A REPRODUCTION TRANSACTION SYSTEM

Master Thesis performed at the Department of Electrical Engineering at Linköping University by

MÅRTEN BJÖRK AND SOFIA MAX

Reg no: LiTH-ISY-EX-3262-2003
Linköping 2003
A REPRODUCTION TRANSACTION SYSTEM

Master Thesis performed at the Department of Electrical Engineering at Linköping University by

MARTEN BJÖRK AND SOFIA MAX

Reg no: LiTH-ISY-EX-3262-2003

Supervisor: KLAS NORDBERG
Johan Wiklund

Examiner: KLAS NORDBERG

Linköping 12th February 2003.
**Titel**

ARTSY - A Reproduction Transaction System

**Författare**

Sofia Max, Mårten Björk

**Sammanfattning**

A Transaction Reproduction System (ARTSY) is a distributed system that enables secure transactions and reproductions of digital content over an insecure network. A field of application is reproductions of visual arts: A print workshop could for example use ARTSY to print a digital image that is located at a remote museum. The purpose of this master thesis project was to propose a specification for ARTSY and to show that it is technically feasible to implement it.

An analysis of the security threats in the ARTSY context was performed and a security model was developed. The security model was approved by a leading computer security expert. The security mechanisms that were chosen for the model were: Asymmetric cryptology, digital signatures, symmetric cryptology and a public key registry. A Software Requirements Specification was developed. It contains extra directives for image reproduction systems but it is possible to use it for an arbitrary type of reproduction system. A prototype of ARTSY was implemented using the Java programming language. The prototype uses XML to manage information and Java RMI to enable remote communication between its components. It was built as a platform independent system and it has been tested and proven to be operational on the Sun Solaris platform as well as the Win32 platform.

**Nyckelord**

distributed system, XML, Java, cryptography, computer security, key establishment protocol, CORBA, Java RMI, RSA, DES, security threats
Abstract

A Transaction Reproduction System (ARTSY) is a distributed system that enables secure transactions and reproductions of digital content over an insecure network. A field of application is reproductions of visual arts: A print workshop could for example use ARTSY to print a digital image that is located at a remote museum. The purpose of this master thesis project was to propose a specification for ARTSY and to show that it is technically feasible to implement it.

An analysis of the security threats in the ARTSY context was performed and a security model was developed. The security model was approved by a leading computer security expert. The security mechanisms that were chosen for the model were: Asymmetric cryptology, digital signatures, symmetric cryptology and a public key registry. A Software Requirements Specification was developed. It contains extra directives for image reproduction systems but it is possible to use it for an arbitrary type of reproduction system. A prototype of ARTSY was implemented using the Java programming language. The prototype uses XML to manage information and Java RMI to enable remote communication between its components. It was built as a platform independent system and it has been tested and proven to be operational on the Sun Solaris platform as well as the Win32 platform.
Vocabulary

This section describes the meaning of some important and frequently used terms. It should be noted that the terms explained have a special meaning within this document and that this meaning in some cases differs from the commonly used meaning of the same terms.

Original
An Original is a physical artifact. A painting can be referred to as an original. A Reproduction of a painting can never be called an original.

Content
Content is an abstract representation of a physical object (Original), it is represented by a set of one or more Content Representations.

Content Representation
A Content Representation is a digital representation of some Content. A picture could for example be represented by a jpeg file and a bitmap file. Content Representations are either permanently stored at a Content Provider or dynamically generated when needed.

Content Provider
A Content Provider is a set of devices capable of storing and transmitting Content Representations. The term includes both the hardware and software needed to accomplish this and the personnel needed to administrate the system. A Content Provider has the legal rights to offer Content Representations to a buyer. A museum that has produced digital versions of some of its paintings could for example act as a Content Provider.

Reproduction
A reproduction is a physical manifestation of a Content Representation. If a museum, for example, creates a digital version of a painting, using a digital camera or a scanner, and then uses this digital version to produce a printed poster, that poster would be a Reproduction.

Reproduction Client
A Reproduction Client is a set of devices capable of receiving Content Representations and making Reproductions. The Reproduction Client uses a Reproduction Unit to do make Reproductions. The term Reproduction Client
includes the Reproduction Unit, the components needed to receive a Transaction and the personnel needed to administrate this system. A print workshop could for example act as a Reproduction Client.

Reproduction Unit
A reproduction unit is the hardware and software needed to reproduce Content Representations. Every Reproduction Unit is a part of a Reproduction Client. A printer can for example be a Reproduction Unit.

Transaction
A transaction is a procedure in which a Content Representation is transmitted from a Content Provider to a Reproduction Client. A Transaction Server is involved in the procedure to ensure that the Content Representation is protected if that is needed.

Transaction Server
A Transaction Server mediates Transactions. Its most important task is to ensure that Content Representations that are transmitted reach their right destination and that it is impossible for an outsider to record transmissions and make illicit copies of Content Representations. The Transaction Server also ensures that the Content owner can be paid for each reproduction.

Component
An ARTSY Component is the hard- and software of a Reproduction Client, a Content Provider or a Transaction Server.

Party
An ARTSY Party includes an ARTSY Component and the personnel and users that interact with and administrates that Component.

Client
An ARTSY Client is either a Reproduction Client or a Content Provider. A Transaction Server is not a Client.

Trusted Party
A Trusted Party provides ARTSY Components with public encryption keys by maintaining a public database.
ARTSY Abbreviations

ARTSY  A Reproduction Transaction System  
CP    Content Provider    
RC    Reproduction Client    
RCi   Reproduction Client inner module    
RCo   Reproduction Client outer module    
TS    Transaction Server

Other Abbreviations

AES  Advanced Encryption Standard    
API  Application Programmers Interface    
CA  Certificate Authority    
CBC  Cipher Block Chaining    
CFB  Cipher FeedBack    
DES  Data Encryption Standard    
DTD  Document Type Definition    
ECB  Electronic Code Book    
FIPS  Federal Information Processing Standards    
HTML  Hyper Text Markup Language    
JIT  Just In Time    
IDEA  International Data Encryption Algorithm    
IEEE  Institute of Electrical and Electronics Engineers    
IIOP  Internet Inter-Orb Protocol    
IV  Initialization Vector    
JDOM  Java Document Object Model    
JOMM  Johans ObjektOrienterade Modell    
JVM  Java Virtual Machine    
MD5  Message Digest 5    
NIST  National Institute of Standards and Technology    
OFB  Output Feedback Mode    
OOP  Object Oriented Programming    
PRNG  Pseudo Random Number Generator    
RC2  Rivest Cipher 2    
RC5  Rivest Cipher 5    
RFC  Request For Comments    
RSA  Rivest Shamir Adleman    
SDK  Software Development Kit    
SHA-1  Secure Hash Algorithm    
SQL  Structured Query Language    
TIFF  Tagged Image File Format    
XHTML  eXtended HTML    
XOR  eXclusive OR    
XSL  eXtensible Stylesheet Language
Contents

