# IMPLEMENTING AUTHENTICATION, REMOTE ACCESS AND LOGGING FOR AN OBSERVATORY CONTROL SYSTEM USING THE QT FRAMEWORK

Bhavya Kaushik
Dept. of Computer Science & Engineering
Manipal University Jaipur

***Abstract: Observatory Control System (OCS) of the 3.6 m Telescope, which is Asia's largest optical Telescope, has been created by using the Qt Framework. The OCS requires embedding certain characteristics, including authentication, remote access and logging. The control system was provided with a set of authentication services by maintaining login details. All these details were kept private by using the SHA-1 algorithm. Since the software needs to be accessed by all the collaborated observatories across the world, a mechanism to implement the remote access facility of the OCS was successfully achieved by creating a separate thread in the application which keeps on listening to every incoming connection and reports the IP of that connection to the OCS respectively. As the system would be accessed by multiple users, the maintenance of separate logs which records every activity of the operator is essential. The same was implemented by including the QLogger class of the Qt framework. The paper elucidates on the creation of the OCS and how these features were included successfully.***
***Keywords: Observatory Control System, Remote Access, Authemtication through SHA-1***

## I. INTRODUCTION

Aryabhatta Research Institute of Observational Sciences (ARIES) is one of the leading research institutes of India that specializes in Astronomy, Astrophysics and Atmospheric Sciences. Their most anticipated project so far is the 3.6m Optical telescope located at their Devasthal Campus, India. The 3.6m telescope, which would be the largest optical telescope of Asia, requires a software control system in order to be accessed and operated successfully. An Observatory Control System (OCS) is a software system which provides an interface to operate different sub-systems involved in it. It runs on its own dedicated specialized computer that is associated with the Telescope hardware. The software is able to control both the operational and engineering controls of the telescope hardware through a graphical screen and a command line user interface [1]. An OCS includes three major sub-systems, which are Telescope Control System (TCS), Instrument Control System (ICS) and a Dome Control System (DCS). Some other functional systems like web access, weather server, log system, etc. are also attached to the OCS. As Telescopes are getting complicated with time, manual control is impossible. An OCS has become an essential part of any modern telescope [2]. Fig. 1 depicts the central hierarchy of the 3.6 m OCS system.

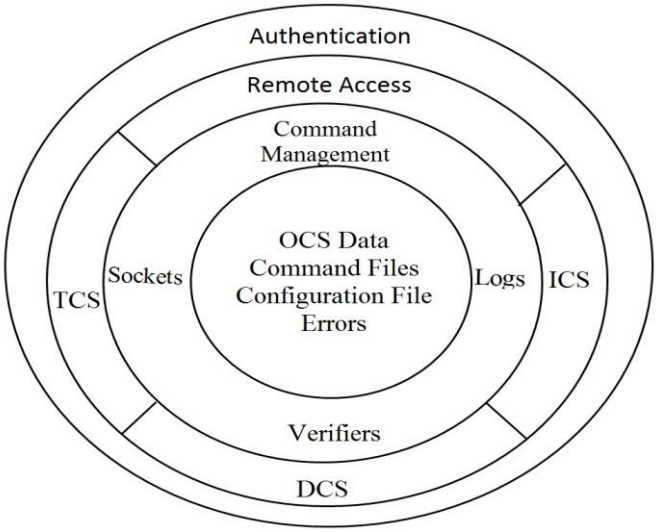


Fig. 1 Hierarchy of the 3.6m OCS

The TCS is the software system that forms the interface between the telescope hardware and the user. It runs on its own dedicated PC. The OCS forms a graphical interface which connects the user to the Telescope control system that sends various commands to the telescope hardware in a parallel fashion. An alternative but restricted programmatic interface has been provided through socket connections. ICS is another software component which deals with the instrument that is used to capture data during the observation. Similarly, a DCS is another stand-alone application that is used to control various activities of the Dome. Qt framework has been selected for designing the OCS, which is an open source, cross-platform framework and is widely used to create console applications [3]. The connectivity of an OCS with the TCS/DCS Server was implemented by using socket programming after inheriting the socket classes from Qt Network Module [4]. Since the software would be used by a number of operators (including various scientists and research scholars), a mechanism for authentication needs to be implemented in order to embed security in the application. For the telescope functionalities to be available across other observatories as well, which are located throughout the world - a facility of remote access should also be implemented. Furthermore, a logging system should be managed to view the records of various operations. The paper elucidates on how these services were successfully executed on the OCS.

