Networking and Security Issues for Remote Gaming: The Approach of G@L

Christos Bouras, Vassilis Poulopoulos and Vassilis Tsogkas
Research Academic Computer Technology Institute,
N. Kazantzaki, Panepistimioupoli Patras, Greece
bouras@cti.gr, poulop@cti.gr, tsogkas@cti.gr
tel. +302610996951
fax. +302610996358

Abstract

As the evolution of computer technology introduces new advances in networks among others, online gaming becomes a new trend. Following the trends of our era, The Games At Large IST Project introduces an innovative platform for running interactive, rich content multimedia applications over a Wireless Local Area Network. The Games at Large project's vision is to provide a new system architecture for Interactive Multimedia that will enhance existing CE devices such as, Set Top Boxes (STB), Small Screen and other devices, which are lacking both the CPU power and the graphical performance to provide a rich user experience. In this study we present the controllers' sub-system of the innovative mechanism that is implemented within the context of the Games at Large project. We furthermore provide information on the encryption and security of the aforementioned communication channel.

Index Terms — *remote control channel, online gaming, remote command execution, input device capturing, asymmetric encryption, reverse channel*

1. Introduction

The future home is an always-on connected digital home. By the year 2010, there will be more than 420 million broadband households worldwide [8][16]. With the standard set for super-high speed, always-on connection, the way people view entertainment has fundamentally changed and new standards for consumption were created. Consumers no longer expect their Internet access to be only from a desktop PC - now they want it through the TV in their living room or in the palm of their hand, inside the house and on the go. The presented scenario [5] bundles video gaming capabilities into consumer electronics devices, such as Set-Top Boxes (STBs), Digital Video Recorders (DVRs), home entertainment systems, TVs, handhelds and other devices that are not considered, today, as real gaming devices since they lack the necessary CPU and GPU power. In this study we present a new system for pervasive gaming and multimedia, which is being developed under the EU FP6 project, Games At Large (G@L). This study is dedicated to the design testing concept elaboration, in order to base the approach for the development of evaluation and testing methodologies. The testing and verification process is part of the iterative, spiral-life workflow model (user-centered design and incremental improvement based on feedback from user and expert evaluation of prototypes).

The main idea of the project is that one or more powerful servers will actually execute the game on behalf of the client, which will be presented only with the screenshots of the game and not the game loader or the execution of complex graphics. On the other hand, the basic aspect of a game is the interaction with the end user (gamer). This means that apart from only presenting the game frames to the user (through a client – server architecture) the system must be able to capture any input from the input devices of the end user and transfer them to the server in order to emulate the interaction that is done on a physical level when playing a game. An important aspect of the aforementioned procedure is security when transferring the input commands from the client to the server. In particular, keyboard input, which in most cases depicts user sensitive data such as passwords or credit card numbers, must be foolproof.

In this study we present a mechanism for transferring input commands from any device, acting as the client, to execution commands at the corresponding program - game of the server. The purpose of this mechanism is to be able to control a program that runs on the centralized server from a remote operating

system. This mechanism is created within the scope of the Games at Large project. Meeting the demand of highly interactive multimedia systems with low cost end devices (CE), requires a radical change in the system's architecture. The Games At Large project intends to design a platform for running interactive rich content multimedia applications. Games At Large vision is to provide a novel system architecture for Interactive Multimedia which will enhance existing CE devices such as, Set Top Boxes (STB) and other devices which are lacking both the CPU power and the graphical performance, to provide a rich user gaming experience. We thus present the general architecture of the sub-system that controls the input of the client devices and their server side execution. More specifically, we examine how, input is able to be captured by any input device on the different end devices and on different operating systems, how commands are sent over the network and finally, how commands are executed at the target software of the server. Moreover, we present the general architecture of the encryption subsystem which ensures that the input from any keyboard devices connected to the client side is encrypted before being transmitted to the server side for execution. The purpose of this mechanism is to expand the capabilities of the command transferring channel. More specifically, we examine how capturing from any input device on different end devices and on different operating systems is done, how public key encryption is applied and how commands are decrypted and executed at the target software of the server. In our work, we are considering only the confidentiality issues of the cryptographic module assuming that authenticity should be provided by the general architecture of the system or by a different module.

The rest of the manuscript is structured as follows: the next section provides information about related work. Section 3 describes the vision and goal of the Games at Large project. Section 4 describes the general architecture of the system and the architecture on each device (the end device and the server). Section 5 describes the encrypted command channel infrastructure, while section 6 presents the encryption subsystem. In section 7 we present the general client-server infrastructure. The paper concludes with general remarks and future work that will be done within the scope of the project.