1 Introduction ......................................................... 1
  1.1 Background .................................................. 2
  1.2 Purpose ...................................................... 4
  1.3 Document Overview ........................................... 4

2 Theory .......................................................... 5
  2.1 Network Security .............................................. 5
    2.1.1 Security Services ....................................... 5
    2.1.2 Security Mechanisms ..................................... 7
    2.1.3 Security Attacks ......................................... 19
  2.2 Programming Languages ...................................... 20
    2.2.1 C and C++ ............................................... 20
    2.2.2 Java ..................................................... 21
  2.3 Distributed Systems .......................................... 25
    2.3.1 Java RMI ................................................ 26
    2.3.2 CORBA .................................................. 27
    2.3.3 Java RMI versus CORBA ................................ 27
  2.4 Markup Languages ........................................... 28
    2.4.1 SGML .................................................... 28
    2.4.2 XML ..................................................... 29

3 Method .......................................................... 33
  3.1 Main Features Model ......................................... 33
  3.2 Security Analysis ............................................ 33
  3.3 Software Requirements Specification .......................... 34
  3.4 Prototype .................................................... 34

4 Result .......................................................... 35
  4.1 Main Features Model ......................................... 35
    4.1.1 Information Request ...................................... 37
    4.1.2 Tender Submission ....................................... 39
    4.1.3 Transaction ............................................... 41
  4.2 Security Analysis ............................................ 43
## Contents

4.2.1 Security Threats ................................................. 43
4.2.2 Security Protocol ............................................. 44
4.3 Software Requirements Specification ........................... 49
4.4 Prototype ......................................................... 49
  4.4.1 Dependencies ................................................. 51
  4.4.2 Code Structure ............................................... 51
  4.4.3 Security ......................................................... 53
  4.4.4 Flowcharts of the Code ..................................... 55

5 Discussion ....................................................... 61
  5.1 Main Features Model ........................................... 61
    5.1.1 Method ....................................................... 61
    5.1.2 Result ......................................................... 62
  5.2 Security Analysis ............................................. 64
    5.2.1 Method ....................................................... 64
    5.2.2 Result ......................................................... 64
  5.3 Software Requirements Specification ........................ 69
    5.3.1 Method ....................................................... 69
    5.3.2 Result ......................................................... 69
  5.4 Prototype ....................................................... 70
    5.4.1 Method ....................................................... 70
    5.4.2 Result ......................................................... 71

6 Conclusion ..................................................... 77
  6.1 Recommendations ............................................... 78

Appendices

A ARTSY Software Requirement Specifications
B ARTSY Prototype documentation
Contents

List of Figures

2.1 A taxonomy of cryptographic primitives [1] .......................... 7
2.2 Digital signature generation and verification [2] .................. 12
2.3 A classification of key establishment techniques [1] ............. 16
2.4 Communication layers .............................................. 25
2.5 A method is invoked on the server via Java RMI .................. 27

4.1 An RC orders a Content Representation .......................... 35
4.2 A small ARTSY .................................................. 36
4.3 Information Request ................................................ 37
4.4 An Information Response describing an image .................. 38
4.5 Tender Submission .................................................. 39
4.6 A Tender .......................................................... 40
4.7 Transaction - Step 1 ............................................... 41
4.8 Transaction - Step 2 ............................................... 41
4.9 An order message is secured and unsecured ...................... 46
4.10 Public Keys are acquired from the Public Key Registry .. 47
4.11 The ARTSY prototype ............................................. 49
4.12 Remote method invocation during a Tender Submission .. 50
4.13 Information Request .............................................. 55
4.14 Tender Submission ................................................ 56
4.15 Transaction part 1: Transaction Start messages are received . 57
4.16 Transaction part 2: Ready To Transmit messages are received . 58
4.17 Transaction part 3: An image is printed ......................... 59

List of Tables

2.1 The average time needed to break a symmetric key cryptosystem . 13
2.2 Recommended asymmetric key lengths (in bits) ................... 14
2.3 Symmetric and asymmetric key sizes that are comparably strong (bits) 14

4.1 Transaction result ................................................... 42
4.2 Security level comparison chart .................................... 49

5.1 Consequences for a CP when choosing a security level .......... 68
5.2 Consequences for an RC when choosing a security level .......... 68
Chapter 1

Introduction

Miss Jonsson lives in Stockholm, Sweden. Some time ago she attended an exhibition at the National Museum of Art in Stockholm. She enjoyed the exhibition much and there was one painting in particular that she liked. The painting was called *Sunrise in Tibet* and was borrowed from the Museum of Modern Art in Brussels, Belgium. Unfortunately the National Museum of Art in Stockholm did not offer a reproduction of the painting. Miss Jonsson started looking for a reproduction all over. After a couple of weeks of unsuccessful searching Miss Jonsson realized that the only place from where she could get a reproduction of *Sunrise in Tibet* was from the museum shop at the Museum of Modern Art in Brussels. She then gave up the idea of getting a reproduction. Two years later she overhears a conversation on the subway. An elderly couple are talking about a poster shop:

"You can get a reproduction of any piece of art that you like, from a museum anywhere in the world. And you can get it the same day that you order it!"

Miss Jonsson is very excited when she enters the poster shop. She approaches the man at the desk and asks him if she could order *Sunrise in Tibet* from the Museum of Modern Art in Brussels. The poster shop and the Museum of Modern Art in Brussels are both connected to ARTSY. The Museum of Modern Art in Brussels stores all its art digitally and makes it available for sale all over the world through ARTSY. The man at the poster shop starts tapping the keyboard of his computer and replies, after a couple of seconds, that the reproduction is available for 240 kronor.

Miss Jonsson is thrilled and one minute later her long awaited reproduction has been ordered. The Museum of Modern Art in Brussels receives the order from the poster shop in Stockholm and automatically sends the digital image to the shop via the Internet. The museum charges the poster shop five euros for printing a copy of *Sunrise in Tibet* and the poster shop charges another 190 kronor for making a high quality print-out. Miss Jonsson returns to the poster shop an hour later and finally she gets the reproduction that she has longed for.
1.1 Background

Many museums, archives and libraries have established digital archives containing digital representations of material in their holding. Some of these organizations use these archives for producing reproductions. At present such reproductions are made in-house, right next to the digital archive. There is yet no universally accepted system for making digital archives available at external sites. A need for such a system arises when individuals wish to be provided with high-quality reproductions of material for which the original is not directly available.

A digital original that can be used for producing reproductions usually represents a substantial economical value for the owner. This means that owners of this type of content expect to be paid for each reproduction, even if it is being made at an external site. Digital originals can not be shared over a public medium such as the Internet unless the data is encrypted and the transmission is secure.

