



2014-12-01

# Framework to Implement Authentication, Authorization and Secure Communications in a Multiuser Collaborative CAx Environment

Francis Mensah

*Brigham Young University - Provo*

Follow this and additional works at: <https://scholarsarchive.byu.edu/etd>



Part of the [Computer Sciences Commons](#)

---

## BYU ScholarsArchive Citation

Mensah, Francis, "Framework to Implement Authentication, Authorization and Secure Communications in a Multiuser Collaborative CAx Environment" (2014). *All Theses and Dissertations*. 4314.

<https://scholarsarchive.byu.edu/etd/4314>

This Thesis is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in All Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact [scholarsarchive@byu.edu](mailto:scholarsarchive@byu.edu), [ellen\\_amatangelo@byu.edu](mailto:ellen_amatangelo@byu.edu).

Framework to Implement Authentication, Authorization and  
Secure Communications in a Multiuser  
Collaborative CAx Environment

Francis Noble Mensah

A thesis submitted to the faculty of  
Brigham Young University  
in partial fulfillment of the requirements for the degree of  
Master of Science

Chia-Chi Teng, Chair  
C. Richard G. Helps  
Joseph J. Ekstrom

School of Technology  
Brigham Young University  
December 2014

Copyright © 2014 Francis Noble Mensah

All Rights Reserved

## ABSTRACT

### Framework to Implement Authentication, Authorization and Secure Communications in a Multiuser Collaborative CAx Environment

Francis Noble Mensah  
School of Technology, BYU  
Master of Science

Computer Aided Design (CAD) applications have historically been based on a single user per application architecture. Although this architecture is still popular to date, it does have several drawbacks. First of all the single user CAD architecture inhibits a concurrent engineering design process where several designers can work on the same model simultaneously. This limitation introduces time inefficiency especially when a project involves geographically dispersed designers. A solution to these drawbacks could be a transition from the traditional single user CAD architecture to a multiuser collaborative architecture.

Advances in computer networking technologies, especially relating to the Internet, have provided the needed tools to make this transition a reality, thus making it possible for designers to simultaneously work on geometric models from one or more networked computers regardless of the location of the user. This new paradigm is expected to improve collaboration and greatly reduce product design times and consequently reduce cost and improve productivity. The multi-user architecture will, however, also require reliable security mechanisms to ensure its successful deployment in an enterprise environment where protection of intellectual property is of critical importance.

This thesis proposes a framework to implement authentication, authorization and secure data communications in a multiuser collaborative CAD software system. This framework has been tested on an emerging multiuser collaborative CAD system called v-CAx being developed at Brigham Young University.

Keywords: CAx, CAD, TLS, PKI, multiuser, authentication, authorization, access control, secure communications, active directory

## ACKNOWLEDGEMENTS

I would like to express my appreciation to all who supported me throughout the course of this thesis. First, I would like to thank my committee members, Dr. Chia-chi Teng, Dr. Richard Helps, and Dr. Joseph Jones Ekstrom for their guidance and encouragement to work hard to complete this research work. I would also like to thank my parents and my siblings for supporting me in coming this far in my education. Finally, my thanks also goes to all my friends who assisted in one way or the other towards to success of this research.

## TABLE OF CONTENTS

<b>LIST OF TABLES .....</b>	<b>viii</b>
<b>LIST OF FIGURES .....</b>	<b>ix</b>
<b>1 INTRODUCTION.....</b>	<b>1</b>
1.1 Problem Statement.....	3
1.2 Thesis Statement.....	4
1.3 Assumptions.....	4
1.4 Delimitations.....	4
<b>2 REVIEW OF LITERATURE .....</b>	<b>6</b>
2.1 Current State of Collaborative Design in the Engineering Industry.....	6
2.2 Previous Work on Multi-User Collaborative CAD Systems .....	8
2.2.1 Multiuser Architectures .....	8
2.2.2 NX-Connect .....	9
2.3 Multiuser CAD Security Requirements.....	10
2.3.1 Security Risks in a Multiuser Collaborative CAD.....	11
2.3.2 Previous Work on Multi-user Collaborative Security.....	12
2.3.3 Security Requirements and Options for Multi-user CAD.....	14
<b>3 METHODOLOGY .....</b>	<b>17</b>
3.1 NX Connect Architecture .....	18
3.1.1 NX Unigraphics (U-G) .....	18
3.1.2 NX-Connect .....	18
3.1.3 Information Storage Module (ISM).....	19
3.1.4 Data Capture Module (DCM).....	19
3.1.5 Data Sync Module (DSM) .....	20

3.1.6	NX Controller .....	20
3.2	Review of NX Connect Security State .....	20
3.3	NX Connect Security Framework Overview .....	21
3.3.1	Authentication and Authorization.....	22
3.3.2	Confidentiality .....	29
3.3.3	Certificate/Key Management (Public Key Infrastructure (PKI)).....	30
3.4	Security Framework Design .....	32
<b>4</b>	<b>IMPLEMENTATION OF FRAMEWORK.....</b>	<b>34</b>
4.1	Implementation of Authentication Framework.....	35
4.1.1	The Authentication Mechanism .....	35
4.1.2	Implementing Authentication .....	36
4.1.3	Access Control (Authorization) .....	39
4.2	Implementing Secure Communication Channel .....	46
4.2.1	Implementing TLS .....	46
4.3	Certificate/ Key Management.....	47
4.4	Testing and Results .....	49
4.4.1	Test Prototype .....	49
4.4.2	Results.....	51
<b>5</b>	<b>CONCLUSION AND FUTURE WORK .....</b>	<b>62</b>
5.1	Conclusion .....	62
5.2	Future Work .....	63
	<b>REFERENCES.....</b>	<b>65</b>
	<b>APPENDICES .....</b>	<b>68</b>
	<b>APPENDIX A .....</b>	<b>69</b>
	Implementation of Secure Communications (TLS) in code .....	69

<b>APPENDIX B</b> .....	<b>72</b>
Implementing Authentication in NX Connect with Active Directory .....	72
<b>APPENDIX C</b> .....	<b>75</b>
Implementing Access Control (Authorization) in NX Connect .....	75
<b>APPENDIX D</b> .....	<b>78</b>
Administering Security Objects in NX Connect.....	78

## LIST OF TABLES

Table 1 Authentication Tests and Results.....	54
---	----



## LIST OF FIGURES

Figure 1 Architecture of NX Connect.....	18
Figure 2 Original Architecture of NX Connect .....	21
Figure 3 Original Authentication Mechanism for NX Connect.....	24
Figure 4 Proposed Authentication Mechanism for NX Connect.....	25
Figure 5 Operation of NX Access Control Model.....	28
Figure 6 Proposed Architecture for NX Connect with Enhanced Security .....	33
Figure 7 Authentication Class Definition .....	37
Figure 8 NX Connect Authentication Mechanism.....	38
Figure 9 SendAuthenticationRequest Method Definition.....	38
Figure 10 HandleAuthentication Method Definition.....	39
Figure 11 Modified Parts Table .....	40
Figure 12 Permissions Table.....	41
Figure 13 Part-Subject Pairs Table .....	41
Figure 14 Sample Permissions Data .....	42
Figure 15 Sample Part-Subject Pair Data .....	42
Figure 16 Part-Subject-Permission Table .....	42
Figure 17 Modified Database Schema for Access Control.....	43
Figure 18 Security Class Definition.....	44
Figure 19 SendAuthorizatoinRequest Method Definition .....	45
Figure 20 Authenticate and Connect Method Definitions .....	47
Figure 21 Implementing TLS on the Server .....	47
Figure 22 Methods to Check Certificate Validity for Client and Server Certificates.....	48
Figure 23 Test Prototype Setup.....	49

Figure 24 Network Traffic When Using No Encryption .....	52
Figure 25 Network Traffic When Using TLS for Encryption .....	53
Figure 26 Security Labels for SimpleAssembly.prt.....	56
Figure 27 Security Labels and Roles for User-1 and User-2 .....	56
Figure 28 Role and Security Class Hierarchies .....	57
Figure 29 User Views for SimpleAssembly.prt.....	57
Figure 30 User Views After Granting User-2 View Access for PistonRod.prt.....	58
Figure 31 User-2 Not Allowed to Make Modification to PistonRod.prt .....	59
Figure 32 Security Configuration for Senior Engineer Role .....	60
Figure 33 Diagnostic Results for NX Connect with Security Framework .....	61
Figure 34 Diagnostic Results for NX Connect without Security Framework .....	61

## 1 INTRODUCTION

Computer Aided Design (CAD) applications have historically been designed and implemented based on single user use cases. With single user CAD architecture members of a design team become contributors in a carefully controlled serial process. This approach clearly does not promote a parallel design process where several designers can work on the same model simultaneously and in a collaborative fashion.

Although single user CAD architectures are still popular, the drawbacks mentioned in the previous paragraph, coupled with the increasing trend of globalization of the world economy, increased outsourcing of components, increased competition, and pressures to reduce product development time call for a new approach toward engineering design and development processes. Ian Ziskin, chief human resource and administrative officer for Northrop Grumman, stated in an interview that in the next decade about 50% of aerospace engineers will reach retirement age (Brandon 2008). This is approximately true for other engineering industries as well. The problem with this scenario is that it could affect the transfer of knowledge from experienced to less experienced engineers which in turn could affect productivity. This situation, thus, also calls for new learning technologies to aid in knowledge transfer to novice engineers without the challenge of geographic limitations (Red et al. 2013). In line with this trend research work has gone into seeking techniques that will allow the design of engineering parts through the use of collaborative tools.

In addition to ongoing research in collaborative computing, advances in computer networking technologies, especially the Internet, have also enhanced the requisite tools and standards that make development of collaborative multi-user application architectures a reality. Several applications including massive multiplayer online role playing games (MMORPG), Google Docs, Microsoft Office 365, among others currently make use of multiuser collaborative technologies. The same, however, cannot be said for CAD applications. While it is true that researchers have tested the multiuser collaborative CAD paradigm, the major CAD companies have not offered collaborative multiuser versions of their core software (Red et al. 2013).

Several researchers have developed prototypes of multiuser collaborative CAD architectures and have demonstrated that such architectures are feasible and desirable. Why then are these prototypes not being adopted in industry? Red et al. (2013) argue that when technologies such as multiuser collaborative CAD are not adopted by industry players, it can be attributed to several reasons including the fact that they view the architectural enhancements ill-defined, difficult to adopt and/or risky. In addition they may not have considered the competitive advantages in the light of required organizational and proprietary process changes. Removing these architectural and other limitations and demonstrating the benefits of multiuser collaborative CAD will make the industry more likely to adopt these techniques.

A National Science Foundation (NSF) site under the acronym v-CAx (“Nu = new” computer aided applications) was organized at Brigham Young University in 2010. The premise in organizing this site was to address the limitations identified in previous multiuser collaborative CAD research work and demonstrate its practical adoption in industry. Currently several prototypes have been developed and development of others is still in progress. These prototypes are making use of application programming interface (API) libraries from major CAD applications

including NX from Siemens, CATIA from Dassault Systems, CUBIT from Sandia Corporation, Autodesk Inventor from Autodesk, among others, to transform the single user architectures of these CAD applications to collaborative multiuser architectures.

## **1.1 Problem Statement**

The original research question of the v-CAX project was: Can modern single user CAD applications be converted to multiuser collaborative architectures and what productivity improvements might be expected? Research work done so far on the v-CAX project with major CAD applications mentioned above reveal that the answer to this question is a definite yes. Several architectural limitations with previous research have been addressed including multithreading of CAD APIs and GUIs, access to CAD event handlers and interrupts, among others (Red et al. 2013). However, one area that has not been addressed sufficiently is security.

Security is a vital component of every corporate structure, and organizations that work with engineering design data are no exception. Engineering design data represents the intellectual property of an organization and is of critical importance and significant worth and must be protected from falling into the wrong hands. In single-user architecture CAD, security is less of a challenge because the design data is usually confined to a single workstation or in a private network that is usually protected by physical isolation and by the limited accessibility of in-house networks. In a multiuser collaborative environment, however, further security challenges are presented due to the open and insecure nature of the Internet over which a lot collaborative work will be carried out. The fact that there is a multiuser component also brings in the challenge of identity management. Lack of a robust security infrastructure will thus be a huge disincentive to deployment in an enterprise environment.

## **1.2 Thesis Statement**

In line with the importance of security with regard to the v-CAx project this thesis asks the question: Can the v-CAx project be successfully deployed in an enterprise environment while meeting information security requirements of secure communications, authentication, and authorization?

To answer the question, this research proposes a security framework to implement authentication, authorization and secure communications in a collaborative multi-user CAD software system. The research solely focuses on the NX Connect prototype in the v-CAx project (Winn 2012). As part of the research the NX Connect application architecture is reviewed. Situations that could compromise any of the above mentioned security requirements are identified and presented. Several existing security technologies are explored and the best suited are selected. A methodology for developing the security framework using the selected technologies is presented and implemented. Finally the framework is integrated into the NX Connect application and tested for compliance with research objectives.

## **1.3 Assumptions**

This research assumes that the target deployment environment of v-CAx consists of only Microsoft technologies, and thus is only applicable to such an environment. Existing best practices for security will be used. It is assumed that the current CAx applications can be adapted as necessary to work with the identified methods.

## **1.4 Delimitations**

The product of this research is a proof of concept and does not exhaust the security needs of the v-CAx application in its entirety. The framework has been tested only with the NX Connect

prototype of the project. Other CAD implementations will need to adapt the framework to suit the needs of those specific applications, making changes where necessary. The security technologies and standards were selected and applied using most recent versions and best practices reasonable at the current time. It cannot be guaranteed that such technologies and standards will be applicable or be the best options in future. This research is limited to the Microsoft platform. Other platforms are not addressed.

## **2 REVIEW OF LITERATURE**

The v-CAX project was initiated with the objective of transitioning CAD applications from the traditional single-user architecture to a multiuser collaborative paradigm. In this research an attempt is being made to enhance this development by proposing a framework to ensure secure operations within the system. For more than a decade, researchers have been testing the multiuser collaborative paradigm, and several prototypes, have either been proposed or developed. While such prototypes have not been adopted into mainstream CAD applications, they present useful insights that can be applied in this research.

A review of the literature will be presented with a focus on system architecture and security considerations of previous work in the field of multiuser collaborative applications. The current state of the v-CAX architecture (without the security framework) will be reviewed. Finally, the review will also explore existing security technologies and standards which could potentially be used to implement the framework.

### **2.1 Current State of Collaborative Design in the Engineering Industry**

The need to support collaboration among users for the facilitation of everyday tasks, communication, work, and training has been recognized since the early days of computer usage (Bouras, Giannaka, and Tsiatsos 2009). During the past decade this need for collaborative work has increased, and industries that make use of Computer Aided Design (CAD) applications are no



exception. There is a growing need to meet increasing demands of globally collaborative design, reduced product development timetables and the outsourcing trends in manufacturing. A collaborative CAD system allows engineers and designers to share their work with globally distributed colleagues in either a private or public network environment. For example, Volvo developed a station wagon through a global collaboration among designers in Sweden, Spain, and the United States. Software called Alias allowed designers on both continents to share and edit engineering drawings (Naughton 2013). In addition collaborative systems also allow engineers and designers to work closely with suppliers, manufacturing partners, and customers across organizational boundaries and time zones to get valuable input into the design process (Li et al. 2005). According to (Shen, Hao, and Li 2008) the benefits for the collaborative paradigm for CAD systems also include better product quality, shorter lead-time, more competitive cost and higher customer satisfaction.

To take advantage of the benefits of collaborative work, companies have resorted to several technological innovations to create a collaborative environment for engineers and designers. According to a recent survey conducted on modern collaborative engineering practices at major industries in the U.S. it is apparent that product development personnel, particularly engineers, always strive to find ways to collaborate using various modern technology tools (Red et al. 2013). Tools used to aid collaboration in these companies include telephones for voice calls and text messages, emails, company wikis, and screen or application sharing technologies. Screen or application sharing technologies appears to be the key collaborative tool used in the engineering industries mainly because it allows engineers to visually communicate details while at different locations. It is also apparent from the survey report that the use of these tools for collaboration constitutes about 76% of the time of engineers and designers, thus confirming that collaboration

is already widespread, but that the existing tools to accomplish it are rather primitive and not necessarily well-suited to engineering design use-cases.

## **2.2 Previous Work on Multi-User Collaborative CAD Systems**

Over the past decade or two the collaborative deficiencies of CAD software have motivated research work in industry and academia with the aim of renovating CAD systems to be distributed and collaborative (Li et al. 2005). Within this period a host of researchers have demonstrated that multiuser collaboration in a CAD environment is both feasible and desirable. Moncur et al. (2013) observes that the research has resulted in two approaches for multi-user CAD design. In the first approach, CAD tools are developed from the ground up, creating an application architecture in such a way that collaborative features are native to the application. Examples of systems that have been developed in this manner include MUG (Cera et al. 2002), CADDAC (Ramani et al. 2003), e-Assembly (Chen, Song, and Feng 2004), WebSpiff (Bidarra, van den Berg, and Bronsvort 2002), WPDSS (Qiang, Zhang, and Nee 2001) among others. These systems, however, have only remained at the prototype phase and have not been developed into commercial grade software.

The second approach, also known as Transparent Adaptation (TA), makes use of plugins provided by an existing CAD application to add collaborative capabilities. The various collaborative prototypes being developed in v-CAX project, including NX Connect, make use of this approach. Details will be discussed in detail in subsequent sections.