2. Related Work

Computer games constitute nowadays one of the most dynamic and fastest changing technological area, both in terms of market evolution and technology development. In this area, as the computer games are evolving and online activities and gaming become parts of our lives, the need for interaction within a client – server architecture becomes very intense. The successful paradigms of online gaming such as WoW [15], Half Life [7] and Second Life [11] are only just the beginning of a new era for the online games. The idea that lies behind online gaming is that a game that can be played by multiple users should not have only a local context. The basic game software is installed on the client machine, while multiple servers are assigned with the task of interconnecting all the possible users to what is called the "world" or the scenario of the game. The Games at Large project, as described in the official website, goes one step further than the classical procedure of online gaming and the main intention is to enhance the idea of application on demand [6], in order not only to support games on demand, but also to enable devices that lack the physical power to load a game, to run games **Error! Reference source not found.**[1][4][13].

The proposed architecture resembles that of thin-client computing [17], consisting of a server and a client that communicate over a network using a remote display protocol. Graphical displays are virtualized and served across a network to a client device by the protocol, while application logic is executed on the server. By using a remote display protocol, the client transmits user input to the server, and the server returns screen up dates of the user interface of the applications from the server to the client.

Some previous works on computing platforms include STARS [9], a unified platform that focuses on tabletop gaming, and [3], where the authors explore how computer games can be designed to regain some of the social aspects of traditional gameplay.

Many of these remote display protocols can effectively be web-enable applications without application modification. Some examples of thin-client platforms include Citrix MetaFrame [18], AT&T Virtual Network Computing (VNC) [19] and Tarantella [20]. The remote server typically runs a standard server operating system and is used for executing all application logic. Because all application processing is done on the server, the client only needs to be able to display and manipulate the user interface. The client can either be a specialized hardware device or simply

an application that runs on a low-end personal computer.

The objective of secure communications has been to provide privacy or secrecy, i.e., to hide the contents of a publicly exposed message from unauthorized recipients. The asymmetric encryption / decryption channel solves the major confidentiality issue of secure communications. Cryptosystems, as explained by the classic work of Simmons [12], are symmetric if either the same piece of information (key) is held in secret by both communicants, or else that each communicant holds one from a pair of related keys where either key is easily derivable from the other. These secret keys are used in the encryption process to introduce uncertainty (to the unauthorized receiver), which can be removed in the process of decryption by an authorized receiver using his copy of the key or the "inverse key." This means, of course, that if a key is compromised, further secure communications are impossible with that key. On the other hand, in asymmetric cryptographic schemes the transmitter and receiver hold different keys at least one of which it is computationally infeasible to derive from the other.

The work on public key cryptographic systems has been rather intense over the last 20 years. The main difficulty in developing secure systems based on public key cryptography is not the problem of choosing appropriately secure algorithms or implementing those algorithms [10]. Rather, it is the deployment and management of infrastructures to support the authenticity of cryptographic keys: there is a need to provide an assurance to the user about the relationship between a public key and the identity (or authority) of the holder of the corresponding private key. In a traditional Public Key Infrastructure (PKI), this assurance is delivered in the form of certificate, essentially a signature by a Certification Authority (CA) on a public key [2].

3. The Games at Large Project

Games at Large (Games@Large) being an Integrated Project (IP) intends to research, develop and implement a new architecture to provide users with a richer variety of entertainment experience in their entire houses, hotel rooms, cruise ships and Internet Cafés, incorporating unprecedented ubiquitous game-play. The project evolved from the home environment to other local Focus Areas (FA) regarding the benefits such FA may gain based on the unique technology approach of Games at Large. The Integrated Project includes activities of TV Multimedia and Gaming using Enhanced Media Extender, Local Processing and Storage Server(s), Handheld Devices and Local Wireless Network. Games at Large intends to enhance the existing Digital Living Network Alliance (DLNA) and the UPnP Forum standards by introducing the unique set of features required for running games over a local network, like all other media and content types (video, audio).

Market interest is now revolving around capitalizing on the rapid increase of always-on broadband connectivity. Broadband connection drives to a new, digital, “Future Home” as part of a communications revolution, which will affect every aspect of consumers’ lives, not the least of which is the change it brings in terms of options for enjoying entertainment. Taking into account that Movies and Music provided by outside sources were at home long before the Internet and Broadband, the challenge is to invent new content consumption patterns and new types of content and services.

Games offer a leisure time activity for every member of the household – from avid gamers to kids, as well as allowing whole families to play together. Games offer also leisure time activity for guests in hotels and visitors in Internet Cafes. Games at Large offers ubiquitous accessibility for all members of the household on all desired entertainment devices. The project focuses on new innovative ideas such as multiple-game execution on the Games Gateway and delivery of graphics-rendering meta-data over the home network via low latency, low bandwidth Pre-Rendering Protocol to achieve low-cost implementation of ubiquitous game play throughout the house, while taking advantage of existing hardware, and providing multiple members of the family with the ability to play simultaneously.