An infrastructure that supports secure remote reproductions must implement a reliable technique for distributing encryption keys to both the content provider and the reproduction unit. Such a system must also implement a reliable technique for authenticating the identity of the involved parties. Moreover, it must be feasible to add and remove parties to and from the system without affecting the remaining parties.

The ARTSY-project (A Reproduction Transaction System) is an attempt to define and implement a prototype of the secure print-on-demand transaction system described above. The main part of the project is conducted as a master thesis project at the Department of Electrical Engineering at Linköping University. The primary customer of the project is the Computer Vision Laboratory, Department of Electrical Engineering, at Linköping University. The project is supported by Hewlett-Packard within their Art & Science Programme 2002. The primary customer cooperates in the project with the Museum Domain Management Association (MuseDoma), the administrator of the museum top level domain, and the Division of Information Theory, Department of Electrical Engineering, at Linköping University.

The Louvre Museum in Paris, France and the National Gallery in London, UK are two potential future users of ARTSY. Representatives from these two museums participated to the HP Art & Science conference held on Cyprus in September 2002. Klas Nordberg, a representative from the Electrical Engineering Department at Linköping University, and the examiner of this master thesis work, also participated to the conference. Contact was established between the museums and the Electrical Engineering Department and the museums declared that they were willing to participate in the ARTSY-project.

Requirements

The primary customer of this project has specified a number of formal requirements that apply to the project. The requirements are presented below:
1.1 Background

- ARTSY shall be based on three different types of components: Content Providers (CPs), Reproduction Clients (RCs) and Transaction Servers (TSs). CPs store and send Content Representations. RCs receive Content Representations and make Reproductions. TSs mediate transactions and grant security to the parties involved in a transaction.

- The project shall result in a Software Requirements Specification for ARTSY as well as an ARTSY prototype implementation.

- The Software Requirements Specification shall describe an encryption and verification procedure which makes it possible to securely transmit Content Representations from a CP to an RC.

- The Software Requirements Specification shall describe how a Reproduction Transaction is carried out.

- The Software Requirements Specification shall describe how to deal with cost specifications in multiple currencies.

- The Software Requirements Specification shall describe how multiple TSs are supported by ARTSY.

- The prototype implementation shall utilize existing software whenever possible.

- The prototype implementation shall utilize existing and open standards for information representation and communication, e.g. established by W3C.

- It shall be possible to deploy the prototype implementation on multiple hardware/software platforms.

Delimitations

The primary customer has also specified a number of delimitations that simplifies the project. The delimitations are presented below:

- The Software Requirements Specification should be easily adapted to various types of content, but only Reproductions of images need to be demonstrated in the project.

- The Software Requirements Specification does not have to specify a protocol for managing money transactions.

- The Software Requirements Specification must consider Network security. Local security, such as access control and security of operating systems etc, does however not have to be considered.

- The Software Requirements Specification does not have to consider Human Computer Interaction (HCI) aspects.
1.2 Purpose

The purpose of this master thesis project is to propose a specification of ARTSY and to show that it is technically feasible to implement ARTSY. More specifically this includes:

- Developing an ARTSY Main Features Model
- Analyzing the security threats in the context of ARTSY and establishing a security protocol for ARTSY
- Developing a Software Requirements Specification for ARTSY
- Implementing a prototype for ARTSY and writing a users guide for this prototype

1.3 Document Overview

This document is divided into seven chapters. Chapter one gives an introduction to the project and presents its purpose. Chapter two contains theory that may be needed to understand the subsequent chapters. Chapter three presents the methods that were used to achieve the purpose. Chapter four presents the results. Chapter five discusses the results as well as the methods used for achieving them. Chapter six contains a concise summary of the results. Chapter seven gives suggestions on how ARTSY could be developed further. There are two appendices: Appendix A - ARTSY Software Requirements Specification, and appendix B - ARTSY prototype documentation.

Intended Audience

The reader is expected to have basic knowledge in the following areas: Computer science, computer security and computer networks.
Chapter 2

Theory

This chapter contains theory that may be needed to understand the subsequent chapters. Chapter 4 (where the result of the project is presented) contains references to this chapter.

2.1 Network Security

To assess the security of an organization some systematic way of defining the requirements of security is needed. One approach is to consider three aspects of information security:

- Security services
- Security mechanisms
- Security attacks

A security service is a service that enhances the security of the data processing systems and the information transfers of an organization. The service counters security attacks and make use of one or more security mechanisms to provide the service. A security mechanism is a mechanism that is designed to detect, prevent, or recover from a security attack. A security attack is any action that compromises the security of information owned by an organization [3]. The three aspects are described further below.

2.1.1 Security Services

The security services can be divided into six categories: Confidentiality, data integrity, authenticity, non-repudiation, access control and availability [3]. The six categories are described below.
Confidentiality

Confidentiality is a service used to keep the content of information from all but those authorized to have it. This service safeguards data against the threat of communications monitoring. Encryption is a common mechanism to implement the confidentiality service. Secrecy and privacy are two terms synonymous with confidentiality.

Data integrity

Data integrity is a service which addresses the unauthorized manipulation of data. Data manipulation includes such things as duplication of a message, deletion of part or all of a message, reordering of a message sequence and substitution of a message. Encryption and hash functions are two commonly used mechanisms for providing data integrity [1].

Authenticity

Authenticity is a service related to identification. It provides assurance for the claimed identity of a user — it verifies that you are who you claim you are. Authentication also guards against improper creation of messages or replay of old messages [1].

Non-repudiation

Non-repudiation is a service which prevents a party from denying previous commitments or actions. A party may deny its involvement to cheat another party. It may dispute the content of a transaction that has occurred, the transaction time, or the identity of the other party. A non-repudiation service gathers strong evidence during a communication that can be used to resolve such disagreements. A procedure involving a trusted third party is needed to resolve the dispute [1]. Encryption and digital signatures are two techniques used to implement this service.

Access Control

Access control is a service that guards against improper, unauthorized access to data or to a computer resource. Access control establishes what a user is allowed to do. A security system often needs to authenticate a user before it can determine his or her access privileges [3].

Availability

Availability is a service that ensures that a computer system, data, hardware and software resources are available to authorized users when needed [3].
2.1 Network Security

2.1.2 Security Mechanisms

This section presents a number of basic cryptographic tools, also called \textit{primitives}, that are used to provide information security. These primitives can be divided into different categories according to figure 2.1. Security mechanisms that make

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{taxonomy.png}
\caption{A taxonomy of cryptographic primitives [1]}
\end{figure}

use of the primitives are then explained. The mechanisms that are explained are certificates and key establishment protocols.
Symmetric Cryptography

Symmetric cryptography\(^1\) uses a secret key to encrypt plaintext into ciphertext and the same key to decrypt the ciphertext back into plaintext. Two basic categories of symmetric ciphers are block ciphers and stream ciphers [2].