  

## II. AUTHENTICATION

The process of password authentication, which is both a simple and effective method to provide security in any application, has been inculcated in the OCS. Every operator is provided with a unique username and a respective password. Furthermore, they can be distinguished with their project codes. All this information is stored in a secured file and would be matched at the run-time to provide access to the OCS. In a traditional password authentication scheme, a user is required to enter a unique code followed by the password, which are compared with the stored values at the server end. Since this scheme is vulnerable to many attacks, an additional encryption layer powered by the SHA-1 algorithm has been implemented in the OCS [5]. The scheme can be elucidated through Fig. 2

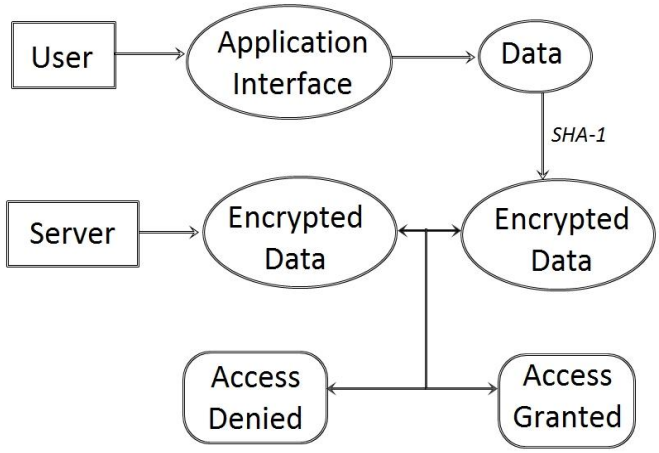


Fig. 2 Authentication scheme of the OCS

SHA-1 is used as an encryption scheme to provide data security. Originated by the National Security Agency, it uses a hash function to produce a 160 bit hash value, which typically is 40 digits long and comprises of a hexadecimal value [6]. It is one of the most widely used hashing algorithm which is used to provide data security for several applications and protocols.

The steps for encryption using an SHA-1 algorithm are as follows [7]:

Step 1: If the length of the original message L is less than (264 – 1) bits, add additional K bits, such that (L + K) MOD 512 = 448. The principle involves appending a '1' at the end of the message followed by trailing '0' until the length satisfies the requirement.

Step 2: Append the length of the original data with a trail of 64 unsigned length binary bits. After both these steps, the message will look like Fig. 3

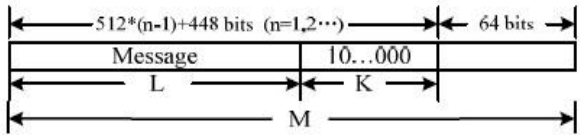


Fig. 3 An instance of filled message

Step 3: The message is parsed into n-512 blocks (Mo, M1... Mn-1), n = M/512

Step 4: Initialize the following five 32-bit words as:

H0 = 0x67452301;
H1 = 0xEFCDAB89;
H2 = 0x98BADCFE;
H3 = 0x10325476;
H4 = 0xC3D2E1F0;

Step 5: Each message block Mi (0≤i≤n-1) is processed as:

1. Parsing Mi into 16 32-bit words W0, W1,..., W15.
2. When 16≤t≤79, Wt is defined as follows:

Wt = S1 (Wt−3 ⊕ Wt−8 ⊕ Wt−14 ⊕ Wt−16)

, where S1 is circular left shift by 1 bit

3. Initialize the five 32-bit working variables, A, B, C, D and E with H0-H4 as:

A = H0 ； B = H1； C = H2； D = H3； E = H4；

4.

```
For (t=0; t<80; t++)
{
Temp = S5(A) + ft(B,C,D) + E + Wt + Kt;
E = D;
D = C;
C = S30(B);
B = A;
A = Temp;
}
```

Where Sn(A) is circular left shift of 32-bit A by n bits.

SHA-1 defines the logical functions f0(B,C,D) - f79(B,C,D), and 80 constant 32-bit words,K0-K79 as:

```
if 0 ≤ t ≤ 19 then
        ft = (B and C) or ((not B) and D)
        Kt = 0x5A827999
  else if 20 ≤ t ≤ 39
        ft = B xor C xor D
        Kt = 0x6ED9EBA1
  else if 40 ≤ t ≤ 59
        ft = (B and C) or (B and D) or (C and D)
        Kt = 0x8F1BBCDC
  else if 60 ≤ t ≤ 79
        ft = B xor C xor D
        Kt = 0xCA62C1D6
```