Games at Large intends to enable the Games to diversify from dedicated appliances and a single corner of the house, to any place at home such as, the TV in the living room, the handheld device or any other device with the relevant screen, controls and connectivity. The project will also provide the required infrastructure for running games on the hotel guest room TV or on small screens for people sitting in Internet Cafés, cruise ships, trains or airplanes.

The technological challenges of the the Games at Large project are:

- Distributed computing and storage
- Video/Image/Graphics delivery with very low latency through a wired/wireless home network
- Adaptation of PC screen-images to TV screen and handheld devices
- Integration of wireless users’ game control devices

- Translation of user ergonomics to different devices and form factors
- Research of new class of Media Extenders for games
- Enhancement of STBs to support video games
- Development of new methods for QoS linking Consumer prospective with system measurements
- Enhancement of relevant industry standards for time critical multimedia content while maximizing Users Experience
- Security aspects for the system's architecture as a whole and for each subsystem independently

The Games at Large project's mission is to develop a new method for ubiquitous video games through unique technology to transfer graphical data while reducing latency and ensuring QoS in a cost-effective manner. Main focus will be given on studying and supporting the use of video games within four different focus areas: User's home, Hotels, Internet Café, and Elderly Houses. A multi-layer approach will cut horizontally across the Games at Large focus areas, aiming to assess the conditions under which a Games at Large platform may frame within and improve the state of the art of each business domain, through performing the following, logically consecutive activities: collecting user requirements, researching and developing common Technologies, implementing and integrating those technologies within the required Servers and prototype CE Devices, running technology verification and Training and evaluating all results.

4. System Architecture

Figure 1 depicts the general system architecture. As it is obvious, the system consists of two different "levels". The first level includes all the possible servers that will be used for the system, while the second level includes the connection of the different end devices of the system. The server side constitutes of multiple different servers that are assigned with the task of serving the games and require a very quick and stable communication between them, which is guaranteed using a wired LAN network. The second level of the depicted architecture is the interconnection of any possible end device with the server in order to communicate and interact so as to load and play a game. Connected clients can utilize a variety of end devices, such as set top boxes, laptops, PDA's or IPTVs.

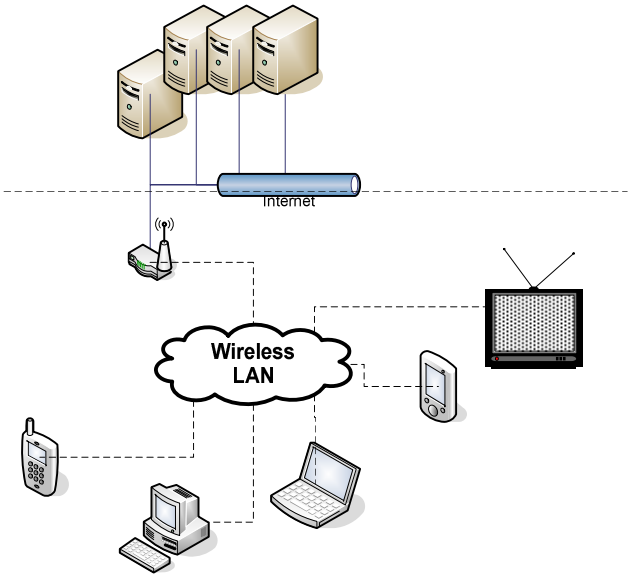

Figure. 1 General System Architecture

While on the architecture that is described all the servers can communicate to one another, the end devices can "see" only one server which is the main serving and processing server for the games. The server side of the system is assigned with various tasks, the most important of which is that of executing the game and sending the corresponding game screen to the connected clients. The Local Processing Server (LPS) coordinates the server infrastructure, and is responsible for the execution of the game graphics and the delivery of data between the clients and the server.

The clients are constantly sending feedback to the server which describes the input commands that are to be executed to the game instance. Thus, the server should be able to have at least two communication channels with each client: one for sending the game frames or 3D commands (direct channel), and one for receiving the input from the clients of the game (reverse channel). An important aspect to notice is that the channel which, if hijacked, could jeopardize the system's security, is the return channel since it contains not only the input commands that are for execution to the game instance, but also any other input from user. For instance, given the fact that the platform is targeted for commercial use, it is possible that the users will be required at some point to insert personal information, passwords, or even a credit card numbers. Hence, the encryption of the command communication channel and more specifically, the encryption of keyboard input commands is of major interest.