Block ciphers take a fixed-size block of plaintext, usually 64 bits, and produce a fixed-size block of ciphertext, usually the same size as the input block. If the plaintext is longer than one blocksize a cipher mode (see page 9) must be used that describes how the cipher shall be applied block by block to the data stream. The key bit length varies, but it is always a multiple of 2 and it is normally not less than 56 bits. The block size must be long enough to prevent known-plaintext attacks and short enough to avoid complications and implementation inefficiencies [2]. The mapping from an input block must be one-to-one to make the algorithm reversible. If more than one input block is mapped to one output block the algorithm can encrypt, but it cannot decrypt. Patterns in the ciphertext become a problem when block ciphers are applied to streams of data [4]. A block that is encrypted twice with the same key will result in the same ciphertext. These patterns can be used by an adversary to attack the cipher. Some common symmetric block ciphers are: DES, IDEA, RC2, RC5 and triple DES [2]. AES is a Federal Information Processing Standard (FIPS) [5], which is intended to replace DES. DES was developed more than 20 years ago and has been used in many systems over the years. It has withstood intense cryptanalytic scrutiny and no easy way of attacking DES has yet been discovered. But since DES uses encryption keys that are only 56 bits long it is not considered secure [6]. A brute-force exhaustive search of the key space could be done in less than 11 minutes using the most powerful computer in the world (see table 2.1, page 13).

Stream ciphers operate over one bit of input data at a time [2]. They are, in one sense, very simple block ciphers having a block length equal to one. In situations where transmission errors are highly probable, stream ciphers are advantageous because they have no error propagation [1]. They can also be used when the data must be processed one symbol at the time (if the equipment has no memory or buffering of data is limited). Today, very few stream ciphers are used in commercial systems. The most well known is RC4 [4]. The one-time pad is a perfect encryption scheme and was invented as early as 1917. A one-time pad is a large non-repeating set of truly random numbers. The pad has the same size as the plaintext and can only be used once. By XOR:ing the pad with the plaintext the ciphertext is produced. If the adversary is unable to get access to the pad this is encryption scheme is perfect [7].

---

\(^1\)Symmetric cryptography is also called secret-key cryptography.
2.1 Network Security

Asymmetric Cryptography

Asymmetric cryptography\(^2\) involves the use of two distinct keys, one public and one private. The private key is kept secret and must never be compromised. The public key is not secret and can be freely shared with anyone. A public key and its associated private key are called a key pair. The keys in the key pair are mathematically related, but it is computationally infeasible to derive the private key from the public key. When using an asymmetric algorithm, a message is encrypted with the public key and decrypted with the private key. If the message is encrypted with the private key, everyone that is in possession of the public key can decrypt the encrypted message. But since there is only one that is in possession of the private key, everybody that decrypts the encrypted plaintext can verify that it was sent from private key owner. A message that is encrypted with a private key is called a digital signature. The process of decrypting the message is called signature verification. An asymmetric algorithm is reversible if it can be used for both digital signatures and encryption. If it can be used only for digital signatures it is called irreversible \(^2\). A third category can be used neither for digital signatures nor encryption. These kind of algorithms are called key establishment algorithms because they can be used to securely negotiate a secret key between two parties. Only three irreversible algorithms are secure and practically usable: RSA, ElGamal, and Rabin \(^7\). Asymmetric algorithms are about 100 to 1000 times slower than symmetric algorithms and are therefore rarely used to encrypt a large amount of data \(^2\).

RSA is the most popular algorithm and the easiest to implement. Its security is based on the difficulty of factoring large numbers. The public and private keys are calculated using a pair of large prime numbers. Recovering the plaintext from the public key and the ciphertext is believed to be equivalent to factoring the product of the two numbers, but it has not been mathematically proven. If DES is implemented in software it is about 100 times faster than RSA. If DES is implemented in hardware it performs even better \(^7\).

Cipher Mode

The term cipher mode refers to a set of techniques used to apply a block cipher to a data stream \(^4\). A cipher mode must be used if the plaintext is longer than the block size. Each mode defines a method of combining the plaintext, key, and encrypted ciphertext in a special way to generate the stream of ciphertext actually transmitted to the recipient. There are certain considerations that have to be taken before applying a mode to a cipher. The mode must not compromise the security of the underlying algorithm and the plaintext should be concealed. The mode should not be significantly less efficient than the underlying cipher and it might be important that the decrypting process should be able to recover from bit errors in the ciphertext stream \(^7\). In practice, four basic modes are used \(^4\):

\(^2\)Asymmetric cryptography is also called public-key cryptography
• Electronic Code Book (ECB) is the most obvious way to use a block cipher. One block of plaintext is encrypted into one block of ciphertext [7]. This is the most efficient mode [4]. One or more bit errors in a single ciphertext block affect decryption of that block only [1]. The ECB mode does however have security problems since patterns in the plaintext can yield patterns in the ciphertext. It is also easy to modify a ciphertext message by adding, removing, or switching encrypted blocks [4].

• Cipher Block Chaining (CBC) XORs a plaintext block with the ciphertext of the previous block prior to encryption. Since the first block does not have a preceding block, an initialization vector (IV) is used to modify the first block [2]. This vector does not need to be secret but its integrity should be protected [1]. This mode requires that a complete block of data is received before encryption can begin. Two messages will never yield the same ciphertext, even if the plaintexts are identical, as long as the IV is different for each message [4]. The error recovery is worse than in ECB mode, a single-bit error in the ciphertext affects an entire block and one bit in the following block, of the recovered plaintext. Blocks after the second block are not affected by the error. If a bit is added or lost all the blocks after the lost or added bit will be altered [7].

• Cipher Feedback (CFB) mode enables data to be encrypted in smaller units than the block size. This means that the CFB mode can be used also with stream ciphers. As in CBC an IV is needed to initialize the process. It does not need to be secret but in contrast to CBC it must be unique (when using CBC it should be unique but it does not have to be). If the IV is not unique an adversary can recover the corresponding plaintext. In n-bit CFB a single ciphertext error will affect the decryption of the current and the following $\frac{m}{n-1}$ blocks where m is the block size [7]. For example: In 8-bit CFB mode where the block size is 64 bits, a single-bit error yields 9 bytes of incorrectly decrypted plaintext. The CFB mode can be used with any symmetric block cipher. If it is used with an asymmetric cipher algorithm the cipher must be initialized with the same key each time, when encrypting and decrypting [8]. The CFB mode must therefore not be used if the block cipher is a public-key algorithm; instead, the CBC mode should be used [7].