The operation of a single cycle can be observed from the demonstration in Fig. 4

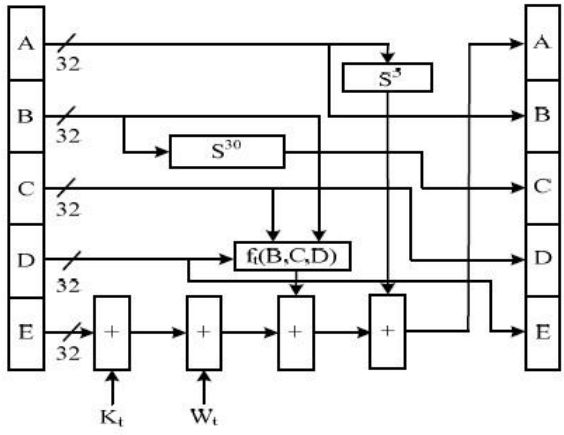


Fig. 4 The operation of a sing-cyclic iteration of SHA-1

  

5. Compute the intermediate hash-values as:
$H0 = H0 + A$;
$H1 = H1 + B$ ;
$H2 = H2 + C$ ;
$H3 = H3 + D$
$H4 = H4 + E$ ;

The final 160 bit length hash value would be derived as:
H = (H0 $<<<_{128}$) or (H1 $<<<_{96}$) or (H2 $<<<_{64}$) or H3 $<<<_{32}$) or H4;

It works in the similar fashion for the OCS application by converting an instance variable to its hash value. For example, the word "admin" would be stored as "d033e22ae348aeb5660fc2140aec35850c4da997".

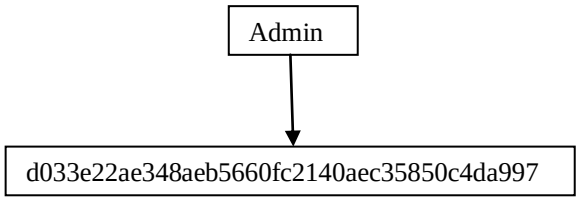


Fig. 5 provides the screenshot of the interface, as it would ask the username, project code and password from the user. These would be stored on the server and every time an operator is required to pass through this authentication layer in order to access the OCS interface.

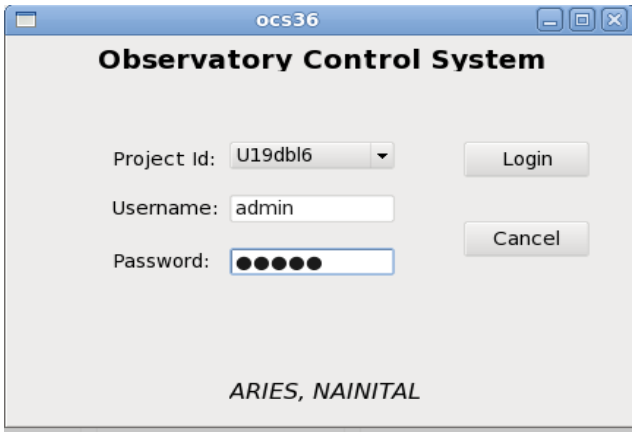


Fig.5 Screenshot of the OCS landing screen

## III. REMOTE ACCESS

With the widespread development of fast and cost-efficient internet services, it is easy to access any application using a remote access mechanism [8]. Thus, a model for accessing the interface of the OCS remotely from any other location was also proposed. Since a facility must be provided for any remote user to acquire the interface functionalities for a specific period of time, a separate thread in the interface was created to empower the same. The remote access thread would be activated as soon as the interface would be build by the Qt Framework. This thread will continuously listen for any remote connection. As soon as any other system would try to communicate with the OCS application, the thread displays a pop-up by reading the IP and other essential details of the incoming connection. An operator, after analyzing the details of the incoming connection can either accept or reject it. After accepting the connection, the remote user can control the application like any other operator. Fig. 5 denotes the entire remote access process in a nutshell.

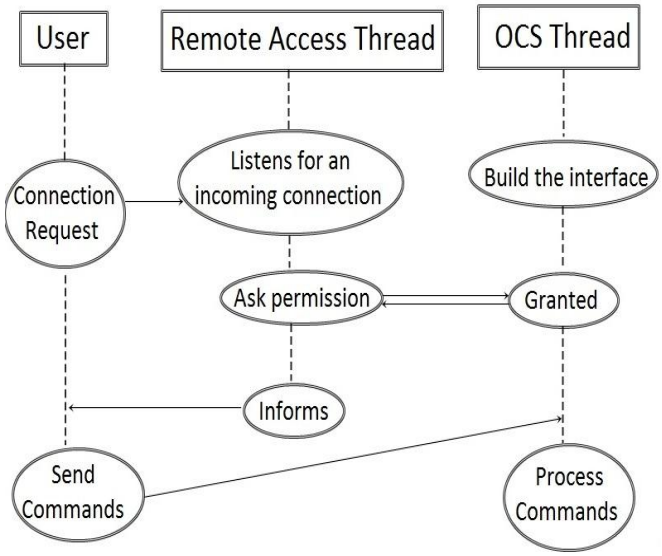


Fig. 6 Remote Access for the OCS

A pop-up screen as shown in Fig. 7 could be received at any time from an incoming connection.