5. Command Channel Infrastructure

The idea that lies beneath the communication command channel architecture is depicted in the flow diagram of Figure 2. Each end device consists of many possible input devices that enable the user to interact with the device and thus enable the user to interact with the game that is played on the end device.

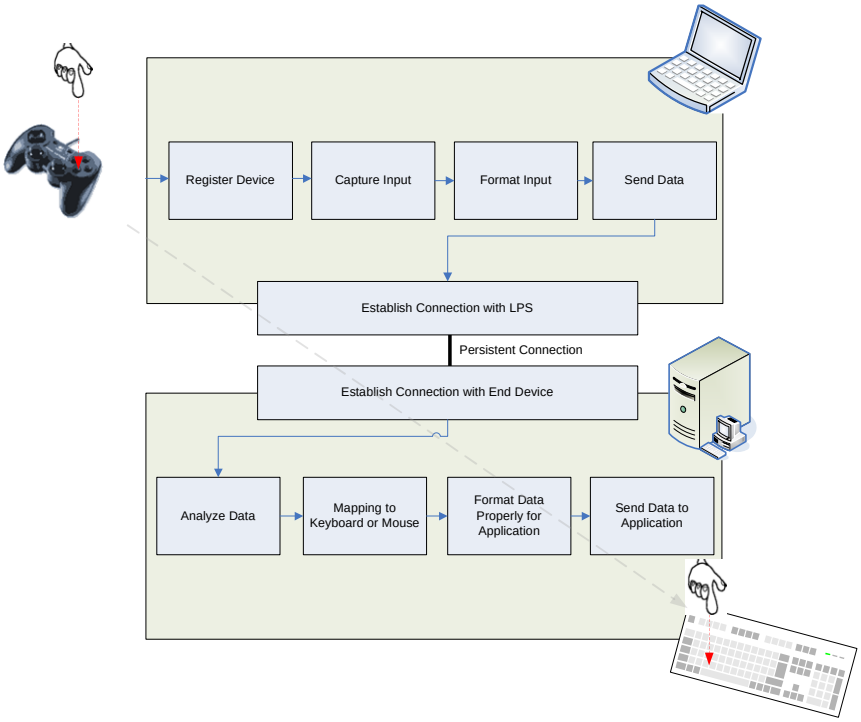

Figure. 2: Communication Channel Architecture.

When the client program starts, it initiates the device discovery procedure, which may be offered either by a separate architectural module, e.g. the device discovery module which uses UPnP, or by a system call causing the discovery for input devices attached to the system. It is essential afterwards, that the results of the device discovery are registered in our program so that we are aware of the existing input devices marking out several other non-existing.

The next step of the procedure is to capture the input coming from the input controllers. This is achieved by recording the key codes coming from the input devices. Input devices such as mice or keyboards are interrupt-driven while with joysticks or joy pads the polling method is used for reading. The previous means that whenever an input event is caused by a keyboard or a mouse, an interrupt message is sent to the message queue of our program; then it is translated and finally recorded. However, the polling case of joysticks or joy-pads means that these devices have to be polled by a program's thread in order to sense motion or button presses. The polling period has to be small enough to capture any input, but not too small to monopolize the system's CPU. A period of 10ms seems to be in our occasion a wise trade off.

After an input key code has been captured, the transmission of it takes place. This is achieved using an already open socket connection with the server side. Data is transmitted through the socket in the form of a string with a certain communication protocol. The socket connection can either be of TCP or UDP protocol. Since UDP emphasizes on real time, low latency transmission, it is preferable for this type of communication. Even if some key codes are lost in the process of transmitting them over the network, there is no real loss since there is a flow of key codes that can overcome this possible threat. Even though, in real life, error prune networks, such as WiFi's, the TCP protocol is selected avoiding the possible game experience fall caused by lost controller's packets transmission, for wired networks UDP is preferred.

Since the key codes have arrived at the server side, they are executed at the running game instance. At this point, there needs to be a distinction between the different types of transmitted key codes. There are basically four types of possible input device’s data

transmission. Commands may be coming from: (a) keyboard, (b) mouse, (c) joystick / joy-pad device or (d) any other HID input device.

In the first case, the server has to recognize the virtual key code, or the “pressed” / “released” event of a keyboard button, then do a possible mapping to some other key code, based on the game and user profile, and finally deliver it to the active application window for execution.

In the case of mouse input, the server has to recognize the virtual key code or the “pressed” / “released” event of a mouse button, recognize any mouse wheel event or any mouse movement (absolute or relative), then do a possible mapping to some other key code, based on the game and user profile, and finally deliver the key code to the active application window for execution.