### **2.2.1 Multiuser Architectures**

The architecture of multiuser collaborative systems, as depicted in previous research work, can be generally grouped into three categories – these are thin client-strong server, strong client-thin server and peer to peer (Fan et al. 2008). In a thin client-strong server model there is a central

server where all modeling operations are executed and the central CAD model is stored. The clients are responsible for visual rendering tasks and interactions between the clients and server. In a strong client, thin server model the each client handles all modeling operations and puts a whole geometry kernel in each client. The server in this case plays the role of an information exchanger to broadcast messages generated by a client to other clients during a collaborative design session (Li et al. 2005).

However, no matter what architecture is adopted, there are some basic components that are common to all collaborative systems. (Zhang, Wang, and Zeng 2008) identifies these components as follows:

- Data management component, which stores and manages distributed and replicated data.
- Communication component, which provides a means for data to be exchanged between components of the application
- Viewing component, which renders the actual design to the user.

NX-Connect, the prototype being used in this research makes use of the strong client-thin server model. A review of its architecture will be presented next.

### **2.2.2 NX-Connect**

NX Connect is an add-on to an existing Computer Aided Design (CAD) software, Siemens NX Unigraphics. By integrating this add-on with the CAD software, multiple users are able to access and make changes simultaneously to a single part file. The add-on makes use of an extensive API library (NX Open) provided with the CAD software. This approach of introducing multi-user functionality to an existing CAD software is known as Transparent Adaptation as mentioned in section 2.2.

As mentioned in section 2.2 NX Connect uses a strong client - thin server architecture. This means that the server retains all necessary information for the part file and each of the workstations will then utilize that information to create and modify identical parts on each of their respective workstations. The server also retains and makes available each change made during the design session. Each client generates a local copy of the part file and utilizes information gained during the design session to locally update the part. Therefore most of the computation and processing is done on the local workstation, allowing for a much faster runtime environment (Red et al. 2010).

### **2.3 Multiuser CAD Security Requirements**

Having identified the architectural components of Engineering Collaborative design systems, their security requirements will now be discussed. Information security refers to methodologies to protect and defend information and information systems by ensuring their availability, integrity, authentication, confidentiality, and non-repudiation. In the field of engineering design, information security is critical due to the prevalence of intellectual property (IP) which is of significant worth and represents a competitive advantage (Red et al. 2013). With single-user CAD architectures, deployment is usually confined to a private network. This allows for greater control over the security of data since the CAD application usually relies on the tight security of the corporate network. In a multiuser collaborative environment, however, the deployment of the CAD application extends beyond the boundaries of the private corporate network to include public networks like the Internet. Taking into consideration the open and insecure nature of the Internet, the characteristics of cross-organization collaborations and the complexity and the sensitivity of engineering design data, multiuser CAD architecture requires extra security mechanisms to be in place.

### **2.3.1 Security Risks in a Multiuser Collaborative CAD**

By transitioning to multiuser collaborative architecture, the CAD application is exposed to any one of the following threats:

- Unauthorized system access
- Server identity theft
- Eavesdropping/ Sniffing
- Unauthorized user operations

Ideally only users who are authorized to use the CAD application should be allowed access to the system. In a traditional single-user environment, the client workstation or the private corporate network usually implements authentication mechanisms to ensure that only authorized users can gain access to the CAD software. In a multiuser environment, especially one that spans beyond the private corporate network, the risk of having unauthorized access increases due to the insecure nature of the Internet. An attack of such nature is possible if an attacker obtains valid user credentials and uses them to gain access into the system. This can be possible through brute-force and dictionary attacks.

It is also possible for an attacker to take on the identity of an authentic server and have users access a fake server rather the authentic server. The risk involved here is that sensitive design data can be made available to the wrong person. Such an attack is possible through means including DNS (Domain Name Service) cache poisoning as described by (Friedl 2008).

Every communication done over a network can be sniffed or captured using various network capture tools. If the communication happens to be done in clear text then the sniffed data can easily be read in its original form. In a private network where only authorized users are allowed the risk of such an attack is low but not so in case of the Internet.

Lastly, when users gain access to the system they should be able to have access to and perform operations only as they are permitted. An attacker can take advantage of a vulnerability in the system to gain elevated privileges and perform operations which hitherto were not permissible.

The exposure of a multiuser CAD system to any of these threats can result in some devastating consequences and thus steps need to be taken to minimize the risks and mitigate the effects of such an exposure. It is the goal of this research to design a framework that will implement security mechanisms in a multiuser collaborative environment which will mitigate these risks.

### **2.3.2 Previous Work on Multi-user Collaborative Security**

The literature reveals that substantial work has been done in establishing strong security mechanisms for general IT systems based on the security concepts of availability, confidentiality, integrity, non-repudiation and authentication. Few of these, however, have focused on security in the context of multi-user collaborative CAD (Zhang, Wang, and Zeng 2008). Zhang posits that even though a collaborative system requires all the security properties that a general IT system requires, it also requires some unique properties that do not necessarily apply to all IT systems in general. An example is fine grained access control on geometric parts and assembly models. In this section some previous work in the context of multi-user collaborative security will be reviewed.

In a prototype collaborative product definition management (PDM) system developed by (Rouibah and Ould-Ali 2007) , a security mechanism is implemented based on CLAVISTM, a full-strength security framework, which includes state of the art standards such as public key infrastructure (PKI), transport layer security (TLS) for online interaction and Secure Multipurpose Internet Mail Extensions (S/MIME) for offline communication (e-mail), as well as a certificate

server, integrated with a LDAP (Lightweight Directory Access Protocol) server (Rouibah and Ould-Ali 2007). There is no record of an implementation of this system beyond the prototype developed, but recognizing the similarities in the operations of a collaborative PDM system and a collaborative CAD system, the principles applied here could be tested and applied to the NX Connect prototype with appropriate modifications and adaptations.

Kim et al. (2011) proposed an approach for a hybrid access control model which combines Role-based Access Control (RBAC) and Mandatory Access Control (MAC) models. RBAC and MAC are widely used access control models and are often used together in domains where both data integrity and information flow are concerned. In their approach, RBAC and MAC are designed in terms of features and features are configured based on requirements. Configured features are then composed to produce a design model that supports hybrid access control. The approach enables systematic development of hybrid systems of RBAC and MAC and reduces development complexity and errors through need-based configuration of features in early development phases. The approach is demonstrated using a hospital system. Even though the proposed hybrid access control model approach did not target collaborative systems specifically, the features described could well be applied with necessary modifications.

Even though it was not specifically mentioned in their paper, Christopher D Cera and Regli (2004) applied the concept of hybrid access control in a technique called Role-based Viewing for collaborative 3D geometric model design. In this technique a variable level-of-detail meshes is created across both individual parts and assemblies to provide a model suitable for access rights for individual actors within a collaborative design environment. This result is achieved through an integration of RBAC and Bell-La Padula model which is an implementation of MAC.

With the right modifications and adaptations, these research work provide a foundation that can be built upon to produce a security framework for NX Connect, the multi-user collaborative CAD prototype used in this research.

### **2.3.3 Security Requirements and Options for Multi-user CAD**

To make it simple to analyze the security requirements of multiuser collaborative CAD (Zhang, Wang, and Zeng 2008) proposed a layered approach. Three security layers were identified as follows:

- Data and communication layer
- Access control layer
- Collaborative process layer

In section 2.2.1 three basic components were identified as forming part of any multiuser collaborative architecture as identified by (Zhang, Wang, and Zeng 2008). Each one of those components could be assigned to one of the security layers. For example the data management and communication components belong to the data and communication layer. The viewing component belongs to the access control layer. The purpose of the collaborative process layer is to coordinate activities in the design process. The rest of the discussion in this review will be done in the context of these three security layers.

Each security layer has specific security challenges associated with it. The security challenges associated with the data and communication layer include confidentiality, data integrity, origin integrity, and availability. The access control layer is concerned with authentication and authorization. For the purpose of this research the security issues that will be considered are confidentiality, origin integrity (authenticity), authentication and authorization.



## **Confidentiality**

Confidentiality is the assurance that information is not disclosed to unauthorized individuals, programs, or processes (Harris 2012). A common mechanism used to ensure confidentiality is encryption. Encryption ciphers data to make it unreadable if intercepted. Common encryption algorithms include DES (Data Encryption Standard), AES (Advanced Encryption Standard) and RC4. These are all examples of symmetric encryption algorithms. Asymmetric encryption algorithms include Diffie-Helman and RSA. Several protocols and standards have been developed to implement mechanisms to enforce confidentiality in networked environments. Examples include Transport Layer Security (TLS) and Internet Protocol Security (IPSec). Rouibah and Ould-Ali (2007) demonstrated how TLS and Public Key Infrastructure (PKI) standards and protocols was used to implement security in collaborative environment in a prototype they developed for a collaborative product definition management system. The prototype's architecture used TLS for ensuring secure communications between the various clients and servers in the system. A local Certificate Authority integrated with a directory server was responsible for the management of digital certificates for use by TLS.

## **Integrity**

Integrity provides assurances that data, whether in storage or transit, has not been modified. In the context of communication the definition of integrity can be extended to include the assurance that the data is from the sender as it is claimed (Zhang, Wang, and Zeng 2008). This is also known as authenticity. Cryptographic hashing and digital signatures are two mechanisms that are mostly used to ensure integrity. To ensure authenticity in collaborative environments, use of digital signatures have often been proposed (Rouibah and Ould-Ali 2007).

## **Authentication and Authorization**

Authentication and Authorization are two processes that usually complement one another. While authentication ensures that only legitimate users have access to a system, authorization manages the rights of the users after they have been granted access to the system. Several techniques exist to implement authentication including Kerberos, TLS and directory services (Harris 2012). Kerberos was designed to provide authentication services for large distributed systems. It is currently included in several network operating systems including Windows Server (2000 to present editions). The TLS protocol also provides mechanisms for authentication in addition to encrypting data. The authentication is based on X.509 certificates that are present on a server and optionally a client. In a mutual authentication scenario which requires a server to authenticate to a client and vice versa, both server and client must have certificates issued to them. Most enterprises have some type of directory that contains security and identity information pertaining to an organization's network resources and users. Most directories follow a hierarchical format based on the X.500 standard and a type of access protocol, as in LDAP, that allows users and applications to interact with the directory. Directory services can also be used in conjunction with other authentication mechanisms such as Kerberos and certificate management systems.

The access control layer is responsible for ensuring authentication and authorization, thus ensuring that only authorized users can access specific geometric parts and assembly models. Several access control models exist that provide a framework to dictate how subjects or users access objects or resources in a system. Three main models as recognized in the literature include Discretionary, Mandatory and Role-based access control models (Harris 2012). While each of these models have their merits and demerits, other models have, over time, been developed that bring together the strengths of these models.

### **3 METHODOLOGY**

The objective of this research is to design a framework to implement authentication, authorization and secure communications in a multiuser collaborative CAD environment. The design is intended to be extended and applied to the broad field of computer aided applications (CAx). This framework will make it possible to deploy such applications in an enterprise environment while satisfying basic security requirements. The design is validated and tested on a specific CAD platform. The design for the framework was developed by creating security structures for the NX-Connect plug-in used with the NX Unigraphics CAD software from Siemens.

The current state of the NX Connect application was reviewed. Previous research work in the field of multiuser collaborative CAD and how they handled security were also reviewed. Available security technologies and standards were explored. Based on the original architecture of v-CAx and building upon previous research work and applying current technologies and standards, a security framework was designed to accomplish the objectives of the research. NX Connect was reverse engineered and integrated with the security framework in a manner that did not disrupt its original functionality. To ascertain the correct functionality of the framework a test platform was developed.

### 3.1 NX Connect Architecture

This section presents the architecture of NX Connect as applied to NX Unigraphics CAD software from Siemens. Figure 1 shows the layout of the current NX Connect architecture.

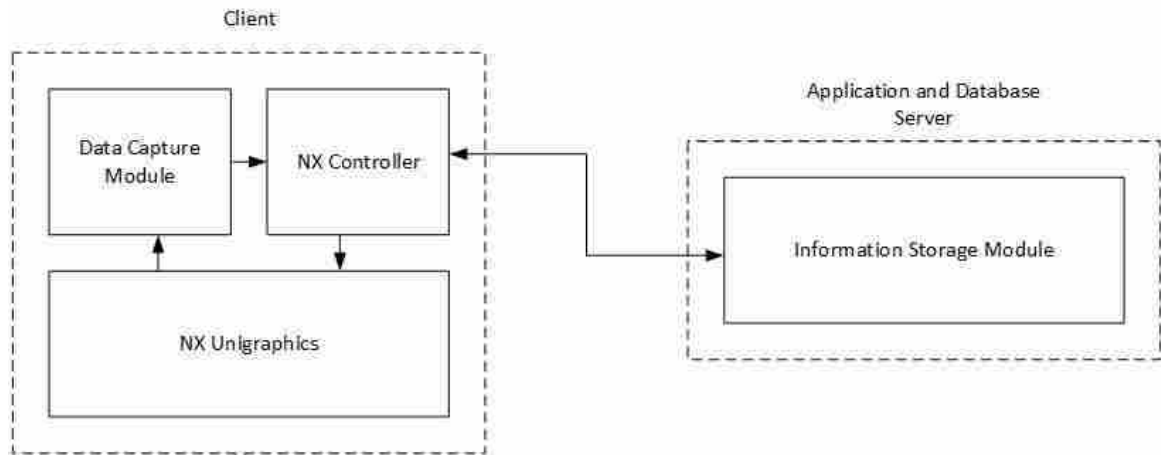


Figure 1 Architecture of NX Connect

#### 3.1.1 NX Unigraphics (U-G)

NX U-G is an advanced CAD software package used primarily in engineering for design, engineering analysis, and manufacturing finished design by using included machining modules (Siemens 2013). An application programming interface (API) made available for NX by Siemens allows NX to be extended to provide more capabilities. It is this API that made the creation of NX-Connect plugin possible.

#### 3.1.2 NX-Connect

NX-Connect is a plugin for NX U-G developed at the Mechanical Engineering Department at Brigham Young University. The plugin makes use of the API (mentioned in previous section) to capture data from NX U-G and communicates it to the v-CAX server which in turn pushes it to

other connected clients and stores it in the database. The plugin is also responsible for serializing data received from NX U-G into XML data in compliance with the SOAP protocol. This is necessary because the data received from NX U-G is not compatible with the v-CAX server and the database (Winn 2012). The end result is that multiple users are able to access and simultaneously make changes to a single geometric model. Each user is able to view the part independently and use the zoom and rotational functions of NX without affecting the view of the remaining users. This allows each user to focus on the intended task while viewing updates made to the part file by other users (Red et al. 2010).

NX Connect comprises the following components: Information Storage Module (ISM), Data Capture Module, Data Sync Module and NX Controller. The following sections will discuss how these components work together to achieve the needed functionality and how the proposed framework will be integrated.

### **3.1.3 Information Storage Module (ISM)**

This module uses Microsoft's SQL Server and a hierarchical structure to store and sync the part features and related data, including all information relating to users, parts, and features. It broadcasts changes from each user to all other users as it receives them (Red et al. 2013).

### **3.1.4 Data Capture Module (DCM)**

To eliminate the need for each NX client to constantly send information to the database the DCM monitors the NX session to determine when changes have been made to the part file. When a change is detected the DCM determines the feature that has been created, modified, or deleted and then alerts the NX Controller of the change, then passes the change information for the modified feature.

### **3.1.5 Data Sync Module (DSM)**

The DSM monitors the ISM to determine if any changes have been uploaded by another user. Upon finding a change to the database, the NX Controller is alerted to the changes

### **3.1.6 NX Controller**

This module receives client model changes from the ISM in the form of database objects, extracts the parts and feature information and then renders the corresponding parts and features in the NX session on the user's machine using the NX API (NX Open). The module also converts all parts and feature information to primitive values to allow storage in the database.

## **3.2 Review of NX Connect Security State**

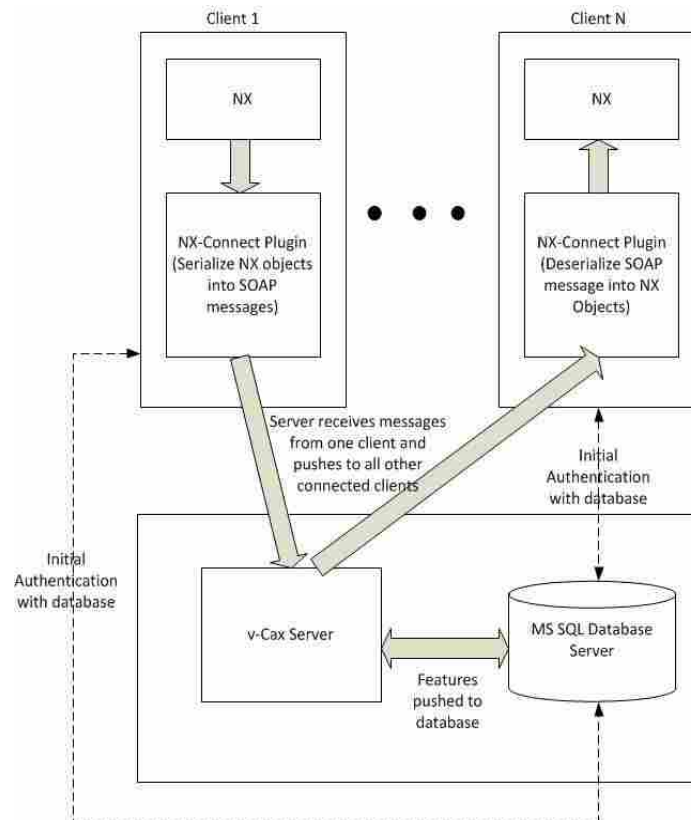
The architecture of NX Connect as described in the previous section has very minimal security. Figure 2 describes the current security situation. It can be seen that user authentication is accomplished by the client directly querying the database. Once a user is authenticated all views (parts and features) are visible regardless of the user's role in the design process. Also all communication between the server and the client is carried out in plain text. This presents a situation where the confidentiality and data integrity can be adversely compromised.

