

**SIXTH FRAMEWORK PROGRAMME
PRIORITY IST-2002-2.3.1.8
Networked Audiovisual Systems**

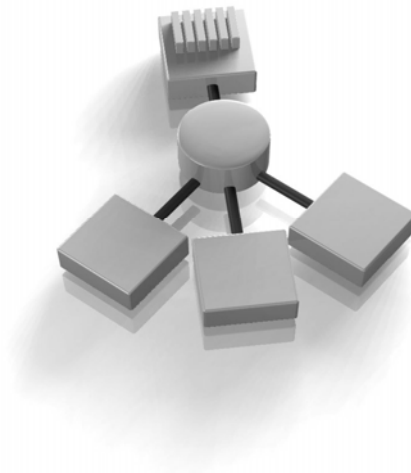


Uni-verse project

Deliverable D 3.2

Protocol security implementation

Feb 1, 2005



STREP project

Project acronym: Uni-Verse

Project full title: A Distributed Interactive Audio-Visual Virtual Reality System

Proposal/Contract no.: 002228

Distribution: Public

Content

Content 2

Summary 4

About the Author 4

Algorithms Used 4

The Login Security 4

The Data Security 4

User Management 5

RSA Key Length 5

Availability 5

WORKPACKAGE DESCRIPTION		
Workpackage Title: Protocol/API extension		WP No: 3
Starting date: month .3	Duration: 9 months	Total Effort in Man-months: 21
Member involved	Task description / Contribution of Member	Effort man-months
KTH	Workpackage leader, Participant	12
II	Participant	1
HUT	Participant	1
FHG/IGD	Participant	1
MINUSPLUS	Participant	0
PAREGOS	Participant	0
BLENDER	Participant	6

Objectives

The current version of the real-time protocol developed by partner II [17] is event-based for graphics and animation. This WP will extend the protocol enabling high quality sound and acoustics modelling, animation and skeleton representation. To facilitate the integration of audio and acoustic simulation in the Uni-Vers system the protocol will be extended to enable the description of virtual sound sources, their behaviour and the acoustic properties of the modelled environment that interacts with the sound sources. Added to existing support for security encrypted login will be implemented. Evaluation and testing of protocol extension will be undertaken according to current software design practices. Delivered as Open Source. In this work package will also deliver the API for the scripting environment. Since the scripting environment is plug-in based this is important for other workpackages that will use it by writing plug-ins.

Description of work / tasks

The work will comprise the following tasks:

WP 3.1 Protocol improvement

Initially a transform and an animation system for efficient low bandwidth communication of 3D motion data will be implemented. Adding audio support.

WP 3.2 Security

Implementation of an authentication mechanism.

WP 3.3 Scripting environment API

Implementation of the Basic API for writing processing plug-ins for the scripting environment

WP3.4 Testing

Testing of the protocol extensions and their implementation. Corrections, bug fixes etc.

Deliverables

The deliverables of WP 3 are:

D 3.1. Implementation of the improved protocol (Month 5)

D 3.2 Security implementation (Month 8)

D 3.3 Implementation of the scripting API (Month 8)

D 3.4 Tested implementation of protocol (Month 12)

Milestone

M 3.1 Tested implementation of protocol; Works according to D 2.2 (Month 12)

Interrelation with other workpackages

Basic task for all subsequent workpackages. Provides input to WP4 - WP7.

Deliverable D3.2 Protocol Security Implementation

Summary

This document, D3.2, is the description of the security features implemented in the Verse protocol in Uni-Verse project, A Distributed Interactive Audio-Visual Virtual Reality System, proposal/Contract no.: 002228. The work has focused around extending and upgrading the Verse protocol with network security. Such security serves three goals; it gives users a way to make sure the server they are communicating with is who it claims to be; it gives the ability to transmit user name and password without risk of third parties learning this sensitive information; and finally it protects the subsequent data traffic from snooping

About the Author

This report was written by Emil Brink, who is employed by partner KTH as a developer/engineer. Emil has a Master of Science in Computer Science and Engineering from KTH, and also worked with Eskil Steenberg on the original Verse implementation at the Interactive Institute in 1999 and onwards.

Algorithms Used

Verse now uses two distinct encryption systems, each used in one of the two non-overlapping "phases" of communication that exist for each session. The first phase, the login, uses RSA public-key encryption. It is considered fairly secure, and the public/private key pair allows a design where server and client can exchange data privately and securely, and where the privacy is not a function of the user's password. The second phase, used for all subsequent actual data traffic, uses simple XOR encryption, which is very fast. Below are further details on how these two systems complement each other.

The Login Security

Logging in, or "connecting", to a Verse host is the first step a client application needs to take in order to start communicating and sharing data. The user typically provides the application with a name and password combination to use, and these need to be sent to the server in a secure manner.

The host has an RSA key pair which is persistent; it remains the same even if the server is restarted¹. Clients generate a new key pair each time they are started. The symmetry of RSA keys means that the host can publish, i.e. Transmit, its public key in the clear to the client, and the client can then use that key to encrypt its name and password. Only the host, with its private key, can decrypt this data. The client also sends its public key to the server, which can then use it to encrypt the key for the data security, see below.

The Data Security

Verse's data security is simple binary XOR of each byte in an outbound packet with one byte from a circular random key. This key is constructed by the host and sent to the client during login, encrypted with RSA. Each client-host session has its own data key. The length of the data key is currently the same as the length of the RSA key. This is not a severe limitation, but might be lifted in the future since having a longer data key is computationally cheap and also increases security. The reason they are the same is that it fit better with the RSA infrastructure.

XOR encryption is very fast on typical microprocessors, which is why it is well suited for the bulk nature of Verse's data traffic. It is implemented at a low level by simply using the packet

¹ This is implemented simply by storing the key on the server's local hard disk.

number as an initial index into the random key, and then XOR:ing sequential bytes together, wrapping the key as necessary. This operation should be much faster than the outbound network link.

User Management

Missing from the current implementation is a user database, which could be used to attach a set of rights to a client, or simply to prevent unknown clients from connecting at all.

RSA Key Length

In the current implementation, an RSA key length of 512 bits is being used. This is less than is desirable, and perhaps even less than is needed to provide even basic security. The main limiting factor at the moment is simply performance; implementing RSA encryption requires the use of big integers since all bits need to be used in arithmetic. Most computers today can only calculate efficiently with numbers of 32 or perhaps 64 bits in length. Extending the precision requires implementing the arithmetic in software, and introduces various overheads. Efficient algorithms exist, but take a lot of time to find, implement, test and optimize, and although quite a chunk of time was spent on the RSA code, more is needed to reach better performance.

Using off-the-shelf code for this goal is harder than it may sound. Although the RSA system is very well known and lots of example implementations exist, many ignore performance for clarity. Of those optimized for performance, it is hard to find one that is compatible with the Verse license requirements, and also some of these are very large (of the same size as the entire Verse code base, or larger) and extracting the needed parts becomes a big project in itself.

Since Verse has existed for many years without any type of network security and without problems, we feel that we now at least have a basic platform to stand on and that work on further extending the security by increasing the key lengths can wait a while.

Availability

The changes described above are integrated into the Verse core codebase, available in public CVS: <<http://projects.blender.org/viewcvs/viewcvs.cgi/verse/?cvsroot=verse>>. At the time of writing, it is not yet released and merged into the head branch, but that is likely to happen within the next few weeks.