In the case of joystick/joy-pad input, the server recognizes the state of the joystick/joy-pad device, maps the state to the appropriate keystrokes using the xml mapping file of the particular game-joystick/joy-pad combination. In this way, we are able to emulate the joystick/joy-pad input using pure keystrokes–mouse movements that represent the actual behavior of the input device. Finally the key code is delivered to the active application window for execution.

For any other HID input device the system treats input similarly to the joystick/joy-pad. The only prerequisite is the existence of a mapping file in order to convert the commands to keyboard and mouse instructions

6. Encryption

As already noted, the encryption procedure is only needed for the keyboard commands that the client transmits. We will now briefly describe the initialization procedure for supporting RSA public key encryption both at the client and the server of the Games At Large environment, as well as the thereafter communication between the LPS server and the connected client.

6.1 Startup phase

When both the client and the server start, some local initializations take place. Following, the client launches a connection request to the server which is advertised to the network neighborhood through the UPnP module. The server accepts the new client generating a unique RSA public-private key combination. Initially, through the persistent connection, the server transmits the modulus size in bits, the public exponent size in bits and the key pair size in bytes of the encrypted fields that follow.

The public key is described by an RSA structure and its fields are transmitted sequentially to the client though the possibly unsafe channel. The client then accepts the structure’s fields and re-generates the server’s public key. From this point on, any keyboard commands are encrypted by the client using the server’s public key and decrypted by the server providing thus the necessary security guarantee for the user-sensitive data. The aforementioned procedures are depicted in Figure 3.

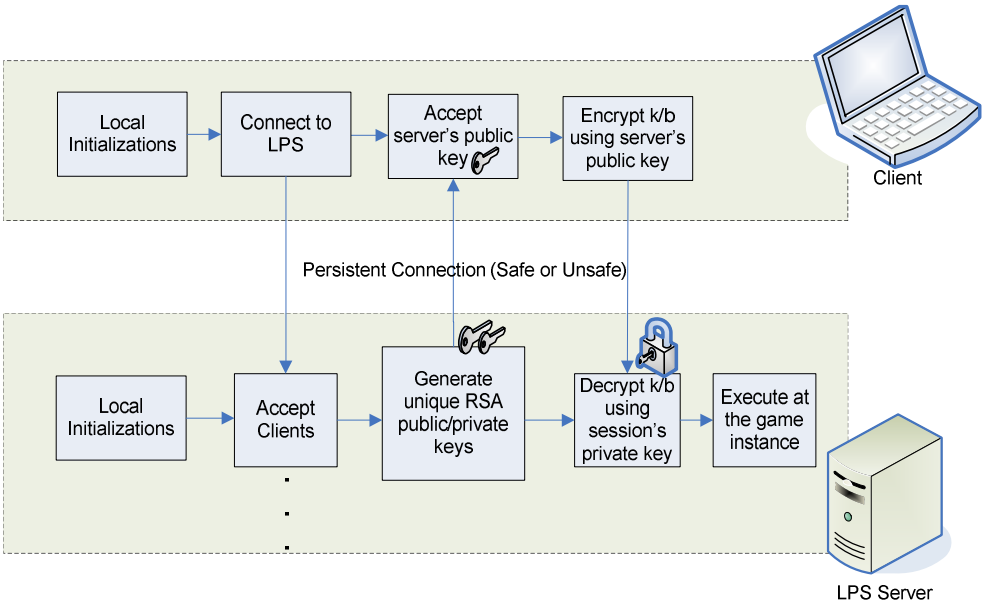

Figure 3 Initialization of the encryption module

6.2 Transfer of encrypted keyboard input

The idea that lies beneath the communication command channel architecture is depicted in Figure 4. Each end device consists of many possible input devices for interacting with the server. When the client program starts, it initiates the device discovery procedure, which may be offered either by a separate architectural module, e.g. the device discovery module which uses UPnP, or by a system call causing the discovery for input devices attached to the system. It is essential afterwards, that the results of the device discovery are registered in our program so that we are aware of the existing input devices marking out several other non-existing.

The next step of the procedure is to capture the input coming from the controllers. This is achieved by recording the key codes coming from the input devices. As already mentioned, input devices such as mice or keyboards are interrupt-driven while with joysticks or joy pads the polling method is used for reading. If the command that is to be transferred is originating from a keyboard device, the client uses the server's public key to encrypt the data after it has been suitably formatted adhering to a certain communication protocol. The encrypted message is transmitted to the server using an already open socket connection.