To improve the security of the system a security framework was designed to meet the following objectives:

- a. Confidentiality will be ensured during transmission of data over the network through the use of encryption techniques.
- b. Authentication and authorization/access control will be achieved through the integration of NX Connect with a directory service. The purpose of authentication is to confirm the identity of a user trying to access the system and the purpose of authorization is to ensure

that an authenticated user only has access to the resources and actions in the system which he is permitted.

- c. The implementation of the security framework will not result in substantial degradation in the performance of the application.



**Figure 2 Original Architecture of NX Connect**

### 3.3 NX Connect Security Framework Overview

Several security technologies and standards were reviewed. This provided an understanding of what is available together with their strengths and weaknesses. With this knowledge appropriate technologies and standards were chosen to accomplish the objectives of this research.

### **3.3.1 Authentication and Authorization**

Before a client or a user is allowed access to NX Connect, it is very important for the user to be identified and for that identity to be proven. This process is known as authentication. Once authentication is successful, the system then has to determine the operations that can be performed by that user. This is known as authorization. It is important to recognize that even though authentication and authorization are closely related principles each has distinct functions in the security process. For example, a user who has successfully been identified and authenticated in NX Connect may not be allowed to view any geometric model or perform an action on the models. Likewise a user may have legitimate rights to view a particular geometric model but until that user is successfully authenticated, the view privilege is not operational.

While there are several technologies that implement authentication and authorization, this research focused on the use of directory services. Most enterprise networks have some form of directory service that is used to manage the company's users and resources. Most of these directories follow a hierarchical database format, based on the X.500 standard, and a protocol, like the Lightweight Directory Access Protocol (LDAP), that allows users and applications to interact with the directory (Harris 2012).

Rather than implementing a fully-featured directory based authentication and authorization system within NX Connect the choice was made to integrate with a well-proven directory service. This arrangement provides the advantage of administering user accounts and their permissions from the same central repository as the rest of the organization. Microsoft Active Directory (AD) is a directory service based on the Lightweight Directory Access Protocol (LDAP) (Microsoft Technet 2011). The strengths of AD including security, scalability, and extensibility make it an appropriate choice for handling authentication for NX Connect users (Gaehtgens 2014).



As a best practice for ensuring strong authentication, it is recommended that at least two different authentication factors be used. Authentication factors generally include the following:

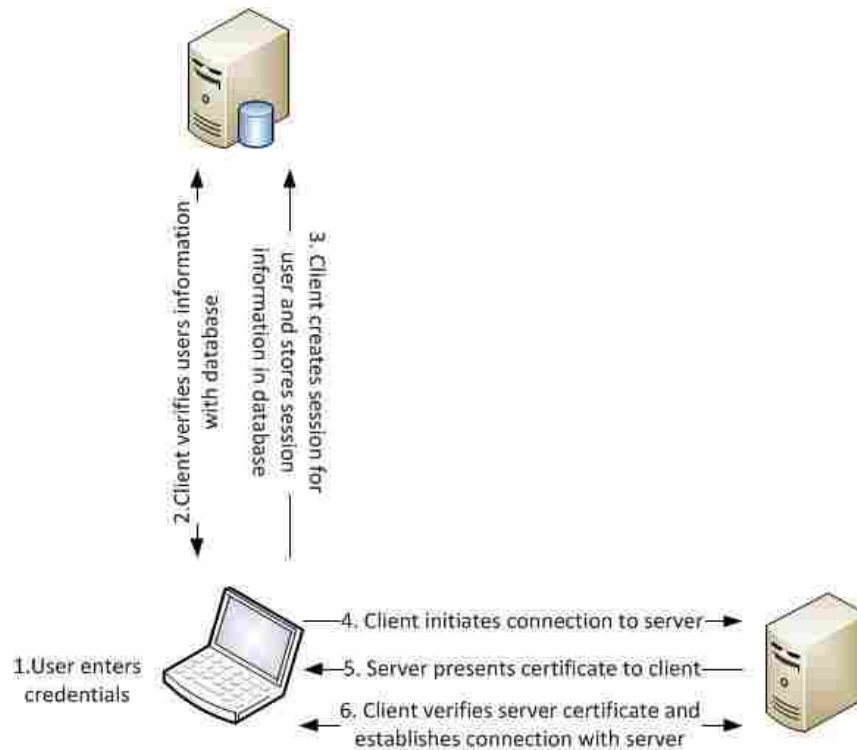
- Something a user knows – username and password,
- Something a user has – smart card, digital certificate,
- Something a user is – a physical (biological) attribute like fingerprint

The integration of the framework with Active Directory, as so far described in the previous paragraphs, allows authentication to be accomplished with something that the user knows, as in username and password. To make the authentication process stronger, digital certificates and a Public Key Infrastructure (PKI) framework are also used. This represents authentication using something that the user has. The use of these two authentication methods constitutes a two factor authentication process. Another best practice for authentication systems requires that a lockout mechanism be put in place as a counter measure for potential brute force or dictionary attacks. Details of implementing this mechanism is given in chapter four.

### **Authentication Design**

A workstation was available that included a working copy and some documentation of the software packages under discussion. A review of the NX Connect authentication processes and source code, and experimentation with the system revealed the authentication mechanism. The original authentication mechanism in NX Connect is shown in Figure 3. The mechanism as shown is achieved with the client querying a SQL database that stores user identity information and confirming the validity of user supplied credentials with the query results. With this approach the database server is exposed to the Internet and for that matter anyone who cares to look. Thus, even

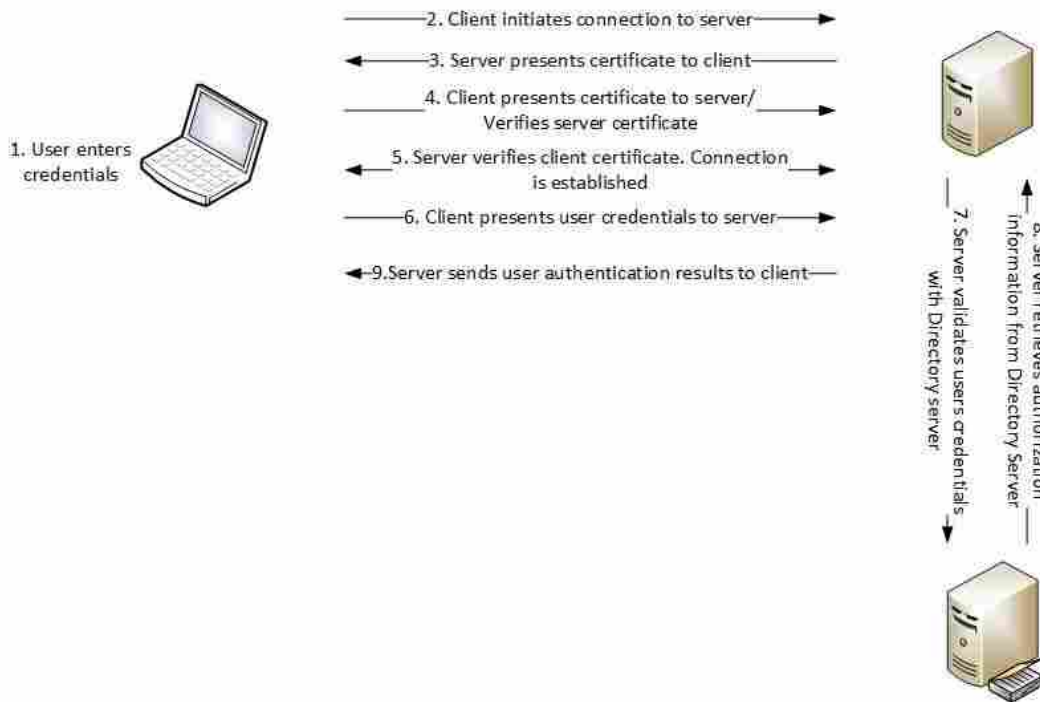
though this approach works, the exposure to the Internet represented an unnecessary risk taking and thus necessitated the need for the redesigning of the authentication mechanism.



**Figure 3 Original Authentication Mechanism for NX Connect**

The new design for the authentication mechanism is shown in Figure 4. This design integrates NX Connect with Active Directory which stores all user identity information. The authentication process involves two stages. First the client initiates a connection with the NX Connect server. During this initial handshake the client and the server engage in a mutual authentication process where certificates are exchanged between the client and server and verified. When the first stage of authentication is successful, a connection is established and then user credentials are passed on to the server. At this point the server acts as a proxy and authenticates the user credentials with the directory server. When authentication is successful a user sessions is

created and the session information is sent to the client. The user may at this point be able to use the application. This design eliminates the exposure of the database server to the Internet. Details of the implementation of this design is given in chapter four.



**Figure 4 Proposed Authentication Mechanism for NX Connect**

### **Access Control (Authorization) Design**

The existing security mechanisms included very rudimentary access control capability. A finer grained access control mechanism was developed. The new method allows for part-level and user-level control. It operates as discussed below.

After a user is successfully authenticated the access control (authorization) mechanism ensures that the user only has access to CAD models that the user's security clearance permits. The mechanism also regulates the operations that can be performed on CAD model based on

defined roles of which a user forms part. The security framework was designed to allow for the implementation of the Mandatory Access Control (MAC) and Role-Based Access Control (RBAC) models. MAC is based on a security label system. In this research the labels used include security classes and security categories. RBAC uses the concept of roles which is defined in terms of the permissions and actions that members of that role can carry out.

### **Security Class**

The security class of a subject or object determines sensitivity levels. In order for a subject (user) to have access to an object (CAD model), the subject must have a security class that is higher or equal to the security class of the object. Security classes follow a hierarchical structure. This means that security classes have associated levels with one level trusted more than the other. For example, in a military setup, security classes may include unclassified, confidential, secret and top secret, with increasing levels in that order.

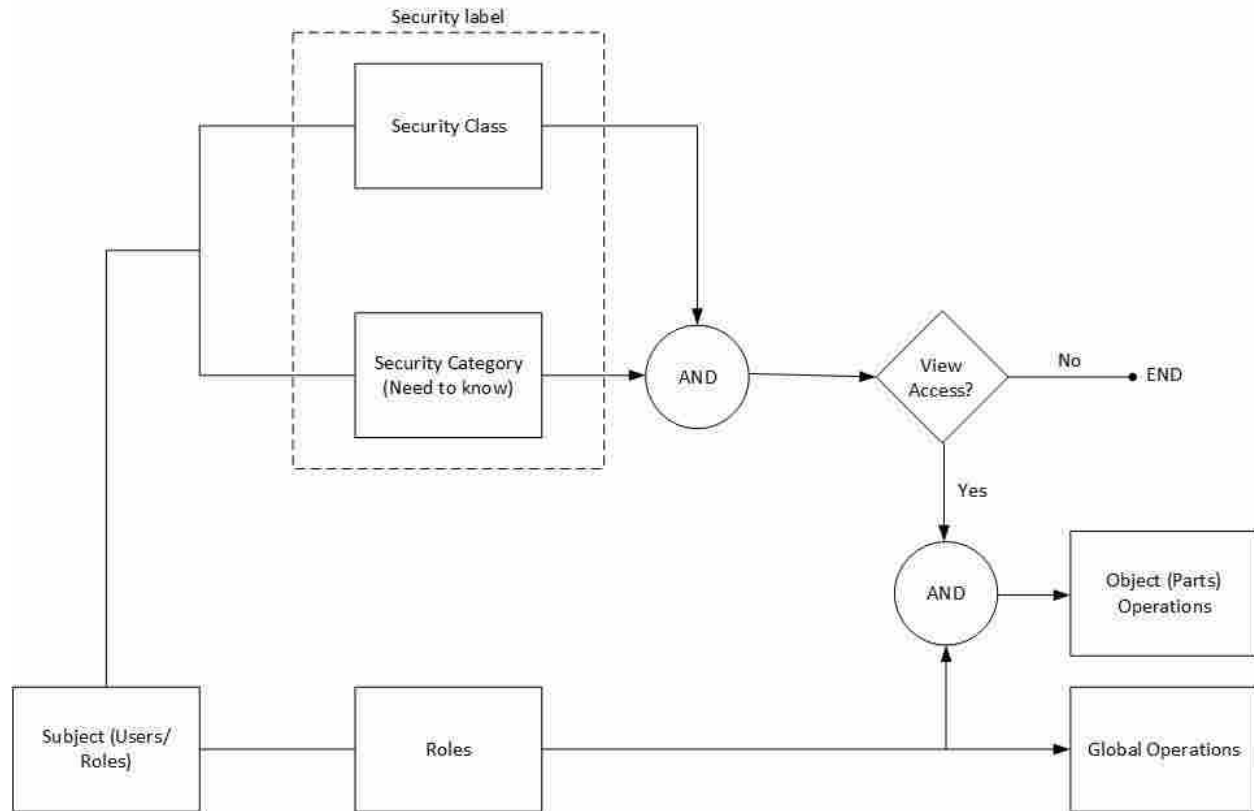
### **Security Category**

Security categories enforce need to know rules. The fact that user A, for example, has a certain security clearance (security class) does not guarantee access to a CAD model, even if that CAD model also belongs to that security class. User A must also belong to the security category of the CAD model part in question. Security categories do not follow a hierarchical structure like security classes, and may be based on projects, geographic locations and so on.

## **Role**

Rather than assigning permissions on objects to individual users, a role can be created to which a user may be assigned. This simplifies the process of granting permissions. Granting permissions to roles is a feature of RBAC. The design of RBAC in the framework is hierarchical, which means that there is an accumulation of permissions of other roles or in other words privilege inheritance. For example, a Technician role may have certain permissions and the Junior Engineer role also has its permissions. A Senior Engineer Role inherits the permissions of the two roles in addition to other permissions assigned to the Senior Engineer role. The operation of the access control model as described is shown in Figure 5.

As shown in Figure 5, a user may be assigned a label comprising a security class and security category. The user's label (both security class and category) must match that of a CAD model's label before the user is granted access to that part. Roles may also be assigned to users. After a user has been granted access a geometric part, the user's roles determine the operations that can be performed on the CAD model. Such actions may include modifying, deleting, or creating a CAD model. The permissions that are assigned to roles fall under two categories: Global permissions and object permissions. Global permissions apply to all CAD models in the NX database. Object permissions on the other hand apply only to specific objects. For example, a global permission "Delete" means that a user has permission to delete any CAD model. An object permission "Delete" for the CAD model "PartA" means that the "Delete" operation can only be performed on PartA. The redesigned access control required changes in the data management structure.



**Figure 5 Operation of NX Access Control Model**

### **Redesigning the Information Storage Module (ISM)**

The re-designed access control mechanism required MAC and RBAC capabilities to be retrofitted into the original NX Connect architecture. Implementing this required several changes in various parts of the system. The Information Storage Module component of NX Connect, especially, required several design modifications in the SQL database schema. Additional design was also necessary to integrate the access control framework with Active Directory.

As mentioned earlier, the ISM uses a SQL database and a hierarchical structure to store and sync part features and related data. In NX Connect, Parts constitute Assemblies, and Assemblies may constitute other assemblies and so on. For this research Parts are considered the fundamental components of CAD models. Thus all access control rules are applied at the Part

level. The access control rules of an Assembly depends on the rules that have been applied to the parts that make up that assembly. Making the ISM access control capable required existing tables to be modified with extra fields, and in some cases new tables were designed and created.

### **Custom Active Directory Objects for Access Control**

The previous section mentions how the ISM needed to be modified to make it capable of handling access control. The ISM alone, however, does not accomplish the entire access control mechanism. The access control mechanism requires that the ISM interacts with Active Directory objects in determining the permissions required by users to perform certain operations. Custom Active Directory objects were designed, using group principals, for Security Class, Security Category, Roles and Global Permissions. The details of this design are provided in chapter four.

### **3.3.2 Confidentiality**

Another of the objectives of this research is to establish a secure communication channel between NX Connect clients and the server. This is to ensure confidentiality when data is transmitted through the channel. A common approach to implement confidentiality is by encrypting the data to be exchanged before transmitting it in the communication channel. While several protocols exist to implement encryption of data during transmission, TLS was chosen. TLS makes use of digital certificates to store the keys that are used during the encryption process. In addition to encrypting data during transmission, TLS also provides a means to authenticate the server to the client or the clients to the server using digital certificates. This feature is useful to enforce origin integrity or authenticity, or in other words prove to the client that the server is who it says it is, or prove to the server that the client is who it says it is.

### **3.3.3 Certificate/Key Management (Public Key Infrastructure (PKI))**

PKI consists of programs, data formats, procedures, protocols and policies working so as to enable a wide range of users to communicate in a secure fashion. PKI provides a number of security functions including confidentiality, integrity and authentication. To understand how PKI provides these services, it will be necessary to understand the functions of the various components that make up a PKI. The following components will be discussed:

- Certificate Authority
- Digital Signatures
- Certificates
- Chain of Trust

#### **Certificate Authority (CA)**

A CA is a trusted third party that maintains and issues digital certificates. Usually an individual or organization will request a certificate from a CA. If the CA is satisfied with the request, it goes ahead to create a certificate, signs it and send it back the requester. The CA maintains this certificate over its lifetime. With the certificate signed by the CA, it is basically vouching for the legitimacy of the individual who possesses the certificate. Thus if Bob has a certificate signed by a CA, and presents that certificate to Alice, Alice can trust Bob because she trusts the CA. CAs can be internal to an organization. This gives the organization more control on how certificates are created, maintained and revoked. The framework being proposed recommends using an internal CA.



## **Digital Signatures**

A digital signature is a hash that has been encrypted with the signer's private key. This provides a means to ensure integrity, authenticity and non-repudiation of a message. For example if Bob signs a document with his private key, Alice can verify that the message truly came from Bob by decrypting the signature and comparing it with her own hash of the message.

