the list of *nodes* obtained from the *repository*. For each *node*, the application has three commands: Details, Update and Connect. The Details command permits to obtain a detailed description of the *node*. The Update command permits to request the update of the local list of the *nodes*. The Connect command permits to connect with the *node* and to obtain the sequences of frames captured by the camera(s) (Figure 10(c)) of the *node*. Moreover, the Exit command permits to disconnect and close the application.



Usemame: Admin

Password: Admin

Host: 192.168.0.101

(a)

Select Server:

● ocalhost

(b)

WebCam: 0

5-8 16:48:42

(c)

Fig. 10. (a)shows an example of the first step of the *portable device* GUI; (b) shows an example of the *portable device* GUI with the list of the *nodes*; (c) shows an example of the image obtained by the selected camera of the *node*.

5 Conclusion and Future Works

In recent years, video surveillance has become an important system for the monitoring of areas, environments, etc. One of the more important application of the video surveillance is the security, but video surveillance is used also for other purposes such as traffic monitoring, etc.

In a real-world scenario, as for instance the homeland security and defense issues, where the video surveillance systems assumes a critical role, it is useful to receive real-time images from different camera(s), directly on a *portable device* as allowed by the presented system in this paper.

With the wide diffusion of *portable devices* such as tablets, smartphones and so on, it is now meaningful to have a really secure video surveillance system accessible from these devices, to reinforce or to extend the existent video surveillance systems.

The presented distributed video surveillance system, accessible from *portable devices*, is based on Client-Server architecture. A *portable device* that joins the system knows the address of the *repository* and obtains the complete list of the *nodes* which are connected to one or more camera(s). When the *portable device* obtains the list, it connects directly to a specific *node* and obtains the images from the camera(s).

Future work will consider the extension of the proposed architecture to a *peer-to-peer* architecture to allow scalability of the system with an increasing number of *nodes*.

Another research aspect could be the support for video-streaming from a *node* to the *portable devices*. Video streaming could allow advanced commands such as *Play*, *Pause*, *Stop* and *FrameCapture* respectively to indicate to *node* that can start the video streaming, that can temporarily stop the streaming, that can stop and deallocate the resources of the video streaming and to indicate to save a frame during the streaming, that, in second time, could send it to the *portable device*.

Moreover, it could be of interest to consider the sending from a *node* to a *portable device* a compressed short video with audio.

References

[1] Castiglione, A., Cepparulo, M., Santis, A.D., Palmieri, F.: Towards a lawfully secure and privacy preserving video surveillance system. In: Proceedings of the 11th International Conference on E-Commerce and Web Technologies (Lecture Notes in Business In-formation Processing). Volume 61. (2010) 73–84

[2] Castiglione, A., De Santis, A., Palmieri, F.: Ensuring privacy and confidentiality in digital video surveillance systems. In: Privacy Protection Measures and Technologies in Business Organizations: Aspects and Standards, Ed. George O.M. Yee, IGI Global (2012) 245–267

[3] Fleck, S., Straßer, W.: Towards secure and privacy sensitive surveillance. In: Proceedings of the Fourth ACM/IEEE International Conference on Distributed Smart Cameras. ICDSC '10, New York, NY, USA, ACM (2010) 126–132

[4] Hunker, J., Probst, C.W.: Insiders and insider threats - an overview of definitions and mitigation techniques. Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA) **2**(1) (3 2011) 4–27

[5] Liang, Q., Ding, Z.: Spread spectrum watermark for color image based on wavelet tree structure. In: Computer Science and Software Engineering, 2008 International Conference on. Volume 3. (dec. 2008) 692 –695

[6] Wang, Y.P., Chen, M.J., Cheng, P.Y.: Robust image watermark with wavelet transform and spread spectrum techniques. In: Signals, Systems and Computers, 2000. Conference Record of the Thirty-Fourth Asilomar Conference on. Volume 2. (29 2000-nov. 1 2000) 1846 –1850 vol.2

[7] Bender, W., Gruhl, D., Morimoto, N., Lu, A.: Techniques for data hiding. IBM Systems Journal **35**(3.4) (1996) 313 –336

[8] Kutter, M., Jordan, F.D., Bossen, F.: Digital watermarking of color images using amplitude modulation. Journal of Electronic Imaging **7** (April 1998) 326–332

[9] Pizzolante, R., Carpentieri, B.: Copyright protection for images on mobile devices. In: Proceedings of IMIS 2012. (2012)

[10] Langelaar, G.C., van der Lubbe, J., Biemond, J.: Copy protection for multimedia data based on labeling techniques. In: 17th Symposium on Information Theory in the Benelux. (1996)