• Output Feedback (OFB) mode may be used for applications in which all error propagation must be avoided [1]. It is similar to CFB and allows encryption of various block sizes, it can therefore also be used with stream ciphers. It uses a secret key to create a large pseudo-random key stream, which is XOR:ed with the plaintext to produce the ciphertext [2]. An efficiency benefit to OFB is that the key stream may be computed before data is ready to be sent or before it is received, since the key stream relies only on the key and the IV [4]. The OFB mode cannot be used with public-key algorithms since it uses the same key for decrypting and for encrypting (as the CFB mode) [8].
2.1 Network Security

Session Keys

A session key is a key that is restricted to a short time period such as a single Internet connection. After a session key has been used all trace of it is eliminated. By using a session key the available ciphertext under a fixed key is limited for cryptanalytic attack. The exposure is also limited in the event of session key compromise. This is with respect to both time period and quantity of data. When session keys are used long-term storage of distinct secret keys is not needed, keys are only created when actually required [1].

One-way Hash Functions

A one-way hash function\(^3\) takes a variable-length message as input and produces a fixed-length hash value as output [2]. A one-way hash function is a function \(h\) that has the following properties [1]:

- \(h\) maps an input \(x\) of arbitrary finite bit length, to an output \(h(x)\) of fixed bit length.
- Given \(h\) and an input \(x\), \(h(x)\) is easy to compute.
- Given \(h\) and an output \(h(x)\), it is computationally infeasible to find \(x\).
- Given \(h\), \(x\) and \(h(x)\), it is computationally infeasible to find an input \(y\), such that \(h(y) = h(x)\).

One-way hash functions are used to create digital signatures [2]. They are also used to create checksums of data that is transmitted over the Internet: A client that downloads data also downloads its hash value. The client then applies the same hash function to the data as the provider of the data did. If the hash value that is produced is identical to the downloaded hash value the origin of the data is verified [4].

MD5 is a frequently used hash function, but it is being slowly phased out due to some theoretical weakness [2]. SHA-1 is an improved version of MD5 that uses 160 bits instead of 128, it is therefore more resistant to brute-force attacks [7].

Digital Signatures

Digital signatures is a mechanism that protects a message from undetected change. The digital signature associates a message with the owner of a specific private key [4]. The sender of a message applies a hash function to the message and a hash value is produced. The hash value is encrypted using the private key of the sender. The encrypted hash value is called a signature. The signature is sent to a receiver along with the message. The receiver applies the same hash function as the sender did to the message. Another hash value is produced, this value is called the actual hash value. The expected hash value that was received from the sender

\(^3\) One-way hash functions are also called message-digest algorithms.
is then compared to the actual hash value. If they match the message is authentic and has not been changed during the transmission.

It is possible to sign messages directly, instead of signing the hash values. This is not done since it has a number of shortcomings. First, signing an entire message and sending the signature along with the message doubles the bandwidth requirements. Second, asymmetric key operations are relatively slow and encrypting an entire message instead of its hash value impacts performance. Third, a cryptanalyst can use a plaintext message and its corresponding ciphertext to perform a known-plaintext attack against the cryptosystem [2]. The signing and verification process is illustrated in figure 2.2.

![Diagram of digital signature generation and verification](image)

**Figure 2.2.** Digital signature generation and verification [2]

**Key Length**

The security of a symmetric cryptosystem depends on two things: The strength of the algorithm used and the length of the encryption key used [7]. To choose the right key is a matter of balancing security against performance. By increasing the key size, the security is increased but the performance is worsened. Long keys make cryptographic algorithms less efficient and increase the amount of ciphertext and bandwidth required. The right key length is found when the value of the encrypted information is less than the money spent to decrypt it [2].

To break a cryptosystem based on symmetric keys that does not have any known weaknesses, an adversary must try each possible key from the key space until the
ciphertext is decrypted into some recognizable message. This is called a brute-force attack [4]. To launch a brute-force attack the adversary needs a small amount of ciphertext and the corresponding plaintext. If the adversary knows anything about the format of the ciphertext some plaintext can be known: For example a TIFF image or UNIX directory file has some predefined bytes [7]. An 8-bit key has a key space of $2^8 = 256$ different possible keys. On average, it takes $2^7 = 128$ tries to recover the key in a brute-force attack. A symmetric key must be large enough to prevent a brute-force attack [2]. Assume that a computer can try one key per clock cycle. A PC that costs about $\$1000$ today\footnote{December 2002} would then have the capacity to try $2 \cdot 10^9$ keys per second. The most powerful computer (Cray X1)$^5$ that can be bought today costs about $\$2.4$ million and would have the capacity to try $5.24 \cdot 10^{12}$ keys per second. The growth of the Internet has made it possible to utilize thousands of machines in a distributed search by partitioning the key space and distributing small portions to each of a large number of computers [6].

Based on the assumptions above, the table 2.1 shows the average time needed to break a symmetric key cryptosystem. According to Moore’s law the computing power doubles approximately every 18 months. This means that every fifth year the computing power increases with a factor 10. In five years a $\$1000$ PC will be able to break a symmetric cryptosystem with 56 bits key length in only 2 days.

Breaking an asymmetric algorithm does not involve trying every possible key, instead it involves trying to factor large numbers [7]. Bruce Schneier recommends asymmetric key lengths according to table 2.2. This table recommends asymmetric key lengths that he thinks are secure when an the threat is comes from an individual, corporation or a government [7]. Schneier wrote his book in 1995 but in a newsletter sent in April, 2002 he still makes the same recommendations [9].

When using a cryptosystem that includes both symmetric and asymmetric techniques it is important to pick key sizes that are comparably strong (see table 2.3) [4].

<table>
<thead>
<tr>
<th>Key length</th>
<th>1 PC</th>
<th>10 PCs</th>
<th>100 PCs</th>
<th>Cray X1</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 bits</td>
<td>0 sec</td>
<td>0 sec</td>
<td>0 sec</td>
<td>0 sec</td>
</tr>
<tr>
<td>40 bits</td>
<td>5 min</td>
<td>30 sec</td>
<td>3 sec</td>
<td>0 sec</td>
</tr>
<tr>
<td>56 bits</td>
<td>208 days</td>
<td>21 days</td>
<td>2 days</td>
<td>11 min</td>
</tr>
<tr>
<td>64 bits</td>
<td>146 years</td>
<td>14 years</td>
<td>1.5 years</td>
<td>2 days</td>
</tr>
<tr>
<td>80 bits</td>
<td>&gt;9 million years</td>
<td>1 million years</td>
<td>100 000 years</td>
<td>365 years</td>
</tr>
</tbody>
</table>

Table 2.1. The average time needed to break a symmetric key cryptosystem

\footnote{http://www.cray.com, (1st February 2003)}
Table 2.2. Recommended asymmetric key lengths (in bits)