## **Certificates**

A digital certificate is a document that associates a public key in an asymmetric cryptographic key pair with the identity of an individual or organization. A common standard for certificates is the X.509, which defines the various fields that the certificate may have. A CA signs a certificate by generating a hash of the certificate and encrypting the hash with its private key. If a user wants to verify the certificate, it uses the public key of the CA to decrypt the hash and compares it user's own generated hash of the certificate.

## **Chain of Trust**

The Chain of Trust of a chain of certificates is an ordered list of certificates, containing an end-user subscriber certificate and intermediate certificates that enables the receiver to verify that the sender and all intermediate certificates are trustworthy. This is made possible by the integrity and non-repudiation properties of digital signatures.

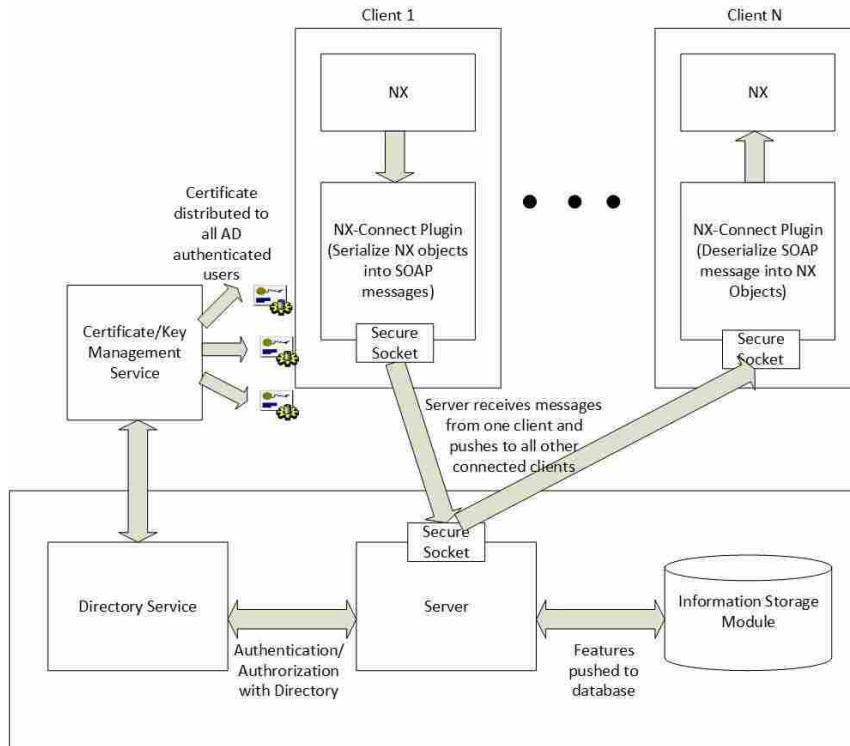
The successful deployment of a PKI will depend on how well the cryptographic keys and the digital certificates are protected from being compromised. As described earlier PKI operations are based on a trust model and so a compromise of the cryptographic keys and the digital certificates will mean the model can no longer be trusted. Thus there is more to key and certificate management than using them for the purposes of encryption and authentication. It must be ensured

that they are protected while being created, transmitted and stored on each workstation or server, and that they are being distributed to the right entities. There also has to be a mechanism in place to ensure that information about the certificates are updated regularly. Key management can be handled either through manual or automatic means.

For the purposes of this research an automatic key management service is used. This management service takes care of the creation, distribution and revocation of certificates. A detailed description of this service is given in chapter 4.

### **3.4 Security Framework Design**

Figure 6 shows how the standards and technologies described in the previous sections were employed and designing a security framework for NX Connect. Authentication has been designed to employ two factors (digital certificates and username/password) and integrates with a directory service – Active Directory. The integration with Active Directory allows user accounts to be managed from a central location. Access control capability is included to manage the operations that users can carry out based on their permissions. By employing TLS all communications between client and server are encrypted and no longer done in clear text. This design now meets the requirements for the secure framework.



**Figure 6 Proposed Architecture for NX Connect with Enhanced Security**

## **4 IMPLEMENTATION OF FRAMEWORK**

In this chapter the details of implementation of the proposed security framework, as described in chapter 3, are presented. In accordance with the objectives of this research the proposed framework seeks to create a secure environment for the operations of a multiuser collaborative CAD system. The framework focuses on providing security services including secure communications between all components of the system, origin integrity (authenticity), authentication and authorization for user subjects. The importance of these services have been elaborated in previous sections. To illustrate the implementation of the framework references will be made to the original NX Connect architecture and then compared with its new architecture after integration with the security framework. The implementation details will be presented under the following three topics:

- Authentication framework
- Secure Communications
- Authorization framework

The design of this framework does not fully handle all the security requirements of NX Connect but does well to address the basic security needs that will allow its safe operation in an enterprise environment. The entire implementation of this framework uses Microsoft technologies as in the .NET framework, Active Directory Domain Services, Active Directory Certificate Services and SQL Server.

## **4.1 Implementation of Authentication Framework**

The original architecture of NX Connect implemented authentication by directly querying a SQL database server as illustrated in Figure 3. While this mechanism works alright, it does not follow best practices for authentication procedures. A new authentication mechanism was thus needed. As several authentication mechanisms exist, one was chosen that will best suit the architecture and nature of operations of NX Connect. That this end, the choice was made integrate NX Connect with a Directory Service, specifically Microsoft Active Directory.

### **4.1.1 The Authentication Mechanism**

Figure 8 illustrates how the authentication procedure works in the new architecture where NX Connect integrates with Active Directory. First the user presents credentials for authentication. These credentials will eventually be presented to the server for validation. But first the client workstation begins to initiate a connection with the server. This connection is initiated using the TLS protocol. In addition to providing a means to encrypt data, TLS also provides authentication services. The server authenticates itself to the client workstation by presenting a certificate that has been signed by a Certificate authority that the client trusts. This proves to the client that the server is legitimate and thus ensuring origin integrity (or authenticity). The client also authenticates itself to the server by presenting a certificate that has been signed by a Certificate Authority that the server trusts. The server also goes further to verify that the subject on the certificate is actually allowed to access its services. This completes the first phase of authentication.

The second phase of authentication proceeds after only after a successful completion of the first phase as described in the previous paragraph. In the second phase, the client presents user credentials to the server. On receiving these credentials the server validates them with Active Directory. The server then retrieves authorization (access control) information for the user from

Active Directory. The authorization information is what determines the actions that the user can perform once it has been granted access to the system.

The use of two phases for the authentication process is in line with the general best practices for authentication where at least two factors are required to be presented. In this framework, the user presents something that is known (user credentials) and something that is owned (digital certificate). This improves the strength of the authentication and makes it more difficult to be compromised.

#### **4.1.2 Implementing Authentication**

The entire authentication framework was implemented with the .NET framework. The .NET framework comes with classes that allows interaction with Active Directory and easy implementation of TLS protocol.

The ‘Authentication’ class (see Figure 7) in the ‘CAxConnect.Data’ namespace encapsulates identity information that the client sends to the server to authenticate a user. The ‘SessionModule’ property of the ‘Authentication’ encapsulates data relating to the user’s session such as session ID and user authorization information (such as roles).

At the time that a client initiates a connection with the server, an instance of the ‘Authentication’ class is created and initialized with relevant user identity information. The ‘Authentication’ object (with user identity information) is sent over to the server for validation with Active Directory. This operation is accomplished by calling the ‘SendAuthenticationRequest’ method in the ‘NXConnectAuthorizationModule’ (see Figure 9). On successful validation the server retrieves authorization information from Active Directory and updates the ‘Authentication’ object with that data. The server also creates new session data for the user.

```

[DataContract]
public class Authentication : Message
{
    [DataMember]
    public string Module { get; private set; }

    [DataMember]
    public string Username { get; private set; }

    [DataMember]
    public string Password { get; private set; }

    [DataMember]
    public bool Authenticated { get; set; }

    [DataMember]
    public SessionModule SessionModule { get; set; }

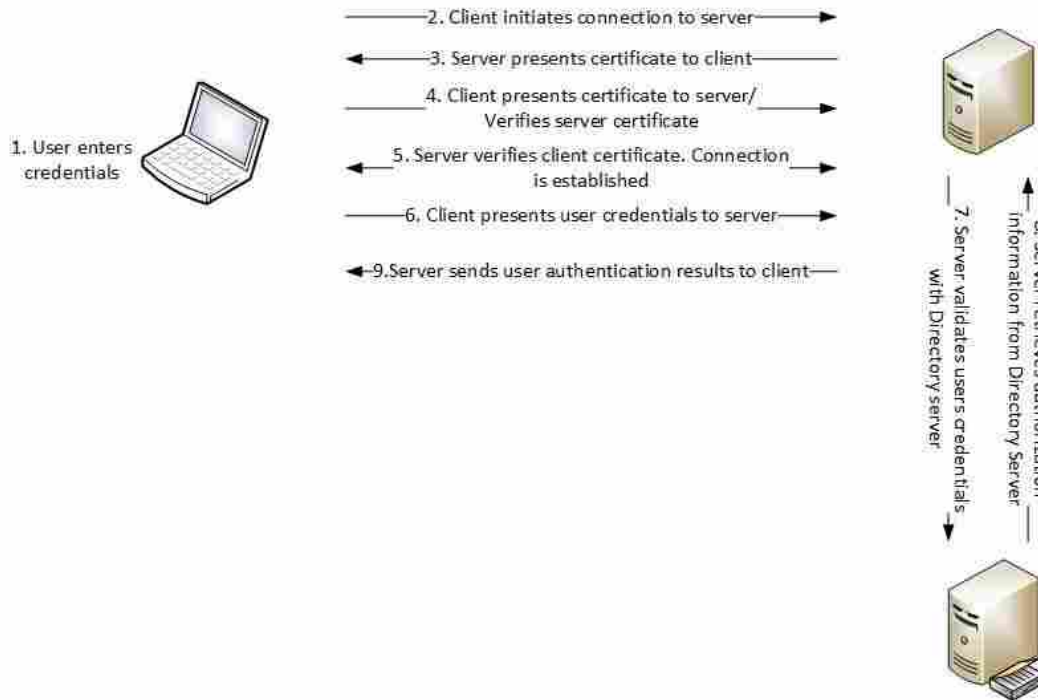
    public Authentication(string module, string username, string password) :
        base(username)
    {
        Username = username;
        Password = password;
        Module = module;
        Authenticated = false;
    }

    public Authentication(Authentication response) : base(response, response)
    { }
}

```

**Figure 7 Authentication Class Definition**

The 'Authentication' object is then sent back the client with information about the status of the success or failure of the authentication. The server's mechanism of handling the authentication message received from the client is accomplished by means of the 'HandleAuthentication' method.



**Figure 8 NX Connect Authentication Mechanism**

```
private Authentication SendAuthenticationRequest(string module, SessionAction
action)
{
    Authentication request = new Authentication(module, Username, Password)
    {
        SessionModule = new SessionModule
        {
            SessionAction = action,
            SessionID = (action == SessionAction.Close ? this.SessionID : 0)
        }
    };

    Authentication response = NXConnectApp.Synchronizer.SendSynchronous(request)
as Authentication;

    if (response.SessionModule.SessionState == SessionState.OpenNew)
    {
        sessionID = response.SessionModule.SessionID;
        UserID = response.SessionModule.UserID;
    }

    return response;
}
```

**Figure 9 SendAuthenticationRequest Method Definition**



```

private static void HandleAuthentication(Client sender, Message request)
{
    Authentication req = request as Authentication;
    if (req.SessionModule.SessionAction == SessionAction.Open ||
        req.SessionModule.SessionAction == SessionAction.ForceOpen)
    {
        req.Authenticated = Server.AuthenticateAD(req.Username, req.Password,
            req.SessionModule);
        Sender.Send(sender, new Authentication(req));
    }
    else if (req.SessionModule.SessionAction == SessionAction.Close)
    {
        Server.CloseSession(req.SessionModule.SessionID, req.SessionModule);
        Sender.Send(sender, new Authentication(req));
    }
}

```

**Figure 10 HandleAuthentication Method Definition**

### 4.1.3 Access Control (Authorization)

A dynamic linked library (dll) called 'NXAccessControl' contains classes and methods which handle all access control (authorization) operations for NX Connect. The library uses various classes from the Systems.DirectoryServices.AccountManagement namespace in the .NET framework to integrate with Active Directory and the NX Connect SQL Database. Active Directory stores the authorization information for all users who can access NX Connect. The authorization model makes use of four Active Directory objects namely:

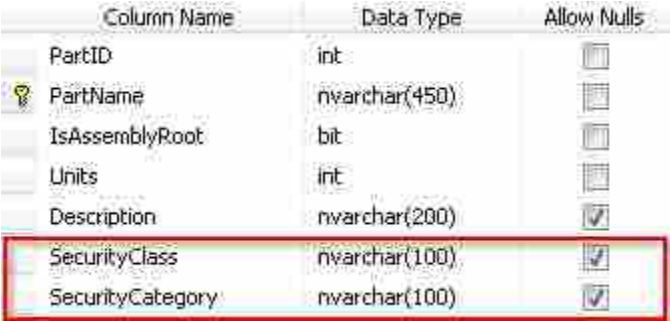
- Security Classes
- Categories
- Roles
- Global Permissions

Details on how these authorization objects were implemented to achieve the desired authorization model will be discussed in a subsequent section. The Information Storage Module (ISM) stores all information about geometric parts and the features that make up the parts. The

ISM also stores authorization data for the parts. As mentioned in chapter three, the ISM needed some redesigning to make it capable of handling access control on the geometric parts stored in it. The details of the design are now presented:

### Redesigning the Information Storage Module (ISM)

In redesigning the ISM to handle access control, two extra fields were created in the ‘Parts’ table of the ISM to add MAC capability (see Figure 11). These fields represent the security class and need-to-know category. The implication of this design is that each CAD model can have only one security class and one need-to-know category.



Column Name	Data Type	Allow Nulls
PartID	int	<input type="checkbox"/>
PartName	nvarchar(450)	<input type="checkbox"/>
IsAssemblyRoot	bit	<input type="checkbox"/>
Units	int	<input type="checkbox"/>
Description	nvarchar(200)	<input checked="" type="checkbox"/>
SecurityClass	nvarchar(100)	<input checked="" type="checkbox"/>
SecurityCategory	nvarchar(100)	<input checked="" type="checkbox"/>

Figure 11 Modified Parts Table

RBAC capability required new tables to be created. First a table was created to store the permissions needed to carry out operations on a CAD model or part as in this case. These operations include delete, modify and view. Other operations could be added as necessary. The design of this table is shown in Figure 12.

Column Name	Data Type	Allow Nulls
PermissionID	int	<input type="checkbox"/>
PermissionName	nvarchar(50)	<input type="checkbox"/>
Description	nvarchar(50)	<input checked="" type="checkbox"/>

**Figure 12 Permissions Table**

A second table was created to store Part-Subject pairs. A subject in this context could be either a user or role. The Part-Subject pair represents an entity to which permissions can be applied. For example, consider a CAD model called PartA, and a user called User-1. To grant User-1 permission to carry out a delete operation on PartA, a Part-Subject pair would be created for PartA and User-1. Permission for the delete operation can then be applied to the Part-Subject pair. Figure 13 shows the design of this table.

Column Name	Data Type	Allow Nulls
PartPermissionID	int	<input type="checkbox"/>
PartID	int	<input type="checkbox"/>
Subject	nvarchar(50)	<input type="checkbox"/>

**Figure 13 Part-Subject Pairs Table**

A many-to-many relationship exists between the Permissions and the Part-Subject pair tables. Consider, for example, the Permission data in Figure 14 and the Part-Subject pair data in Figure 15. The 'View' permission could be applied to several Part-Subject pairs. The part-subject pair '23-Senior Engineer' could also have several permissions applied to it. To manage this relationship, a junction table was needed to link the two tables. The design of this junction table is shown in Figure 16.

PermissionID	PermissionName	Description
3	View	View Part
4	Delete	Delete Part
5	Modify	Modify Part

**Figure 14 Sample Permissions Data**

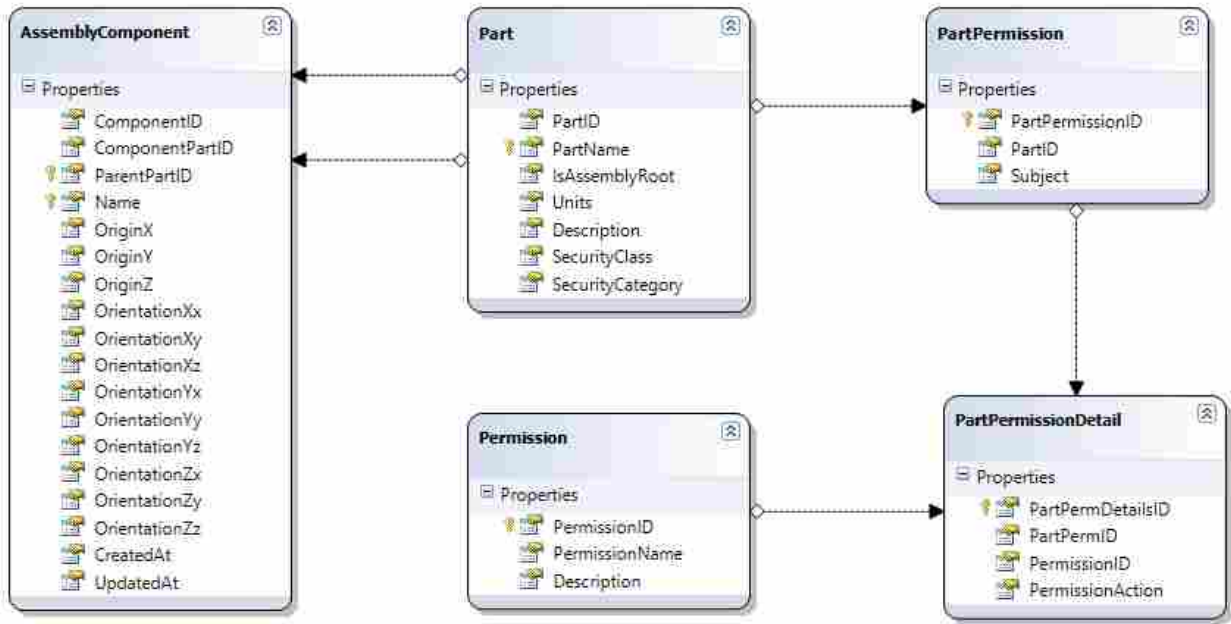
PartPermissionID	PartID	Subject
30	23	Senior Engineer
31	23	User-1
32	22	User-1
33	23	User-2
36	32	User-1
37	32	User-2

**Figure 15 Sample Part-Subject Pair Data**

Column Name	Data Type	Allow Nulls
PartPermDetailsID	int	<input type="checkbox"/>
PartPermID	int	<input type="checkbox"/>
PermissionID	int	<input type="checkbox"/>
PermissionAction	nvarchar(10)	<input type="checkbox"/>

**Figure 16 Part-Subject-Permission Table**

The 'PartPermID' field represents the Part-Subject pair, the 'PermissionID' field represents the Permission, and the 'PermissionAction' field represents the action that is permitted which may be 'Allow' or 'Deny'. Figure 17 illustrates the relationships between the various tables that implement RBAC. The design of this RBAC capability on top of the existing database was done in a manner that did not disrupt the original functionality of the Information Storage Module.