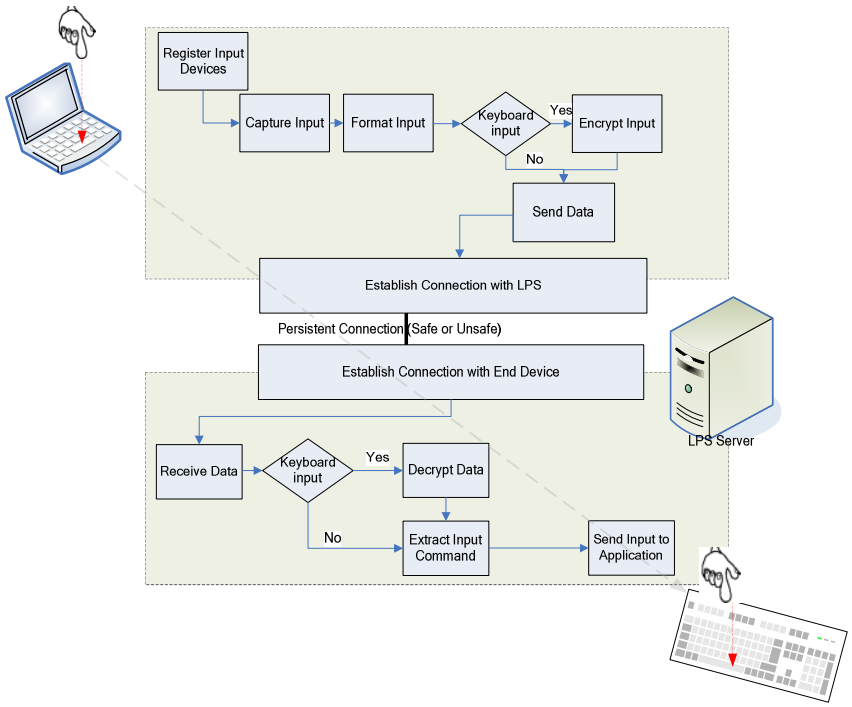

Figure 4 Encrypted Command Channel.

Once the encrypted message has arrived at the server side, the server decrypts it obtaining the initial keyboard commands that the client captured. If the received massage is not a keyboard one, the server bypasses the decryption stage, delivering the commands at the running game instance. The algorithm procedure of this step is presented in Algorithm 1.

```
//-- Client Encrypts and Send Keyboard Data
int encrypt(string message) {
//pk_size is the public key size
server.send_data(keyboard_type);
//notify the server for
//keyboard command that follows
unsigned char *encrypted;
int enc_size =
RSA_public_encrypt(strlen(message)+1,
(unsigned char *)message,
encrypted, PUBLIC_KEY, PADDING);
if (enc_size != pk_size)
{
Error("Ciphertext should match length of key");
return(-1);
}
//-- send encrypted data
return server.send_data((char
*)encrypted,enc_size);
}

//-- Server Receives and Dencrypts Keyboard
Data
int dencrypt(string encrypted) {
unsigned char *decrypted ;
char temp [MSG_SIZE];
//-- receive encrypted data
client.receive_data(temp,kp_size);
memcpy((char *)encrypted,
temp,kp_size*sizeof(char));
int decr_length = RSA_private_decrypt(kp_size,
encrypted, decrypted, PRIVATE_KEY,PADDING);
if(!decrypted){
Error("Encryption failed");
return -1;
}
Retrieve_vkey(decrypted);
}
```

Algorithm 1 Encryption and Decryption of keyboard messages

7. Server / Client Infrastructure

In this section, we describe the infrastructure that was implemented within the scope of the Games At Large project both at the server and the client side.

7.1 Server Side Infrastructure

As long as we are creating an environment with one server and multiple clients, it is essential to analyze how each end device will be able to capture all the commands from the input devices. This is because the unique server of the system should receive data that are sent over the network and execute the commands on the specific procedure that runs each game.

The "gateway" of the servers is the LPS (Local Processing Server). The main goal of Local Processing Server is to run multiple games simultaneously on the server, whereas each game runs in its own game environment and is streamed to an end-device. The game environment is an isolated and encapsulated "sandbox," providing the environment for game execution. The procedure, that makes the simultaneous running of multiple games possible, decouples the game execution from the game output, directed to display card/PC monitor, and all user-facing I/O, directed to the keyboard/mouse/HID. The LPS server also implements the encryption policy of the system by generating random RSA public/private key pairs for any newly connected clients and by decrypting the keyboard commands that come from the clients.

The "sandbox" environment for the server is created dynamically according to: a) the current occupancy of the resources and the hardware requirements that the game sets on the server, b) the software requirements on the client side and c) the current network condition. In order to be able to run a game on the server, the system monitors in a periodical manner the hardware resources of the server and the network conditions (jitter, latency and bandwidth). Additionally, according to the end device specifications (hardware and software), the server decides on the manner that the game will be executed on the client side.