<table>
<thead>
<tr>
<th>Year</th>
<th>vs. Individual</th>
<th>vs. Corporation</th>
<th>vs. Government</th>
</tr>
</thead>
<tbody>
<tr>
<td>1995</td>
<td>768</td>
<td>1280</td>
<td>1536</td>
</tr>
<tr>
<td>2000</td>
<td>1024</td>
<td>1280</td>
<td>1536</td>
</tr>
<tr>
<td>2005</td>
<td>1280</td>
<td>1536</td>
<td>2048</td>
</tr>
<tr>
<td>2010</td>
<td>1280</td>
<td>1536</td>
<td>2048</td>
</tr>
<tr>
<td>2015</td>
<td>1536</td>
<td>2048</td>
<td>2048</td>
</tr>
</tbody>
</table>

Table 2.3. Symmetric and asymmetric key sizes that are comparably strong (bits)

<table>
<thead>
<tr>
<th>Symmetric Key Sizes</th>
<th>Comparable Asymmetric Key Sizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>56</td>
<td>256</td>
</tr>
<tr>
<td>70</td>
<td>384</td>
</tr>
<tr>
<td>80</td>
<td>512</td>
</tr>
<tr>
<td>96</td>
<td>768</td>
</tr>
<tr>
<td>112</td>
<td>1024</td>
</tr>
<tr>
<td>150</td>
<td>2048</td>
</tr>
</tbody>
</table>

ric key length that is much harder to break than the symmetric key length. The asymmetric key is both easier to attack and a more valuable target, since it is often used to encrypt symmetric session keys [4].

Random Key Generation

A practical and secure crypto system needs keys that cannot be guessed. It should not be possible for an adversary to predict what keys are being used. A good key generator will produce keys that cannot be guessed even if attackers know how the generator works. Since a computer is deterministic, software cannot generate truly random numbers by itself [10]. Instead, it creates what are called pseudo-random numbers from a single random "seed". This seed must be truly random. The result of flipping a coin is truly random but a number produced from a mathematical function is only pseudo-random [11]. Unpredictable keys come from a truly random seed and a good pseudo-random number generator. To generate a random sequence on a digital computer, one starts with a certain seed, then iteratively applies some transformations to it, progressively extracting as long as possible a random sequence. In general, one considers a sequence “random” if no patterns can be recognized in it, no predictions can be made about it, and no simple description of it can be found by an adversary.

Collecting truly random numbers is hard. Some methods include making hardware devices that generate noise, observing cosmic ray flux and observing light emissions from trapped mercury atoms. These methods require extra hardware that might not be a part of the already existing hardware where the software
2.1 Network Security

should be run. One solution to the randomness problem is to maintain a pool of information pertaining to physical parameters, properties and activity of the system. Anything that is determined by external factors can be used as input to the pool, such as the time between keystrokes, the timing of disk interrupts, number of network packages arrived, the number of page faults and the number of disk read/writes. The pool of randomness information is often called the entropy pool [12]. Entropy is a way to measure the amount of information in a series of numbers. The higher the entropy in a series of numbers is, the more difficult it is to predict a given number on the basis of the preceding numbers in the series. A sequence of good random numbers will have a high level of entropy, although a high level of entropy does not guarantee randomness. As an example, a file compressed with a software compressor such as Winzip has a high level of entropy, but the data is highly structured and therefore not random. If the entropy in the series of numbers is close to eight bits per character it means that the random numbers have a high level of entropy. A low level of entropy would for example be one bit per character [13]. The entropy of a arbitrary message is 1.3 bits per character, natural language is highly redundant [7]. Typically a cryptographically sound message digest such as MD5 or SHA is computed over the entropy pool and used as the next seed [12].

The Sun Solaris system file /dev/random is a file that returns truly random numbers every time it is read. The random numbers are produced by a random number generator that collects data from devices available to the kernel. The amount of entropy is estimated and the file is only readable if sufficient amount of entropy has been collected, otherwise the file is blocked [14].

The pseudo-random number generator that is used to produce the pseudo-random number shall not be chosen at random. Properties that a good pseudo-random number generator possess is statistical properties, a long period, speed, low memory, portability, and reproducibility. The general assumption is that the adversary knows the algorithm being used in the pseudo-random number generator. The random number generator is easily overlooked, and can easily become the weak line of a cryptosystem [15]. A pseudo-random number generator is needed to generate the prime numbers in the RSA-algorithm.

Key Establishment Protocols

Before a cryptographic algorithm can be used, all keys that are needed have to be in the right places. In a system with \( n \) users that uses symmetric-key techniques, each pair of users may need to communicate securely. This means that each combination of pairs must share a distinct secret key. In this case, each party must have \( n - 1 \) secret keys. The overall numbers of keys in the system (which may need to be centrally backed up) is \( \frac{n(n-1)}{2} \) or approximately \( n^2 \). As the size of a system increases, this number becomes unacceptably large.

Key establishment is a process or a protocol whereby a shared secret becomes available to two or more parties, for subsequent cryptographic use [1]. It solves the \( n^2 \) key distribution problem either by using a key transport protocol or a key
agreement protocol. When using a key transport protocol one party creates or otherwise obtains a secret value, and securely transfers it to the other(s). A key agreement protocol is a key establishment technique in which a shared secret is derived by two or more parties as a function of information contributed by, or associated with, each of these, such that no party can predetermine the resulting value. Many key establishment protocols involve a central trusted party. A key pre-distribution scheme is a key establishment protocol where the resulting established keys are completely determined in advance by initial keying material. In contrast, dynamic key establishment schemes are those whereby the key established varies from time to time. Figure 2.3 presents a classification of key establishment techniques. Reversible asymmetric key algorithms can be used in key establishment protocols, whereas irreversible cannot [2].

![Key establishment diagram](image)

**Figure 2.3.** A classification of key establishment techniques [1]

Protocols that uses both asymmetric and symmetric techniques are called hybrid protocols. The Bell-Hoel protocol is a hybrid protocol that uses both asymmetric and symmetric encryption in addition to both PK encryption and digital signatures. It is a key transport protocol that was designed for applications where there is an imbalance in processing power between two parties. This protocol does not involve a third party. In selected key management applications, hybrid protocols involving both symmetric and asymmetric techniques offer the best alternative. The optimal use of available techniques generally involves combining symmetric techniques for bulk encryption and data integrity with public-key techniques for signatures.
2.1 Network Security

It is generally desired that each party in a key establishment protocol is able to determine the true identity of the other. *Entity authentication* is the process whereby one party is assured of the identity of a second party involved in a protocol and that the second has actually participated. *Key authentication* is the property whereby one party is assured that no other party aside from a specifically identified second party may gain access to a particular key. *Key confirmation* is the property whereby one party is assured that a second party actually has possession of a particular secret key. When designing or selecting a key establishment technique for use, it is important to consider what assurances and properties an intended application requires. Characteristics which differentiate key establishment techniques include [1]:

- **Nature of the authentication** Any combination of the following may be provided: entity authentication, key authentication and key confirmation.