**Figure 17 Modified Database Schema for Access Control**

### **Custom Active Directory Objects for Access Control**

As was discussed in chapter three, the access control mechanism requires that the ISM interacts with Active Directory objects in determining the permissions required by users to perform operations on parts. To implement this, Custom Active Directory objects were designed, using group principal objects, for Security Class, Security Category, Roles and Global Permissions. Utility functions found in NXAccessControl.dll library allow the creation, storage and management of these objects in Active Directory from NX Connect code. The design of the Security Class object is shown in Figure 18. The property ‘NXObjectType’ indicates the type of access control object. This could be ‘SecurityClass’, ‘SecurityCategory’, ‘Role’ or ‘GlobalPermission’. In the case of the ‘NXSecurityClass’ object, the property ‘SecurityClassLevel’ indicates the level of the SecurityClass object in the security class hierarchy.

In Active Directory, these objects are all stored as Group Principals and are distinguished from each other by the value of the NXObjectType property.

```
[DirectoryObjectClass("group")]
[DirectoryRdnPrefix("CN")]
public class NXSecurityClass : GroupPrincipal
{
    public NXSecurityClass(PrincipalContext pC) : base(pC) { }

    public NXSecurityClass(GroupPrincipal gP) : base(gP.Context)
    {
        this.Name = gP.Name;
        this.NXObjectType = gP.Description;
        this.SecurityClassLevel =
        ((DirectoryEntry)gP.GetUnderlyingObject()).Properties["info"].Value.ToString();
    }
    public String NXObjectType
    {
        get
        {
            if (this.Description != null)
                return this.Description;
            else
                return null;
        }
        set
        {
            this.Description = value;
        }
    }
}
[DirectoryProperty("info")]
public String SecurityClassLevel
{
    get
    {
        object[] result = this.ExtensionGet("info");
        if (result != null)
        {
            return result[0].ToString();
        }
        else return null;
    }
    set
    {
        this.ExtensionSet("info", value);
    }
}
}
```

**Figure 18 Security Class Definition**

## Access Control (Authorization) Mechanism

An Authorization object is used to encapsulate authorization requests that are sent to the server. The definition of the Authorization class can be found in the CAxConnect.Data namespace in the framework. A typical authorization operation works as follows: When a security sensitive operation is to be performed, the ‘SendAuthorizationRequest’ method of ‘NXConnectAuthorizationModule’ creates a message containing the user information and part information for which authorization is being sought. The message is then sent synchronously as a serialized ‘Authorization’ object to the server and awaits a response. The response contains the status of the authorization request as stored in the ‘Authorized’ property of the response object. If the value of ‘Authorized’ is true, then the action is permitted. On the server the authorization request is handled by ‘HandleAuthorization’ method in the ‘MessageHandler’ class.

```
public Authorization SendAuthorizationRequest(string user, int partid,
    AuthRequestType authType, AuthRequest authReq, String permission)
{
    Authorization request = new Authorization(user, partid)
    {
        AuthType = authType,
        AuthRequest = authReq,
        Permission = permission
    };

    Authorization response = NXConnectApp.Synchronizer.SendSynchronous(request)
as Authorization;

    return response;
}
```

**Figure 19 SendAuthorizationRequest Method Definition**

## **4.2 Implementing Secure Communication Channel**

Ensuring a secure channel means that all data that is exchanged between the various components of the multi user CAD system are kept confidential from unauthorized users. To implement this feature in NX Connect the Transport Layer Security (TLS) protocol is used. TLS is a popular industry standard for encrypting network communications. TLS is a standard which provides many options for use to fit a specific need. For this research, TLS provides means for an encrypted channel using X.509 certificates the Advanced Encryption Standard (AES), as well as authentication services discussed earlier.

### **4.2.1 Implementing TLS**

The System.Net.Security namespace in the .NET framework contains classes necessary for implementing TLS. The SslStream Class, for example, provides a network stream used for client-server communication (Microsoft MSDN).

Implementing TLS in NX Connect client is accomplished using the ‘TLSCientConnection’ class found in the security framework. It stores all information relating to a TLS session established with server. Using its ‘Authenticate’ method, mutual authentication is achieved by verifying the certificate presented by the server as well as presenting a certificate to the server for verification. The actual TLS connection is initiated from the ‘ServerConnection’ class also found in the framework. The ‘Authenticate’ method is listed in Figure 20 below together with the ‘Connect’ method of ‘ServerConnection’ which initiates the connection with the server.

The implementation of TLS on the server is similar, though with a few differences. The difference lies in how the “SslStream” object handles client and server connections. The server’s implementation of TLS authentication is show in Figure 21. See also Appendix A for more details.



```

public void Authenticate(string server, X509Certificate2Collection cCollection)
{
    Stream.AuthenticateAsClient(server, cCollection, SslProtocols.Default,
false);
    if (Socket != null)
        IsConnected = Stream.IsMutuallyAuthenticated;
}

public static void Connect(IClient client, IPAddress address, int port, int
bufferSize, Boolean useTLS = true)
{
    if (Instance == null)
        Instance = new ServerConnection(client, address, port,
bufferSize, useTLS);
}

```

**Figure 20 Authenticate and Connect Method Definitions**

```

SslClient secureClient = new SslClient(new SslStream(new NetworkStream(client),
false, new RemoteCertificateValidationCallback(ValidateClientCertificate)));
try
{
    secureClient.Stream.AuthenticateAsServer(certificate, true,
SslProtocols.Default, false);
}

catch (Exception ex)
{
    Log.WriteLine("Authentication failed " + ex.Message);
    secureClient.Disconnect();
}

```

**Figure 21 Implementing TLS on the Server**

### 4.3 Certificate/ Key Management

The implementation of authentication and secure channel with TLS makes use of cryptographic keys and digital certificates. A successful operation of these services requires that the keys and certificates are managed effectively and in a secure manner. A Certificate Authority is usually responsible for creating, managing and revoking certificates. After a certificate is created it needs to be distributed in a manner that is secure to avoid it falling in wrong hands. Using Active Directory Certificate Services (ADCS), which integrates with Active Directory Domain Services,

certificates distribution is easily automated. A group policy allows all legitimate users to receive certificates once they login to the domain.

After certificates are created and distributed by the CA, they will need to be managed during their lifetime. Two mechanisms for managing certificates include certificate expirations and certificate revocation lists (CRL). A certificate is valid for a certain period of time. Any attempt to use the certificate before or after the designated period will not be possible. CRLs are a way to notify users about certificates that are no longer valid. The revocation of certificate validity could be due to a compromise of the certificate or that the certificate is no longer needed such as certificates belonging to a former employee. The methods listed in Figure 22 check for certificate errors including expiration and revocation status for client and server certificates respectively.

```
public static bool ValidateClientCertificate(object sender, X509Certificate
    certificate, X509Chain chain, SslPolicyErrors sslPolicyErrors)
{
    if (sslPolicyErrors == SslPolicyErrors.None)
        return true;

    Log.WriteLine("Certificate error: " + sslPolicyErrors.ToString());

    return false;
}

public static bool ValidateServerCertificate(object sender, X509Certificate
    certificate, X509Chain chain, SslPolicyErrors sslPolicyErrors)
{
    if (sslPolicyErrors == SslPolicyErrors.None)
        return true;

    Log.WriteLine("Certificate error: " + sslPolicyErrors.ToString());

    return false;
}
```

**Figure 22 Methods to Check Certificate Validity for Client and Server Certificates**

## 4.4 Testing and Results

The NX Connect security framework has been tested on a prototype platform developed to mimic an environment where a multi-user CAD application could be used (see Figure 23). By developing this test platform the various objectives of the security framework could be assessed thoroughly. The following sections describe the components of the test prototype:

### 4.4.1 Test Prototype

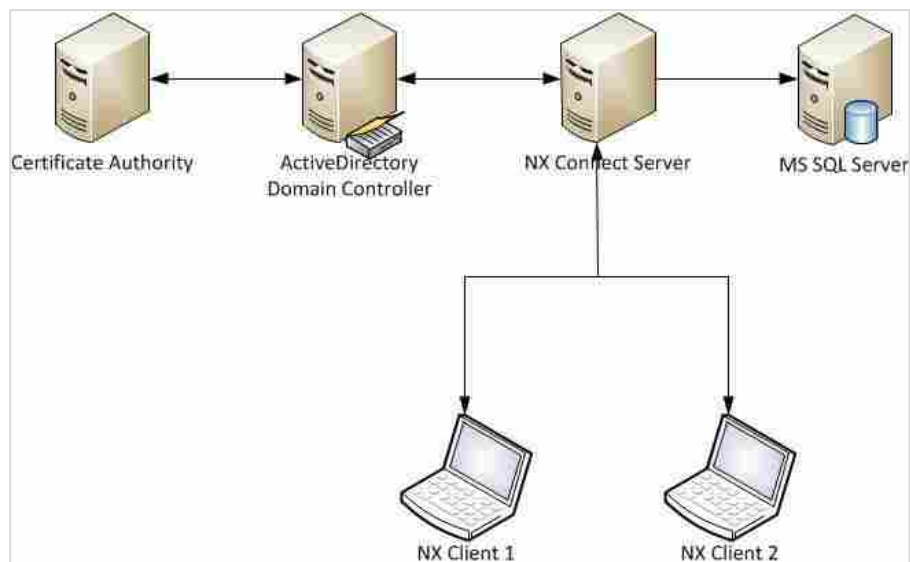


Figure 23 Test Prototype Setup

### VMWare ESXi Server

The ESXi server provided a platform to host the various workstations and servers in the form of virtual machines. In all five virtual machines were used for the testing platform including two client workstations with Siemens NX 8.0, an Active Directory Domain Controller, a Certificate Authority, and an MS SQL server.

## **Client Workstations**

Two client workstations running Windows 7 professional were created for the testing platform. Both were loaded with Siemens NX Unigraphics 8.0, the client software in the NX Connect system. By using two workstations the multi-user and collaborative features of the system could be observed.

## **Active Directory Domain Controller**

Since the framework required the use of a directory service for the implementation of authentication and authorization, an Active Directory Domain Controller was set up to provide that service. Active Directory is widely used in enterprise networks and thus its use in the security framework meant a wide range of target environments.

## **Certificate Authority**

A certificate authority was set up using Active Directory Certificate Services for the purposes of creating, distributing and managing the certificates which will be used with the TLS protocol.

## **SQL Database Server**

The Information Storage Module component of NX-Connect requires a database for the storage of geometric parts and security data. For this purpose a MS SQL database server was set up.

#### **4.4.2 Results**

This section discusses the results of testing the various features of the security framework and how they fall in line with the objectives of this research. As stated earlier the objectives of this research can be summarized as follows:

- The implementation of the security framework will result in data confidentiality during transmission through the use of encryption techniques.
- The implementation of the security framework will achieve authentication and authorization/access control through the integration of NX Connect with a directory service. The purpose of authentication is to confirm the identity of a user trying to access the system and the purpose of authorization is to ensure that an authenticated user only has access to the resources and actions in the system which he is permitted.
- The implementation of the security framework will not result in substantial degradation in the performance of the application.

#### **Secure Communications**

The original implementation of NX Connect has very minimal security, if any, in place to ensure confidentiality of information when clients communicates with the server. This means all message exchanges were carried out in plain text. With the integration of this security framework with NX Connect and making use of TLS communications are encrypted and kept from unauthorized access during transmission. To test that this was the case a network capture tool, Wireshark, was used to monitor traffic between a client and a server in both the original version of NX Connect and the security enhanced NX Connect. Figure 24 and Figure 25 shows samples of network traffic captured when using no encryption and when using encryption. In the

unencrypted version it can be clearly seen that authentication is taking place, several parameters are in clear text and the username and password are clearly visible. In the encrypted version, such is not the case. The data that is seen in clear text is certificate information which cannot be sent encrypted. The reason is the certificate contains a public key which is needed for encryption. Encrypting the certificate will make this key inaccessible.