7.2 Client Side Infrastructure

The possible different clients of the Games At Large environment are: (a) a Laptop with Windows XP / Vista environment, (b) a Set-Top Box with either Linux or Windows CE and (c) an enhanced handheld device with either Windows CE or a Linux version for small screen devices.

Each client implementation should consist of the following components: (a) a device discovery module, (b) the game browser and game launcher modules, (c) authentication modules, (d) input capturing and command transferring modules and finally, (e) decoders in order to run the streamed game that is sent from the server.

The device discovery module is used to seek for an appropriate LPS to connect to and introduce itself to the LPS with the End Device capabilities. The Games At Large Game Browser, which queries the Games Service on the LPS for listing the available games to the user, enables the user to browse the list of available games and select one to launch. Personalization of the UI should be available to the user/provider for enabling different views for users (i.e. Browser skins). When the user selects a game and requests to launch it, the Games At Large Game Browser issues a Start Game request to the Games At Large Client Game Launcher. The Game Browser will show to the user only the list of games that can run on the End Device by filtering the list according to the capabilities of the End Device compared with each game requirement.

Authentication communicates with the Games At Large Game Browser to authenticate the user against the LPS authentication module that authenticates the user against the Management Server. The Client Game Launcher controls all modules on the client side. The Game Launcher communicates with the LPS discovered by the device discovery module. The capture controller captures the Human Input Device (HID) controls and transfers them to the Controller Emulator on the LPS via the network layer, using the Controller protocol.

Each client should also implement the necessary RSA functions for the encryption module. For this cause, we are utilizing the OpenSSL RSA library which is available for the aforementioned platforms [14].

7.2.1 Capturing commands in Windows OS

As already mentioned, the end devices of the system can be multiple and thus they may use various operating systems, whereas the server is based on Windows operating system. When the client utilizes Windows operating system, the implementation of the

modules, and more specifically, the command capturing module, is implemented as a generic driver. This driver is able to recognize any input device and transform the command from them to keyboard and mouse commands, according to mapping files that are utilized for this scope. The aforementioned is a windows application, included in the game launcher of the client, which is called reverse channel module. The main assignments of the reverse channel module is a) to ensure that the connection to the server is established successfully, b) to capture commands from any input device, and c) to send the commands over the channel that is present between the client and the server.

7.2.2 Capturing commands in Linux OS

When the end device utilizes Linux operating system, there is no need for a low level driver (as in the Windows case) to be implemented as a client-side program in order to capture the input devices' input. On the contrary, the command capturing is feasible through the evdev Xorg input driver and the evbug capturing implementation, both of which are available to any modern Linux kernel. Finally, the keycodes from any input device are translated to keyboard and mouse commands and are then transmitted to the server for execution in the game instance.

8. System Evaluation

Extensive testing that is done by teams of the project on the issue of delay over the network has proved that in order to support games that require instant action (like racers or shooters) it is essential to have at most 50ms latency.

In order to be able to have latency of less than 50ms and as long as a large amount of them must be used for the transfer of the game graphics from the server to the client it is important that the return channel has as lowest latency as possible. The evaluation proves that an encrypted channel with dynamic encryption on each command sent from the client to the server requires 10 to 20ms regardless the game that is played. What we need to do is to lessen this latency in order to have the minimum overhead from the return channel.

To achieve lower latency we apply a different type of encryption on the channel. At the initialization phase we apply all the encryption as it is described in section 6 but not in order to send commands but in order to create a secure channel. As long as the secure channel is initiated we are assured that each byte on the channel is encrypted by the channel itself. This means that we are transferring the encryption of the data to a lower level of the ISO/OSI network layering system. From the application level of encryption we are moving to a network level by creating a single secure channel.

This approach has led to less latency on the return channel which is almost as fast as a pinging command from the client to the server. Nevertheless, we already know that we are sending a very small amount of data per second which is usually hundreds of bytes and in average even less. The latest testing that was done on the return channel (specific results cannot be presented due to the confidence terms of the project) prove that the latency of the return channel is at most 5ms.

Considering the scaling of the system, although it is out of the scope of this specific document we can present some preliminary results. By using a server which was in the state of the art during the year 2007 (intel core2 duo 2GHz, 2GB RAM), we have managed to play simultaneously more than 15 casual games from different clients. When talking about demanding games the testing that was done proved that we can run 4 demanding at the same time.

As the project is currently under its third year of running we are not able currently to present extensive results on the evaluation of the system.