Fig. 7 Remote Access request

## IV. LOGGING

Logging is a general purpose recording of any task which takes place in an environment or application. It helps the developers to continuously observe the progress of the system, as well as to spot any malicious attack on the server [9]. Since the OCS would be accessed by multiple parties, it is advisable to implement a logging functionality in order to record every command that is passed by a user and the dispatched response from the OCS. A secure log storage service as well as a log migration mechanism has been implemented which not only stores every detail, but also enables the operator to read and filter these records [10]. The logs were generated after inheriting the QLogger class of the Qt framework which provides a set of methods to generate logs with respect to every date at the server end. Furthermore, these records can be read by an operator after being filtered, in a separate window. The logs were generated in four types – information logs, error logs, warning logs and exception logs. The log viewer window can be seen in Fig. 8

  

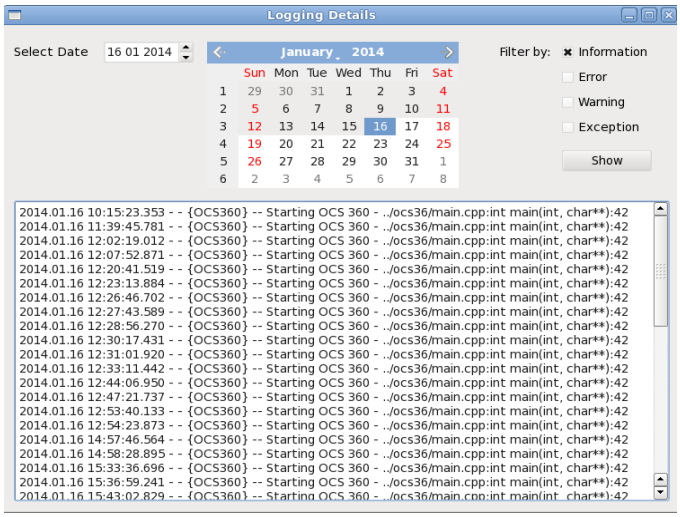


Fig. 8 Log Viewer

## V. CONCLUSION

The creation of a centralized Observatory Control System has been implemented successfully by Qt Framework. The socket programming was achieved through inheriting the network classes of the framework. The essential services like authentication, remote access and logging were embedded in the OCS. It stored a set of login details for all of its users and provided a basic authentication interface by implementing the SHA-1 algorithm. A separate thread was created which always listens for an incoming connection and reports about the details of the connection to the operator. The mechanism of remote access was implemented through this active thread. As the control system would be operated by multiple users, maintenance of logging details were inculcated by inheriting the QLogger class of the framework. Different logs for every respective date were stored and an interface was provided to read and filter these logs.

## REFERENCES

[1] Dong Jian, Wang Jian, Jin Ge, Deng Xiao-chao, Yuan Hai-long, "Research of application layer for large astronomical telescope Observatory Control System framework", International Conference on Computational Intelligence and Software Engineering, pp. 1-4, 2009.

[2] Wang Jian, Jin Ge, and Yu Xiao-qi, "The Design of Observatory Control System of LAMOST," Plasma Science & Technology, vol. 8, no. 3, pp. 347–351, May 2006.

[3] Summerfield, Mark, "Advanced Qt Programming: Creating Great Software with C++ and Qt 4" (1st ed.), Addison-Wesley, August 2010.

[4] Dalheimer, Matthias,"Programming with Qt" (2nd ed.), O'Reilly Media, January 2002.

[5] Xuguang Ren, Xin-Wen Wu, "A Novel Dynamic User Authentication Scheme", International Symposium on Communications and Information Technologies, pp. 713-717, 2012.

[6] Christophe De Cannière, Christian Rechberger, "Finding SHA-1 Characteristics: General Results and Applications", 2006.

[7] Xia Hong, Ning Hui-ming, Yan Jiang-yu, "The Realization and Optimization of Secure Hash Algorithm (SHA-1)", International Conference on Computer Science and Software Engineering, pp. 853-858, 2008.

[8] Nobuyuki Enoto, Hideo Yoshimi, "A secure and easy remote access technology", Asia-Pacific Symposium on Information and Telecommunication Technologies, pp. 364-368, 2005.

[9] Tomono, A. ; Uehara, M. ; Shimada, Y. , "Improvement and Evaluation of a Method to Manage Multiple Types of Logs", International Conference on Advanced Information Networking and Applications, pp. 601 – 606, 2011

[10] Jun Ho Huh, Andrew Martin, "Trusted Logging for Grid Computing", Asia-Pacific Trusted Infrastructure Technologies Conference, pp. 30-42, 2008

  