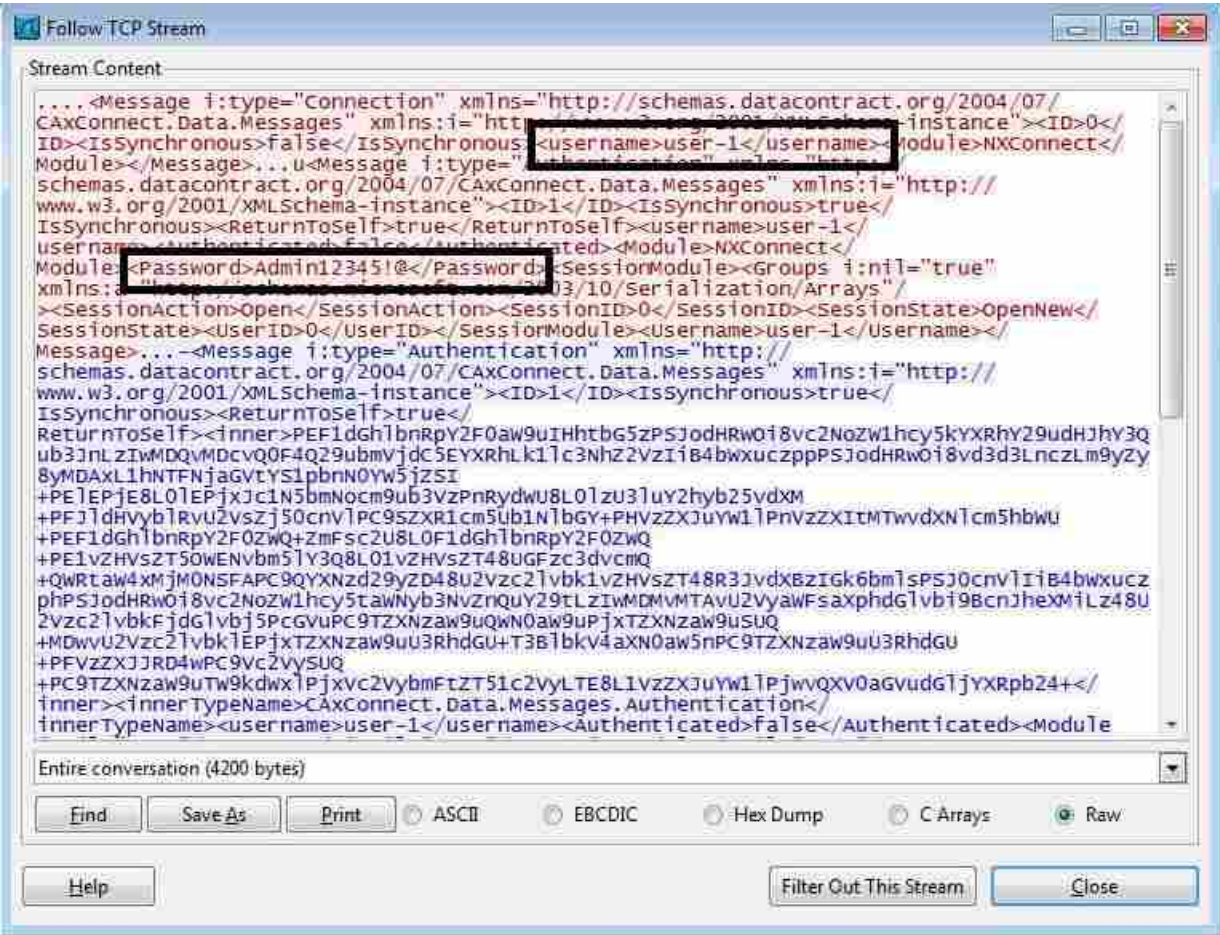


Figure 24 Network Traffic When Using No Encryption

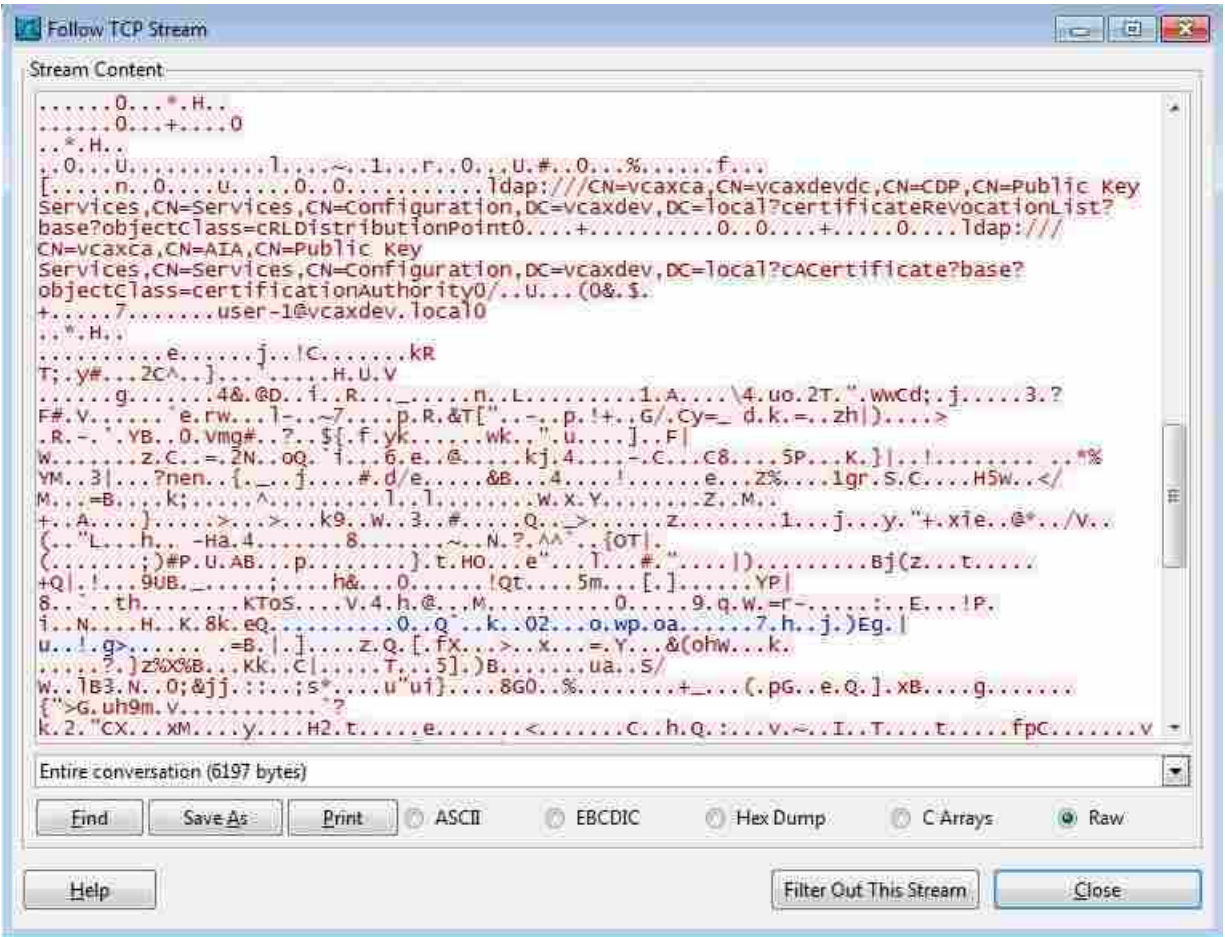


Figure 25 Network Traffic When Using TLS for Encryption

## Authentication

The authentication framework has been designed to accomplish authentication through the use of two factors:

- Something that the user has – digital certificates
- Something that the user knows – username/password

These two factors must all be present and in valid form in order for the authentication process to be successful. The absence or invalidity of any of the factors should result in authentication failure.

Also because the authentication framework integrates with Active Directory, it is possible to make

use of all available user account management controls in Active Directory. An example of a control that is available in Active Directory is account lockout. The purpose of this feature is to mitigate the effects of brute-force attacks. Care should be taken, however, not to set very low lockout thresholds as this can also result in a denial of service attack.

In this section the results of various test cases representing usage of the authentication framework are shown. As mentioned earlier, authentication should only be successful when the required factors are both present and in valid form. Table 1 lists tests that were carried out on the authentication framework and the results.

**Table 1 Authentication Tests and Results**

	<b>Test Action</b>	<b>Data</b>	<b>Expected Results</b>	<b>Status</b>
1.	Login with valid username and password, and certificate	Username: User-1 Password: Admin12345!@ Certificate path: Certificates\Current User\Personal\Certificates\user-1	NX-Connect loads parts	Success
2.	Login with valid username and password, and no certificate	Username: User-1 Password: Admin12345!@	Connection is not established. Error message is displayed	Success
3.	Login with no username, no password, and valid certificate	Username: Password: Certificate path: Certificates-Current User\Personal\Certificates\user-1	User is prompted to supply valid username	Success
4.	Login with valid username, no password, and valid certificate	Username: User-1 Password: Certificate path: Certificates\Current User\Personal\Certificates\user-1	User message: 'User could not be authenticated' is displayed	Success
5.	Login with valid username, invalid password, and valid certificate	Username: User-1 Password: 123 Certificate path: Certificates\Current User\Personal\Certificates\user-1	User message: 'User could not be authenticated' is displayed	Success
6.	Login with valid username, valid password, and revoked certificate	Username: User-1 Password: Admin12345!@ Certificate path: Certificates\Current User\Personal\Certificates\user-1	Connection is not established. Error message is displayed	Success
7.	Set account lockout threshold to 3 attempts. Login with valid username, invalid password three time, and then login with valid username, valid password. Valid certificates always used	Username: User-1 Password: 123 (invalid) Username: User-1 Password: Admin12345!@ (valid) Certificate path: Certificates\Current User\Personal\Certificates\user-1	User message: 'User could not be authenticated' is displayed	Success



## **Authorization**

A hypothetical organization is used as a test case to demonstrate the operation of authorization framework. This organization uses a hybrid access control model making use of RBAC and MAC, thus taking advantage of the administration flexibility of RBAC for managing users as roles and data confidentiality of MAC for engineering parts. Users and engineering parts may be assigned a security class of Confidential-1 (C1), Confidential-2 (C2) or Not Confidential (NC). The security classes determine whether a user will have access to a geometric part or not. In addition project based need-to-know categories further restrict access to geometric parts to users who are assigned to a particular project. Users are assigned to roles, which in turn determine operations that a user can perform on a part. The operation of this hybrid access control model is illustrated in Figure 5.

As an example consider the CAD model called ‘SimpleAssembly.prt’ which consists three parts: Crankshaft.prt, PistonRod.prt and Piston.prt. The security labels for these parts are shown in figure 11. The security labels and roles for two users – User-1 and User-2 – are shown in Figure 27. Figure 28 shows the role and security class hierarchies.

### **Test Case 1**

According to the security configuration described above ‘User-1’ which has a security class of ‘Confidential-2’ will have access to the entire assembly while ‘User-2’ which has a security class of ‘Confidential-1’ will only have access to Crankshaft.prt. Once granted access to the geometric model, User-1’s’ role (Senior Engineer) determine what actions can be carried out on the parts that make up the assembly, which in this case include modification, deleting and creating a new part (observe that Senior Engineer inherits permissions from Junior Engineer and

Technician). User-2, on the other hand, which has the role of Junior Engineer can only modify and create a new part. Figure 29 shows how for the given CAD model, User-1 and User-2, with different security clearances, different views have been rendered to them.

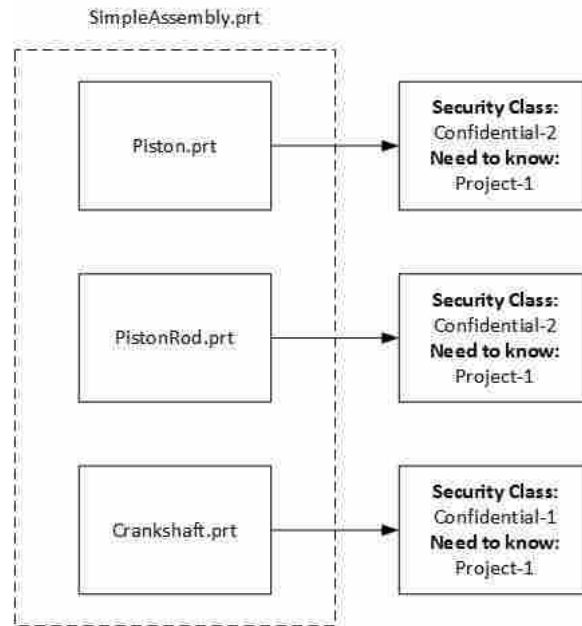


Figure 26 Security Labels for SimpleAssembly.prt

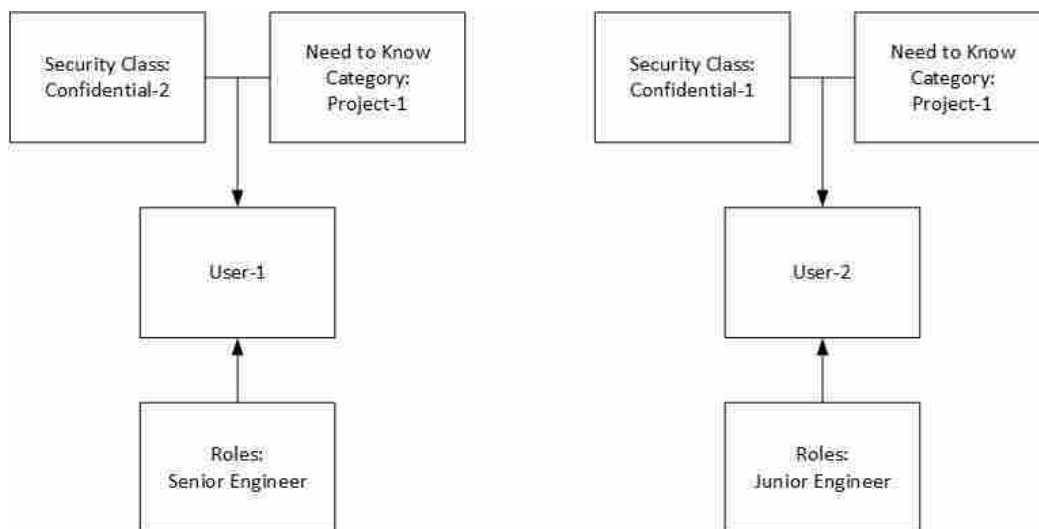


Figure 27 Security Labels and Roles for User-1 and User-2

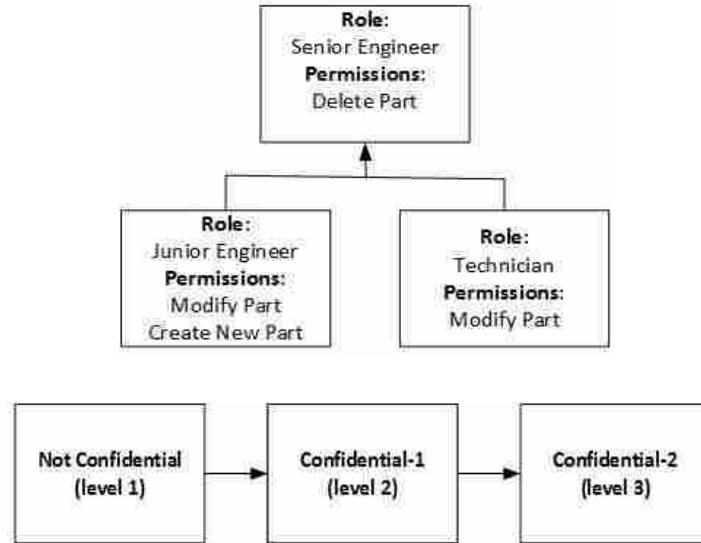


Figure 28 Role and Security Class Hierarchies

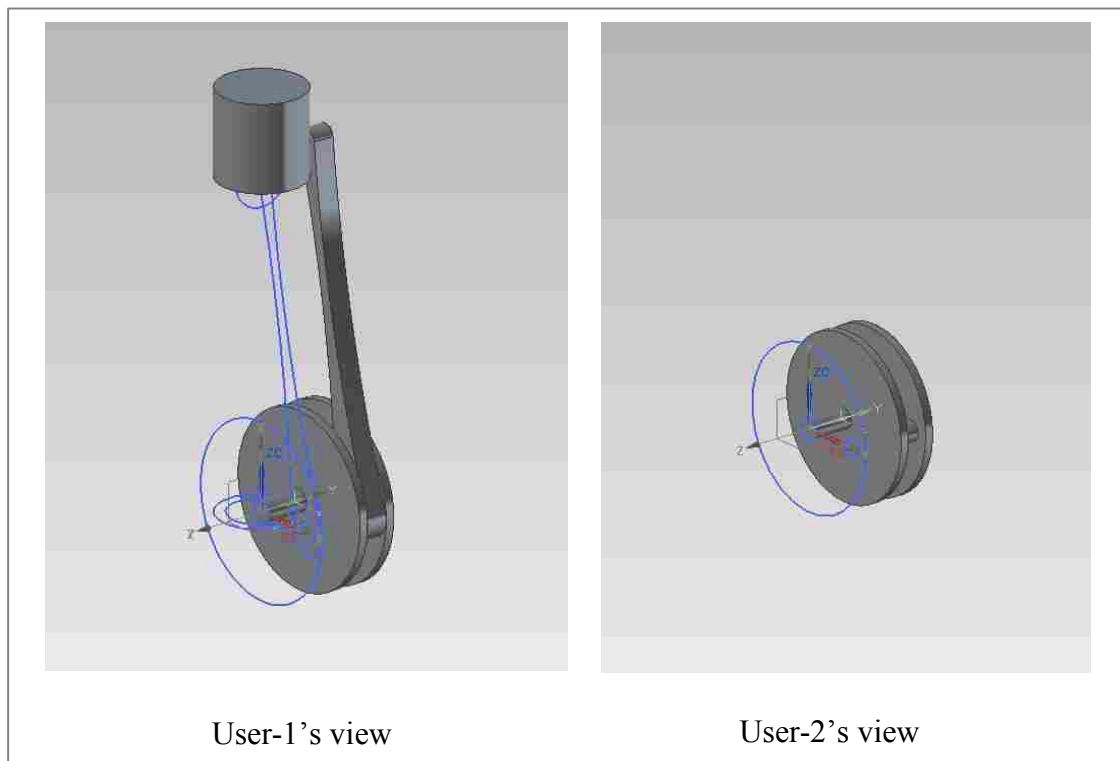
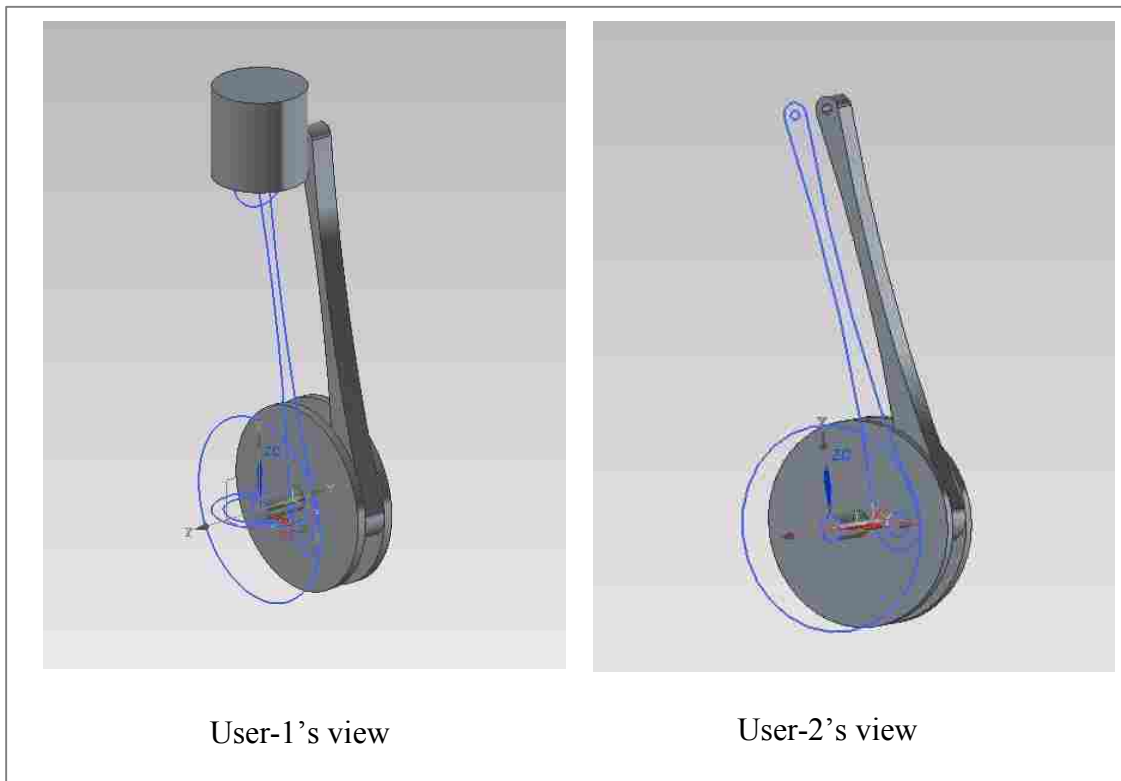


Figure 29 User Views for SimpleAssembly.prt

## Test Case 2

Granting User-2 view access to PistonRod.prt changes its view to include the piston rod as shown in Figure 30. It should be observed that User-2 can only view the piston rod but cannot perform any operations. Any attempt to do so would be denied. Figure 31 shows an error message displayed to User-2 while attempting to modify PistonRod.prt.