- **Reciprocity of authentication** Each of entity authentication, key authentication, and key confirmation may be *unilateral* (provided to one party) or *mutual* (provided to both parties).

- **Key freshness** A key is *fresh* if it can be guaranteed to be new, as opposed to possibly an old key being reused through actions of either an adversary or an authorized party.

- **Key control** In key transport protocols, one party chooses a key value. In key agreement, the key is derived from joint information, and it may be desirable that neither party is able to control or predict the value of the key.

- **Efficiency** Considerations include number of message exchanges (*passes*) required between parties, bandwidth required by messages, complexity of computations by each party (as it effects execution time), and possibility of pre-computation to reduce on-line computational complexity.

- **Third party requirements** Considerations include requirement of an on-line (real-time), off-line, or no third party and degree of trust required in a third party (trusted to certify public keys or trusted not to disclose long-term secret keys).

- **Type of certificates used** More generally, one may consider the manner by which initial keying material is distributed, which may be related to third party requirements.

- **Non-repudiation** A protocol may provide some type of receipt that keying material has been exchanged.
Certificates

A public-key certificate\(^6\) is a binding between an entity's public key and one or more attributes relating to its identity. An identity can be a person, a hardware device or a software process. The public-key certificate provides assurance that the public key belongs to the identified entity and that the entity possesses the corresponding private key [2]. In their simplest form, certificates contain a public key and a name of an entity. As commonly used, a certificate also contains an expiration date, the name of the certifying authority that issued the certificate and a serial number. Most importantly, it contains the digital signature of the certificate issuer, the certification authority [6].

A certificate authority (CA) is a Trusted Party that accepts certificate applications from entities. When an application has been received the CA authenticates the application and then issues the certificate for the entity. A CA also maintains status information about all issued certificates [2]. A CA must be reliable and the public key of the CA must be available to check signatures. By being reliable the authority must not sign a certificate without being confident that the attached public key belongs to the entity named in the certificate. The public key of the CA must be available when the signature of a certificate should be verified. This could be solved by including the CAs public key in the software that authenticates certificates [16].

Public Key Registry

A public key registry is a less complex alternative to certificates which does not involve a CA or certificates. It consists of a trusted public database and is set up to contain the authentic public key of each user in the system. Users in the system acquire keys directly from this registry. Since the keys stored in the registry are public, an interception attack on a transmission of a public key over an unsecured channel is acceptable. But if a modification attack is a relevant threat the public key can not be sent over an insecure channel [1].

Chain of Trust

When a system grows big and spans big geographical area it might be difficult for one Trusted Party to manage all users. Each user in the system must be registered in a Public Key Registry or have a certificate issued, and this could be facilitated if there are several Trusted Parties that certifies separate areas in a system. This could be solved by using a Chain of Trust, in a hierarchical fashion. In the top one single Trusted Party issues certificates for other Trusted Parties. These Trusted Parties issues certificates for all other users in the system. The public key of the top Trusted Party is still the public key that is delivered with the software to the end user. Since no end-user certificate is issued by the top trusted party a certificate of an end user is verified by first verifying the Trusted Party that issued the certificate.

---

\(^6\)Public-key certificates are also called digital certificates, digital IDs or just certificates
This is done with the public key of the top Trusted Party that was installed along with the software [17].

2.1.3 Security Attacks

There are a number of different attacks that could be made to compromise the security of information owned by an organization. These attacks can be classified into four general categories [3].

- **Interruption** An asset of the system is destroyed or becomes unavailable or unusable. This is an attack on **availability**. Examples are destruction of a hard disk or a denial of service attack.

- **Interception** An unauthorized party gains access to an asset. This is an attack on **confidentiality**. The unauthorized party could be a person, program or a computer. Examples include eavesdropping data that is being sent on a network.

- **Modification** An unauthorized party not only gains access to but tampers with an asset. This is an attack on **data integrity**. Examples include modifying the content of messages being transmitted in a network and altering a program so that it performs differently.

- **Fabrication** An unauthorized party inserts fake objects into the system. This is an attack on **authenticity**. Examples include the insertion of false messages in a network, replay of a message or the addition of records to a file. Repudiation is an attack where a sender denies having sent a message at all.

**Denial of Service Attacks**

A Denial Of Service attack (DOS-attack) is an interruption type security attack. Any act intended to cause a service to become unavailable or unusable can be considered a DOS-attack. In an Internet environment, a service might be an application such as a web or mail server, or a network service like routing of datagrams.

A distributed Denial Of Service attack exploits several machines to make the attack. Distributed denial of service attacks are the most effective because they can generate more traffic from more sources. This makes it much harder to identify the source of the attack and more difficult to resolve it. To prevent distributed DOS-attacks the best thing to do is to prevent the hosts and networks from being used to cause denial of service attacks on others, and encourage other network and system administrators to do the same [18].
2.2 Programming Languages

The most popular programming languages are either procedure oriented or object oriented and are either interpreted or compiled.

At a fundamental level, a procedure oriented language is designed first to provide the developer with a framework for issuing commands for the computer to execute and second to allow the developer to organize and manipulate data. An object oriented language is designed first to allow the developer to define objects that make up the program and the data they contain, and second to define the code that makes up the program [19]. Basic and Pascal are two examples of procedure oriented languages. SmallTalk is a pure object oriented language. There are also many hybrid languages such as Delphi (Object Pascal) and Perl 5.0 [19]. Object oriented programming (OOP) is the dominant paradigm today [20]. One promise of object orientation is that it promotes the reuse of code, resulting in better productivity for developers [21].

When using a compiled language, a compiler program is needed. The compiler translates the human-readable source files of a program into a machine-readable format. Compiled languages often have the benefit of producing high-performance code that is tailored for execution on a specific processor or processor architecture. In this case, compiled programs can be run only on the type of computer they were compiled for. Interpreted languages usually exists only as source code. When they are run, a program called an interpreter takes the source file and performs the actions indicated by the commands in the file. The interpreter is the only real application that is running. Among the benefits of interpreted languages is that programs written using them can be run easily on all platforms that the interpreter has been ported to [19].

2.2.1 C and C++

C is a compiled, procedure oriented language [22]. It is a powerful language for writing fast and highly tuned code and it is far more portable than Assembly. Modularity in C is limited to one level of subroutines [23]. C++ is a compiled, hybrid language, derived from C. It is a procedure oriented language with object oriented extensions. It is possible for the developer to use objects within programs, but it is also possible, and sometimes necessary, to use procedural code to accomplish certain tasks [19]. C++ does not enforce object oriented programming [22].