9. Conclusions

In this study we have described the command execution channel as well as the encryption module of the Games at Large project, an IP project with the vision to research, develop and implement a new architecture to provide users with a richer variety of entertainment experience in their entire houses, hotel rooms, cruise ships and Internet Cafés, incorporating unprecedented ubiquitous game-play. We are researching and utilizing new technological techniques to transfer graphical data while reducing latency and ensuring QoS in a cost-effective manner. Main focus is given on studying and supporting the use of video games within four different focus areas: User's home, Hotels, Internet Café, and Elderly Houses. A multi-layer approach cuts horizontally across the Games at Large focus areas, aiming to assess the conditions under which a Games at Large platform may frame within and improve the state of the art of each business domain, through performing the following, logically consecutive activities: collecting user requirements, researching and developing common Technologies, implementing and integrating those technologies within the required Servers and prototype CE Devices,

running technology verification and Training and evaluating all results.

10. Future Work

As the system is implemented, more and more features are included on the release versions. These include modules that utilize network specific characteristics in order to adapt on the possible network environment (QoS support). These characteristics are expected to guarantee a minimum level of quality to any connected client, either utilizing a fast wired network or a noisy wireless one. Additionally, efforts are made towards the direction of creating software for every possible operating system in order to enable more end-devices to be connected to the Games at Large Environment. Furthermore, in our plans is also the incorporation of a media streaming server to the gaming infrastructure which will allow the connected clients to enjoy their preferred music or video clips with music/video on-demand characteristics.

REFERENCES

[1] C. Bouras, V. Poulopoulos, I. Sengounis, V. Tsogkas. "Networking Aspects for Gaming Systems", In Proceedings of the Third International Conference on Internet and Web Applications and Services pp. 650-655, 2008

[2] P.S.L.M. Barreto, H.Y. Kim, B. Lynn, and M. Scott. "Efficient algorithms for pairing-based cryptosystems," In Advances in Cryptology – CRYPTO 2002, volume 2442 of LNCS, pages 354–368. Springer-Verlag, 2002.

[3] S. Bjork, J. Falk, R. Hansson, P Ljungstrand, Pirates! Using the Physical World as a Game Board, Interact 2001

[4] C. Bouras, V. Poulopoulos, I. Sengounis, V. Tsogkas "Input here - Execute there through networks: the case of gaming". The 15th Workshop on Local and Metropolitan Area Networks (LANMAN 2007), Princeton, NJ, USA, 10 - 13 June 2007

[5] P. Casas, D. Guerra, I. Irigaray, User Perceived Quality of Service in Multimedia Networks: a Software Implementation, Joint Research Group of the Electrical Engineering and Mathematics and Statistics Departments, 2006

[6] Games at Large project's official website, http://www.gamesatlarge.eu

[7] Half Life official website, http://orange.half-life2.com/

[8] IPTV, By 2010, One-Third of the Predicted 422m Broadband Households will be Able to Receive IPTV. http://www.findarticles.com/p/articles/mi_m0EIN/is_2006_Sept_26/ai_n16837715

[9] C Magerkurth, R. Stenzel and T. Prante, STARS - a ubiquitous computing platform for computer augmented tabletop games. In Extended Abstract of UbiComp '03, Springer, 267—268 2003

[10] S. Sattam, Al-Riyami and K. G. Paterson, "Certificateless Public Key Cryptography," Lecture Notes in Computer Science, pp. 452 - 473, 2003

[11] Second Life official website, http://www.secondlife.com

[12] G. J. Simmons, "Symmetric and Asymmetric Encryption," in ACM Computing Surveys (CSUR), vol. 11, no. 4, ACM Press New York, NY, USA 1979, pp. 305-330.

[13] Y. Tzruya, A. Shani, F. Bellotti, A. Jurgelionis, Games@Large - a new platform for ubiquitous gaming, BroadBand Europe 2006, Geneva, Switzerland, November 2006

[14] J. Viega, M. Messier, and P. Chandra, 2002. Network Security with OpenSSL, 1st Ed. O'Reilly, Cambridge, MA.

[15] World of Warcraft official website, www.worldofwarcraft.com

[16] Worldwide online access. http://www.emarketer.com/Report.aspx?bband_world_jun06&src=report_summary_reportsell

[17] Albert Lai , Jason Nieh, Limits of wide-area thin-client computing, Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, June 15-19, 2002, Marina Del Rey, California

[18] Citrix MetaFrame 1.8 Backgrounder. Citrix White Paper, Citrix Systems, June 1998.

[19] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper. Virtual Network Computing. IEEE Internet Computing, 2(1), Jan./Feb. 1998.

[20] A. Shaw, K. R. Burgess, J. M. Pullan, and P. C. Cartwright. Method of Displaying an Application on a Variety of Client Devices in a Client/Server Network. US Patent US6104392, Aug. 2000.