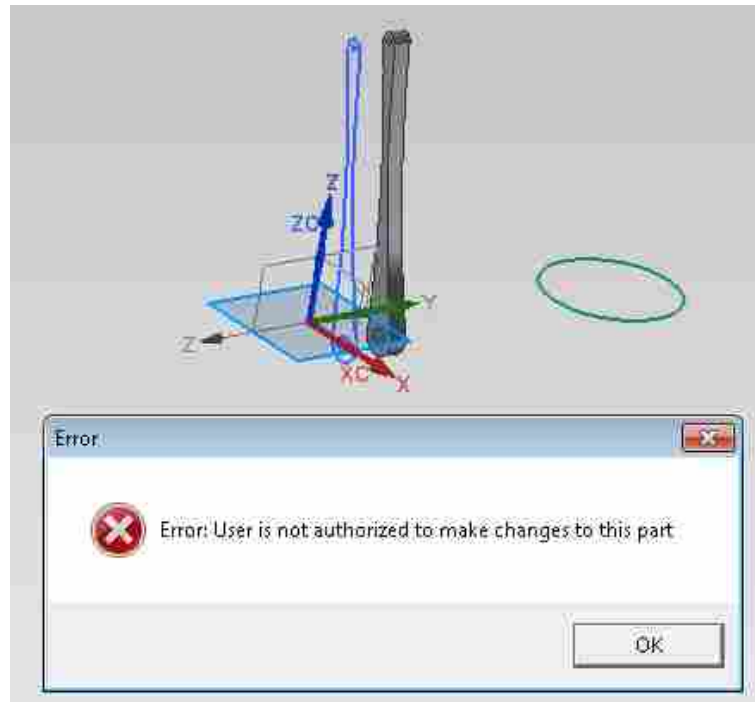


**Figure 30 User Views After Granting User-2 View Access for PistonRod.prt**

## Test Case 3

The hierarchical nature of RBAC roles means that when a role A is made a member of a role B, role A inherits all the permissions of role B. Using the security setup described above as an example, Senior Engineer has only one permission explicitly assigned to it. But because it is

also a member of the Junior Engineer and Technician roles, it inherits their permissions as well. This result is shown in Figure 32.

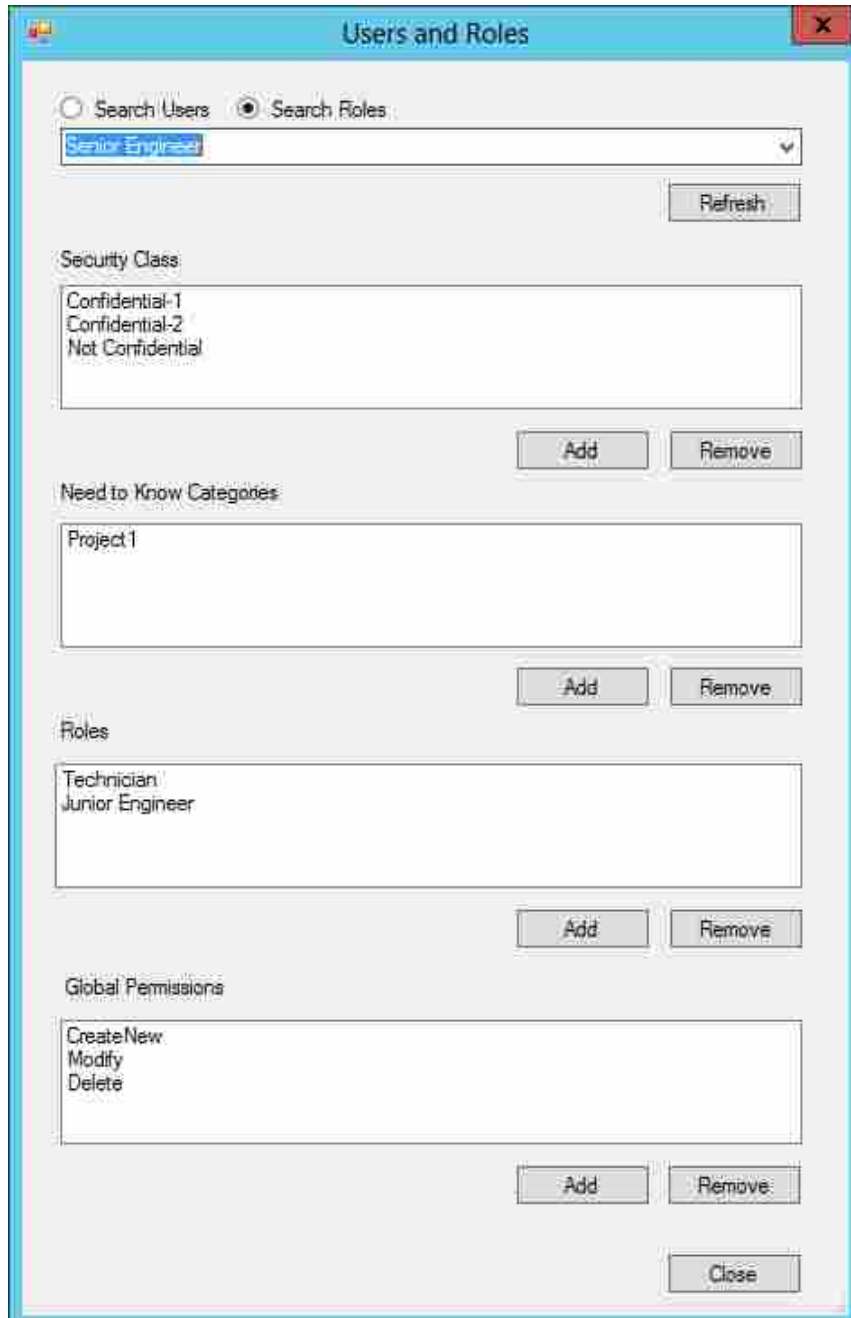


**Figure 31 User-2 Not Allowed to Make Modification to PistonRod.prt**

#### **Test Case 4**

Global permissions allow a user to perform operations regardless of the part that is loaded. Object permissions only allow operations on a specific part. To test this feature, User-1 has been granted global permissions CreateNew, Modify and Delete through its association with the Senior Engineer role. User-2, on the other hand has object permissions Modify and View for only Crankshaft.prt. To test both User-1 and User-2 opened the three parts Piston.prt, PistonRod.prt and Crankshaft.prt and attempted to modify the parts. As expected User-1 was successful on all three occasions. User-2, however, was only successful in modifying

Crankshaft.prt. These results were expected and thus proved the correct functioning of the framework.



**Figure 32 Security Configuration for Senior Engineer Role**

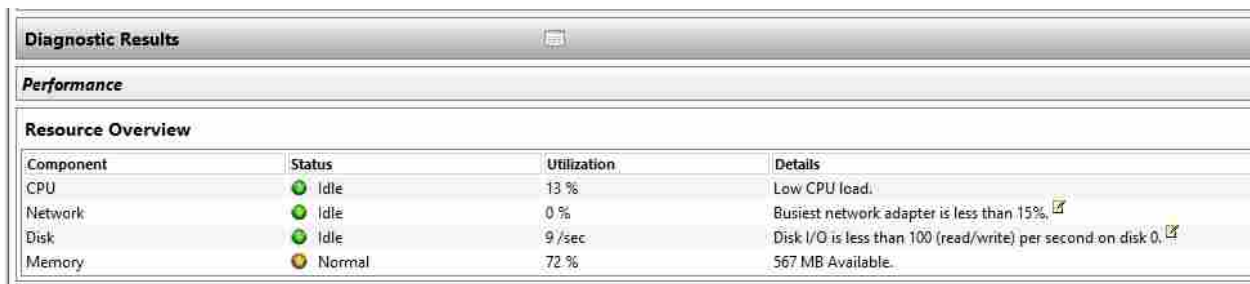
## Performance

The effect of the framework on the performance of NX Connect was monitored using Windows Performance Monitor (Perfmon). The monitoring was done for two situations:

- When the framework had not been applied
- When the framework had been applied

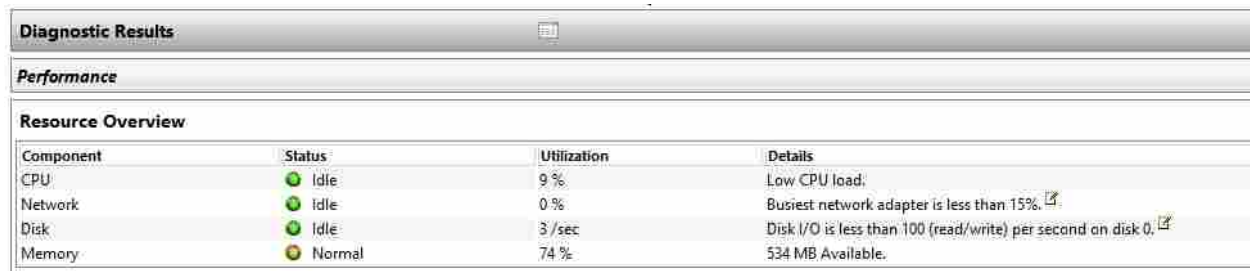
Similar operations were carried out in both situations and the respective reports were generated.

The metrics observed included CPU usage and Memory utilization. The reports, as shown in Figure 33 and Figure 34, indicated that there was no substantial decrease in the performance of NX Connect after the framework had been applied.



Component	Status	Utilization	Details
CPU	Idle	13 %	Low CPU load.
Network	Idle	0 %	Busiest network adapter is less than 15%. <a href="#">[?]</a>
Disk	Idle	9 /sec	Disk I/O is less than 100 (read/write) per second on disk 0. <a href="#">[?]</a>
Memory	Normal	72 %	567 MB Available.

Figure 33 Diagnostic Results for NX Connect with Security Framework



Component	Status	Utilization	Details
CPU	Idle	9 %	Low CPU load.
Network	Idle	0 %	Busiest network adapter is less than 15%. <a href="#">[?]</a>
Disk	Idle	3 /sec	Disk I/O is less than 100 (read/write) per second on disk 0. <a href="#">[?]</a>
Memory	Normal	74 %	534 MB Available.

Figure 34 Diagnostic Results for NX Connect without Security Framework

## **5 CONCLUSION AND FUTURE WORK**

### **5.1 Conclusion**

This research started with the question: Can the v-CAX project be successfully deployed in an enterprise environment while meeting the security requirements of secure communications, authentication, and authorization? The results obtained from the test prototype demonstrates that this is possible. Work done in this research does not comprehensively handle all security requirements of a multiuser CAD, but a foundation has been laid for future enhancements to be carried out.

This research carried out work on proposing a security framework for NX Connect with the following objectives:

- The implementation of the security framework will result in data confidentiality during transmission through the use of encryption techniques.
- The implementation of the security framework will achieve authentication and authorization/access control through the integration of NX Connect with a directory service.
- The implementation of the security framework will not result in any appreciable degradation in the performance of the application.

By integrating NX Connect with Microsoft Active Directory, authentication is achieved with a centralized directory service rather than relying on a direct connection to a database. The



integration with Active Directory also enables the implementation of an Access Control mechanism thus ensuring that users only could perform actions that their permissions allow. By implementing the TLS protocol together with a PKI, both of which are industry standards, the security framework encrypts all communications between the clients and server. By comparing the response time of the original NX Connect and the security enhanced NX Connect, no substantial degradation of performance was observed.

The efficacy of the security framework has been tested by developing a test prototype mimicking an enterprise environment. Various test cases have been used on the test prototype to ensure that the framework is capable of handling as many situations as possible.

## **5.2 Future Work**

Some limitations were identified in the introductory chapter. Based on these limitations and others that may be identified during a review of this research, there are a wide range of possible enhancements that could be applied to the current version of the framework. For example, this research only focused on NX Connect prototype in the v-CAX project. The findings in the research can be used to develop similar security frameworks for the other multiuser CAD prototypes. While the focus in this research has been on collaborative CAD software, the design principles could also be applied to other CAX software including Computer Aided Manufacturing (CAM) and Product Development Management (PDM) seeking to transition to a multiuser collaborative paradigm. In addition, recognizing that not all deployment environments are entirely Microsoft, further research will be needed to give the framework a more generic nature to suit multiple target environments. The use of smart cards to store user certificates could also be implemented to enhance security.

The use of polygon reduction techniques (Cignoni, Montani, and Scopigno 1998) to create a variable level of resolution depending on the security levels of user could be investigated. This

will greatly enhance the authorization model as it will provide more control than the all-or-nothing approach used in this research.

This research, together with other on-going research work clearly confirm that multiuser collaborative CAD looks good going into the future, and with the implementation of a security framework, its acceptance in industry is even more likely.

## REFERENCES

- Bidarra, R., E. van den Berg, and W. F. Bronsvort. 2002. "A Collaborative Feature Modeling System." *Journal of Computing and Information Science in Engineering* 2 (3): 192. doi:10.1115/1.1521435.
- Bouras, C., E. Giannaka, and T. Tsiatsos. 2009. "E-Collaboration Concepts , Systems and Applications E-Collaboration Concepts , Systems and Applications." *Information Science Reference, E-Collaboration: Concepts, Methodologies, Tools and Applications* 1.
- Brandon, E. 2008. "Keeping Older Workers on the Job." *Usnews.com*.  
<http://money.usnews.com/money/blogs/planning-to-retire/2008/07/17/keeping-older-workers-on-the-job>.
- Cera, C. D., and W. C. Regli. 2004. "Hierarchical Role-Based Viewing for Multi-Level Information Security in Collaborative CAD Department of Computer Science", no. January.
- Cera, C.D., W.C. Regli, I. Braude, Y. Shapirstein, and C.V. Foster. 2002. "A Collaborative 3D Environment for Authoring Design Semantics." *IEEE Computer Graphics and Applications* 22 (3): 43–55. doi:10.1109/MCG.2002.999787.
- Chen, L., Z. Song, and L. Feng. 2004. "Internet-Enabled Real-Time Collaborative Assembly Modeling via an E-Assembly System: Status and Promise." *Computer-Aided Design* 36 (9): 835–47. doi:10.1016/j.cad.2003.09.010.
- Cignoni, P., C. Montani, and R. Scopigno. 1998. "A Comparison of Mesh Simplification Algorithms." *Computers & Graphics* 22 (1): 37–54. doi:10.1016/S0097-8493(97)00082-4.
- Fan, L.Q., A. Senthil Kumar, B.N. Jagdish, and S.H. Bok. 2008. "Development of a Distributed Collaborative Design Framework within Peer-to-Peer Environment." *Computer-Aided Design* 40 (9): 891–904. doi:10.1016/j.cad.2008.05.006.
- Friedl, S. 2008. "An Illustrated Guide to the Kaminsky DNS Vulnerability."  
<http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html>.
- Gaetgens, F. 2014. "Technology Overview for Active Directory IAM Tools."  
<http://www.gartner.com/document/2845421?ref=unauthreader&srcId=1-3478922225>.
- Harris, S. 2012. *ALL IN ONE CISSP*. 6th ed.

- Kim, S., D. Kim, L. Lu, S. Park, and S. Kim. 2011. "A Feature-Based Modeling Approach for Building Hybrid Access Control Systems." *2011 Fifth International Conference on Secure Software Integration and Reliability Improvement*, June. Ieee, 88–97. doi:10.1109/SSIRI.2011.16.
- Li, W.D., W.F. Lu, J.Y.H. Fuh, and Y.S. Wong. 2005. "Collaborative Computer-Aided Design—research and Development Status." *Computer-Aided Design* 37 (9): 931–40. doi:10.1016/j.cad.2004.09.020.
- Microsoft MSDN. "System.Net.Security Namespace." [http://msdn.microsoft.com/en-us/library/system.net.security\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/system.net.security(v=vs.110).aspx).
- Microsoft Technet. 2011. "Active Directory Collection." [http://technet.microsoft.com/en-us/library/cc780036\(WS.10\).aspx#w2k3tr\\_ad\\_over\\_qbjd](http://technet.microsoft.com/en-us/library/cc780036(WS.10).aspx#w2k3tr_ad_over_qbjd).
- Moncur, R. A., C. G. Jensen, CC. Teng, and E Red. 2013. "Data Consistency and Conflict Avoidance in a Multi-User CAx Environment." *Computer-Aided Design and Applications* 10 (5): 727–44. doi:10.3722/cadaps.2013.727-744.
- Naughton, K. 2013. "Styling With Digital Clay - Newsweek and The Daily Beast." *The Daily Beast*. <http://www.thedailybeast.com/newsweek/2003/04/27/styling-with-digital-clay.html>.
- Qiang, L., Y. F. Zhang, and a. Y. C. Nee. 2001. "A Distributive and Collaborative Concurrent Product Design System through the WWW/Internet." *The International Journal of Advanced Manufacturing Technology* 17 (5): 315–22. doi:10.1007/s001700170165.
- Ramani, K, A. Agrawal, M. Babu, and C. Hoffman. 2003. "CADDAC: Multi-Client Collaborative Shape Design System with Server-Based Geometry Kernel." *Journal of Computing and Information Science in Engineering(Transactions of the ASME)* 3.2.
- Red, E., D. French, G. Jensen, S. S. Walker, and P. Madsen. 2013. "Emerging Design Methods and Tools in Collaborative Product Development." *Journal of Computing and Information Science in Engineering* 13 (3): 031001. doi:10.1115/1.4023917.
- Red, E., V. Holyoak, C. G. Jensen, F. Marshall, J. Ryskamp, and Y. Xu. 2010. "N-CAx : A Research Agenda for Collaborative Computer-Aided Applications." *Computer-Aided Design and Applications* 7: 387–404. doi:10.3722/cadaps.2010.xxx-yyy.
- Rouibah, K., and S. Ould-Ali. 2007. "Dynamic Data Sharing and Security in a Collaborative Product Definition Management System." *Robotics and Computer-Integrated Manufacturing* 23 (2): 217–33. doi:10.1016/j.rcim.2006.02.011.
- Shen, W., Q. Hao, and W. Li. 2008. "Computer Supported Collaborative Design: Retrospective and Perspective." *Computers in Industry* 59 (9): 855–62. doi:10.1016/j.compind.2008.07.001.

- Siemens. 2013. "Explore NX and Discover Your Solution."  
[http://www.plm.automation.siemens.com/en\\_us/products/nx/](http://www.plm.automation.siemens.com/en_us/products/nx/).
- Winn, J. D. 2012. "Intergration of Massive Multiplayer Online Role-Playing Games Client-Server Architectures with Collaborative Multi-User Engineering CAx Tools". Brigham Young University.
- Zhang, D. Y., L. Wang, and Y. Zeng. 2008. "Secure Collaborative Product Development : A Literature Review." In *International Conference on Product Life Cycle Management*, 331–40.

## APPENDICES

## APPENDIX A

### Implementation of Secure Communications (TLS) in code

In order to establish secure communications between NX Connect clients and server, TLS needed to be programmed on both the client and the server. The .NET framework contains all the necessary classes and methods to make this possible. The following code shows how TLS was implemented on the client (Note: All code can be found in the accompanied files).

The TLSClientConnection object encapsulates properties and methods needed to create and maintain the TLS connection. Part of the TLSClientConnection class definition is shown in the following code listing.

```
class TLSClientConnection : ClientConnection
{
    public SslStream Stream
    {
        get;
        set;
    }

    public TLSClientConnection(SslStream stream, Socket socket)
    {
        Stream = stream;
        Stream.ReadTimeout = 10000;
        Stream.WriteTimeout = 10000;
        Socket = socket;
        IsConnected = false;
    }

    public void Authenticate(string server, X509Certificate2Collection cCollection)
    {
        Stream.AuthenticateAsClient(server, cCollection, SslProtocols.Default, false);
        Stream.ReadTimeout = -1;
        Stream.WriteTimeout = -1;
        if (Socket != null)
            IsConnected = Stream.IsMutuallyAuthenticated;
    }
}
```

The TLS connection is initiated from the 'ServerConnection' class. First, SslStream and Socket objects are created. The SslStream object handles the encryption of network traffic while the Socket objects handles the underlying TCP connection. These two objects are used to instantiate a new TLSClientConnection object.

```
    TLSClientConnection clientConnect = null;
    if (UseTLS == true)
    {
        clientConnection = new TLSClientConnection(new SslStream(new NetworkStream(connection),
            false, new RemoteCertificateValidationCallback(ValidateServerCertificate)),
            connection); //connection is a socket object
        if (clientConnection is TLSClientConnection)
        {
            clientConnect = clientConnection as TLSClientConnection;
            clientConnect.Authenticate("vcaxdevdc.vcaxdev.local", GetClientCertificates());
        }
    }
}
```

The ValidateServerCertificate method checks the server certificate for errors. Potential errors include certificate name mismatch, certificate not available and certificate chain errors. If any of these errors are found the TLS handshake is terminated and no connection is established. GetClientCertificates methods retrieves client certificates from the local computer certificate store.

```
public static bool ValidateServerCertificate(object sender, X509Certificate certificate,
    X509Chain chain, SslPolicyErrors sslPolicyErrors)
{
    if (sslPolicyErrors == SslPolicyErrors.None)
        return true;

    return false;
}

private X509Certificate2Collection GetClientCertificates()
{
    X509Store store = new X509Store(StoreName.My, StoreLocation.CurrentUser);
    store.Open(OpenFlags.ReadWrite);
    X509Certificate2Collection certs = store.Certificates.Find(X509FindType.FindByIssuerName,
        "vcaxca", false);
    return certs;
}
```



The Authenticate method of TLSClientConnection takes two arguments: the name of the server to which the client is connecting, and a certificate collection object which would contain the certificate which the client submits to the server for authentication. Implementing TLS on the server is similar. The following code snippets show how:

```
SslClient secureClient = new SslClient(new SslStream(new NetworkStream(client), false,
    new RemoteCertificateValidationCallback(ValidateClientCertificate)));

try
{
    secureClient.Stream.AuthenticateAsServer(certificate, true, SslProtocols.Default,
        true);
}

catch (Exception ex)
{
    log.WriteLine("Authentication failed " + ex.Message);
    secureClient.Disconnect();
}

c = secureClient;
```

```
public static bool ValidateClientCertificate(object sender, X509Certificate certificate,
    X509Chain chain, SslPolicyErrors sslPolicyErrors)
{
    if (sslPolicyErrors == SslPolicyErrors.None)
        return true;

    log.WriteLine("Certificate error: " + sslPolicyErrors.ToString());

    return false;
}
```

## APPENDIX B

### Implementing Authentication in NX Connect with Active Directory

The .NET framework provides the `System.DirectoryServices.AccountManagement` which provides methods for interacting with Active Directory from code. To authenticate a user in Active Directory, the `ValidateCredentials` method of the `PrincipalContext` object is used, as shown below.

```
public static bool AuthenticateAD(string user, string pass, SessionModule smodule)
{
    PrincipalContext pContext = new PrincipalContext(ContextType.Domain,
        NXSettings.Domain, "cn=users,dc=vcaxdev,dc=local", NXSettings.Username,
        NXSettings.Password);
    bool AccountLockedOut =
        UserPrincipal.FindByIdentity(pContext, user).IsAccountLockedOut();
    bool ADAuthenticated =
        pContext.ValidateCredentials(user, pass, ContextOptions.Negotiate);
    if (ADAuthenticated && !AccountLockedOut)
    {
        NXConnectDB db = new NXConnectDB(Properties.Settings.Default.NXConnectDB);
        User dbUser = (from u in db.Users
            where u.Username == user
            select u).SingleOrDefault();

        if (dbUser == null)
        {
            dbUser = AddUser(user);
            Log.WriteLine("Adding new user to database");
        }

        Session session =
            dbUser.Sessions.Where(x => x.EndTime == null).FirstOrDefault();
    }
}
```

The authentication process is started when the client creates an Authentication object. This object encapsulates all authentication data for the user, including username and password. The object is serialized and sent synchronously to the server. The method responsible for this process is 'SendAuthenticationRequest' in the 'NXConnectAuthorizationModule' class.

```
public bool SendAuthenticationRequest(string module)
{
    Authentication response = Authenticate(module, SessionAction.Open);

    if (response.SessionModule.SessionState == SessionState.OpenNew)
    {
        return response.Authenticated;
    }
    else if (response.SessionModule.SessionState == SessionState.OpenExisting)
    {
        response = Authenticate(module, SessionAction.ForceOpen);
        if (response.SessionModule.SessionState == SessionState.OpenNew)
        {
            return response.Authenticated;
        }
    }

    return response.Authenticated;
}
```

Because the network traffic is encrypted with TLS, the authentication data is not exposed as clear text. On receiving the Authentication object, the server's authentication handler extracts all user data and authenticates with Active Directory. The results are encapsulated in another Authentication object and sent back to the client. Access is only granted when authentication is successful.

```
private static void HandleAuthentication(Client sender, Message request)
{
    Authentication req = request as Authentication;
    if (req.SessionModule.SessionAction == SessionAction.Open ||
        req.SessionModule.SessionAction == SessionAction.ForceOpen)
    {
        req.Authenticated = Server.AuthenticateAD(req.Username, req.Password, req.SessionModule);
        Sender.Send(sender, new Authentication(req));
    }
    else if (req.SessionModule.SessionAction == SessionAction.Close)
    {
        Server.CloseSession(req.SessionModule.SessionID, req.SessionModule);
        Sender.Send(sender, new Authentication(req));
    }
}
```

## APPENDIX C

### Implementing Access Control (Authorization) in NX Connect

To apply access control capability to NX Connect, some method calls are needed. First of all the methods that carry out the operations which need access control must be identified. For this research the methods which perform the following operations were identified:

- Loading Parts and Assemblies in the Parts Listbox.
- Loading Parts that constitute Assemblies
- Creating a new Part
- Deleting an existing Part
- Modifying an existing Part

The table below lists the names of the methods that perform these functions as well as the files in which they are found.

#### Methods for Access Control Code Insertion

Method Name	Description	Filename
resetPartsSync	Loads Parts that constitute an assembly	NXConnectDataSyncModule.cs
populatePartList	Loads Parts and Assemblies in the Parts Listbox	MasterForm.cs
btnNew_Click	Creates a new Part	MasterForm.cs
btnDelete_Click	Deletes an existing Part	MasterForm.cs
PushFeatures	Modifies and existing Part	NXConnectController.cs

In each of these methods an authorization request code has been inserted. It is only when the response of the request is positive that the method continues with its original functionality. If not

the method raises an exception and returns. The following code listing shows part of the `resetPartSync` method. From this code it can be noticed a call is made to the method `SendAuthorizationRequest`. The method arguments passed include the username of the logged in user and a list of part IDs of the assembly components. The rest are the request type which in this case is `AuthRequestType.Batch`, the reason being that an authorization request is being sought for a batch of parts. Other values for this parameter are:

- `AuthRequestType.New` – used when a new part is being created
- `AuthRequestType.Existing` – used when authorization information is requested for a single existing part
- `AuthRequestType.GlobalPermission` – used when authorization information is requested for a global permission

The next argument specifies whether the request is requesting for security clearance information, authorization information for an object permission, or authorization information for a global permission. The last argument specifies the name of the operation for which authorization information is being sought. This could be 'Modify', 'Delete', 'View' or any other operation that has been defined.

```

List<Authorization> partsAuthorization =
    NXConnectApp.Authorization.SendAuthorizationRequest(NXConnectApp.Authorization.Username,
        partsInAssembly.ToArray(), AuthRequestType.Batch, AuthRequest.CheckSecurityClearance, "");

if (partsAuthorization == null)
{
    return;
}

List<AssemblyComponent> filteredComponents = new List<AssemblyComponent>();

foreach (AssemblyComponent component in assemblyComponents)
{
    Boolean authorized = (from part in partsAuthorization
        where part.PartID == component.ComponentPartID
        select part.Authorized).SingleOrDefault();

    if (authorized)
        filteredComponents.Add(component);
}

```

The code for PushFeatures shown next. Once again authorization request code has been inserted. The authorization code is very similar to that of resetPartSync, however, with a few differences. It would be noticed that two authorization requests are made. This is because both object level and global level authorization information are being checked. This is not necessary when the authorization request is to check security clearance. In addition there is a check for explicit denial. If an operation is explicitly denied the operation cannot be performed.

```

Authorization partAuthorization =
    NXConnectApp.Authorization.SendAuthorizationRequest(NXConnectApp.Authorization.Username,
        partname + ".prt", AuthRequestType.ExistingPart,
        AuthRequest.CheckSubjectPartPermission, "Modify");

if (partAuthorization.ExplicitDenied == true)
{
    throw new Exception("User is not authorized to make changes to this part");
}
else if (partAuthorization.Authorized == false)
{
    partAuthorization =
        NXConnectApp.Authorization.SendAuthorizationRequest(NXConnectApp.Authorization.Username,
            AuthRequestType.General, AuthRequest.CheckSubjectGlobalPermission, "Modify");
    if (partAuthorization.Authorized == false)
        throw new Exception("User is not authorized to make changes to this part");
}
}

```

## APPENDIX D

### Administering Security Objects in NX Connect

A security administration console was designed for the administration of security objects.

Using this tool the following operations can be carried out:

- Create new security objects – security class, category, role, global permission
- Manage users with respect to their roles and permissions
- Assign object level permissions to Parts and specify which users or roles are permitted

Locate the administrative console in the NXAccessControlConsole folder. When it is first run the dashboard shown below is displayed. Click on the appropriate action link to perform a desired operation.



**Access Control Administration Dashboard**



## Managing Users and Roles

From the Users and Roles console, users can be assigned security classes, need-to-know categories, roles and global permissions. To assign security objects to User-1, for example, lookup User-1 from the dropdown box. Then under each of the security object sections, click on the 'Add' button, and then select the desired object to assign to the user. To remove, click on the security object to highlight it and click the 'Remove' button.

The screenshot shows a window titled "Users and Roles" with a search bar containing "User-1" and a "Refresh" button. Below are four sections, each with a list of items and "Add" and "Remove" buttons:

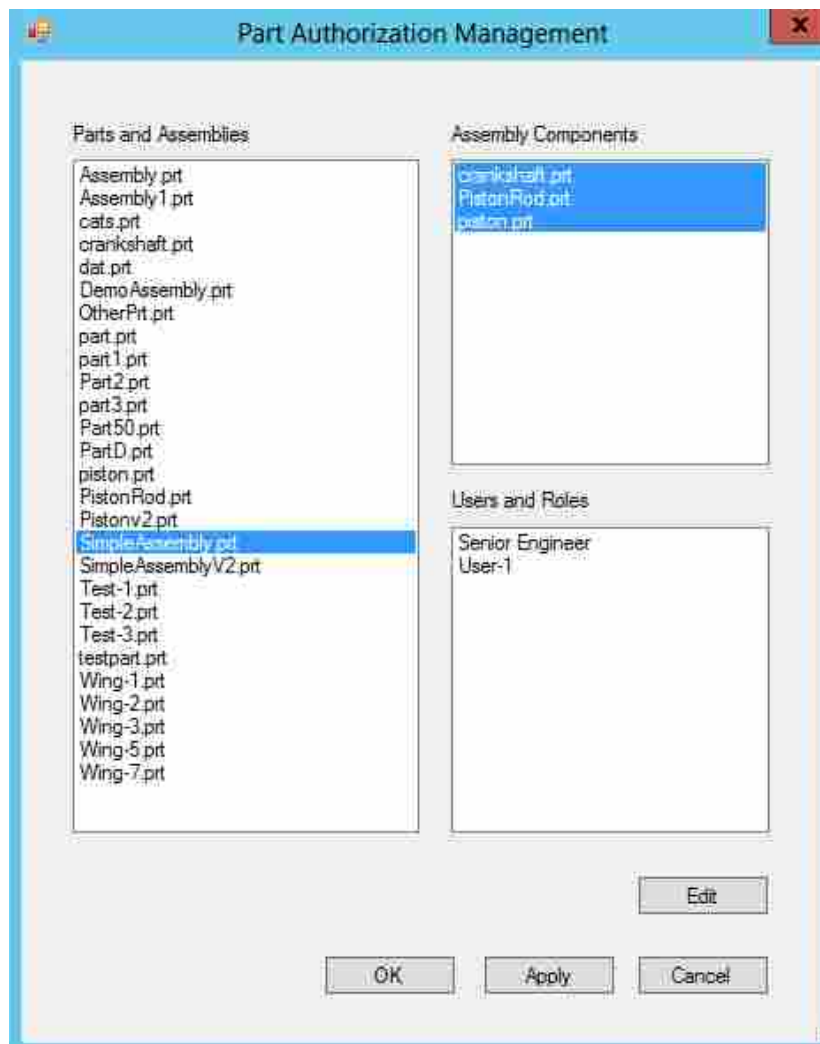
- Security Class:** Confidential-1, Confidential-2, Not Confidential
- Need to Know Categories:** Project 1, Default
- Roles:** Technician, Junior Engineer, Senior Engineer
- Global Permissions:** Create New, Modify, Delete

A "Close" button is located at the bottom right of the window.

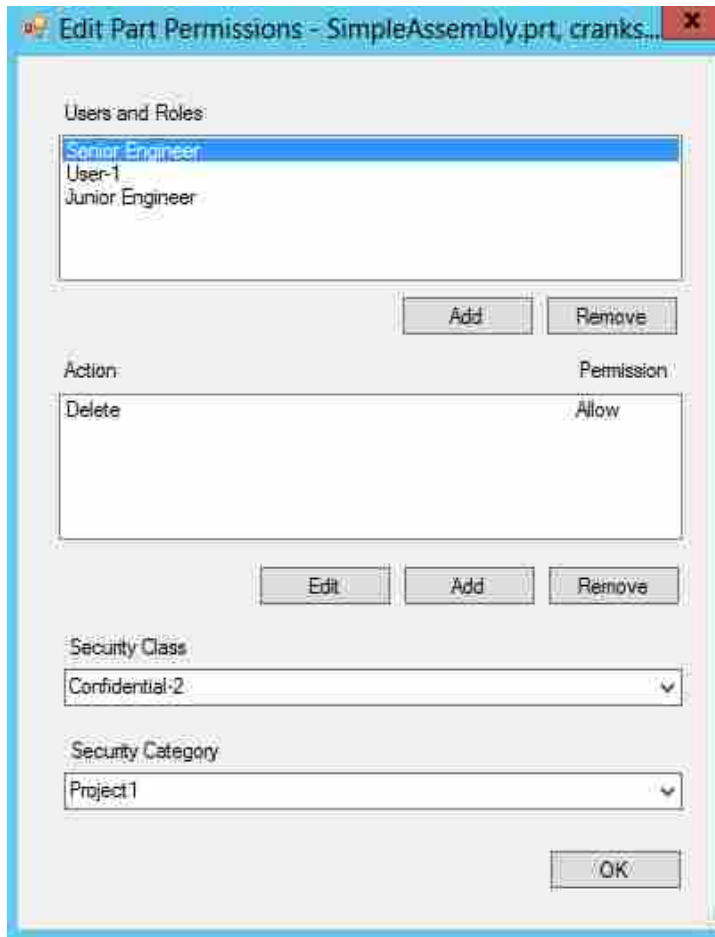
## Managing Users and Roles

## Managing Part Authorization

To manage authorization for a part or an assembly, select the name from the Parts and Assemblies listbox. If it is an assembly the list of the assembly components will appear in the Assembly Components listbox. Click the 'Edit' button to manage which users or roles have access to the assembly or part. Permissions can be managed either at the assembly level or at individual part level.



**Part Authorization Management**



**Interface for Editing Part Permissions**