Direct memory management is central to C and C++. The programmer is responsible for allocating and freeing memory. Pointers are used to access memory locations directly. This allows lots of efficient low-level manipulations and allows programs to run at the fastest possible speed. But it also has many less desirable consequences. Any memory related error can result in very subtle bugs. It is, for example, possible to crash the system by directly accessing the areas of memory reserved for system use [19].
2.2 Programming Languages

C++ has become a popular hybrid programming language because its parent-language, C, is popular. C, in turn, has become popular by virtue of attractive characteristics, such as the following [24]:

- C is easy to learn
- C programs are fast
- C programs are concise
- C compilers are usually fast and concise
- C compilers and C programs run on many different sorts of computers.
- The popular operating system UNIX is written in C.

2.2.2 Java

Java is a simple, object oriented language that has many elements in common with C and C++, but has removed or streamlined the areas where many programmers have had difficulties or that have been the most frequent source of bugs [19]. The developers of Java, Sun Microsystems\(^7\), describes the Java platform in the following way:

*The Java platform is based on the power of networks and the idea that the same software should run on many different kinds of computers, consumer gadgets, and other devices [25].*

Java Programming Language

According to the white paper definition of Java, the Java programming language has the following design goals [25]:

- Simple, Object Oriented and Familiar
- Architecture Neutral and Portable
- Robust and Secure
- High Performance
- Interpreted, Threaded and Dynamic

The design goals are discussed below.

\(^7\)http://www.sun.com/ (1st February 2003)
Simple, Object Oriented and Familiar

The popularity of C++ was acknowledged when the Java language was created. The "look and feel" of Java was made as close as possible to that of C++ [22]. The syntax for Java is basically, a cleaned-up version of the syntax for C++ [20]. This allows C++ programmers to take advantage of C++ experience when designing Java programs. Java’s similarity with C++, combined with the elimination of some problematic features makes Java a simple programming language [22].

Java is an object oriented language. It does not implement the basic simple data types, such as integers, characters and floating point numbers, as objects, but in all other ways Java is a pure object oriented language. All program code and data reside within objects [19].

Architecture Neutral and Portable

Java code is usually both compiled and interpreted. Java source files can be compiled into a byte code format, which is not directly executable. This byte code is as close as possible to machine code without becoming platform specific, it represents machine language instructions for a virtual processor, the Java Virtual machine (JVM) [19]. To execute a Java program in byte code format a Java interpreter must be used, either a standalone or as a part of a Web browser [22]. A Java interpreter implements the JVM in software so that it can execute Java byte code files. This process is not unlike the way that emulator software makes it possible to run programs for one type of computer on another. Because Java code is being run by an interpreter, Java programs will run unmodified on any platform to which the Java interpreter has been ported. Recently it has become possible to compile Java programs directly into machine code for a specific processor. By doing so, the need for an interpreter is eliminated and the performance of the Java programs is enhanced. The disadvantage of compiling Java programs into processor specific code is that platform independence is lost. Such code cannot be distributed and executed on different computer platforms [19].

Robust and Secure

In Java, it is not possible to directly access memory by arbitrarily casting pointers to a different type or by using pointer arithmetic, as with C and C++. Java requires that the rules of type are strictly obeyed when working with objects. It is possible to cast a reference to a different type in Java, but only if the object really is of the new type. Because Java enforces strict type rules at runtime, it is not possible to directly manipulate memory in ways that can accidentally corrupt it. Java also protects the integrity of memory at runtime by checking array bounds. The developer is not allowed to corrupt memory by writing beyond the end of an array.

A garbage collector prevents Java programmers from needing to explicitly indicate which object should be freed from memory. If an object is not referenced any longer, the garbage collector will reclaim the memory occupied
by the object. As a consequence Java programmers do not have to worry about memory leaks that are hard to track.

Moreover, the Java *sandbox model* restricts code from taking actions that could possibly harm the system [21]. The Java interpreter stands between Java applications and the system. The interpreter restricts Java programs to an acceptable range of behavior. There are no Java methods for low-level mischief and no ways to link in compiled code to perform mischief by proxy. By deferring memory mapping to runtime, Java prevents hackers from forging data structures. There are no pointers in Java and this makes it impossible to manipulate data at restricted addresses [22].

**High Performance**

Java is designed as an architecture that favors network-oriented features — such as platform independence, program robustness, security and network mobility — over other concerns. The primary downside of Java, is its execution speed. When running on a virtual machine that interprets bytecodes, a Java program may be 10 to 30 times slower than an equivalent C++ program compiled to native machine code. In many cases the benefits of platform independence and improved program robustness will be worth the speed degradation. Sometimes, however, Java may be disqualified as a tool to solve a certain problem, because that problem requires the utmost in speed, and Java cannot deliver it. The garbage collector in Java can help make programs more robust and easier to design, but it adds a level of uncertainty to the runtime performance of a program. It is not possible to predict when the garbage collector will decide to collect garbage nor how long it will take. This lack of control of memory management makes Java a questionable candidate for software problems that require a real-time response to events.

If execution speed is critical, a Java program could alternatively be compiled to a platform-specific, monolithic native executable. Such a strategy will boost execution speed at the cost of binary platform-independence [21]. Recently, several Just-In-Time (JIT) Java compilers have been developed. A JIT compiler produces native code from Java bytecode instructions during program execution. The native code can then be sent directly to the processor. Code that is compiled by a JIT compiler, usually runs more quickly than code that is interpreted in the conventional way. Since JIT compilers compile code at run-time, platform independence is not lost.

**Interpreted, Threaded and Dynamic**

In the Java system, multiple processes (threads) execute simultaneously with Java itself apportioning resources to each of these threads. Java makes use of the preemptive, multi-threaded capabilities of the modern operating systems on which it has been implemented. Preemptive here means that different threads have different priorities. Low-priority tasks, such as garbage collection, can be executed in the background, waiting for available CPU cycles in order to do their work. Java does not restrict its multi-threadness to sys-
tem tasks. It also provides the developer with synchronization primitives to simplify thread management [22].

When a Java program is compiled, a separate class file for each class or interface is produced. When the program is interpreted the JVM loads the classes and interfaces and hook them together in a process of dynamic linking [21]. Linking in Java is thus deferred until runtime. One side effect of this dynamic approach is that compile time is quicker than for a traditional compiled language. The Java compiler checks all references to other classes for type correctness, but does not spend time assembling all the programs objects into a single executable. Another consequence of dynamic linking is that it allows for a flexible use of classes. For example, a string variable can dynamically be populated with the name of a class, and then be used to create an object of that class type [22].

Java as an interpreted language is discussed above (see *Architecture Neutral and Portable*).

**Java Software Development Kit**

The Java Software Development Kit (SDK), includes two components: Class libraries that are useful when creating Java programs, often referred to as the Java Application Interface (Java API), and tools for managing Java programs. The class libraries implement the general data types and services that a developer is likely to need when programming in Java. The SDK tools include programs for compiling, running, debugging and analyzing Java programs as well as generating documentation [22]. Some of the classes that are included in the Java 1.4 API packages are described below.

**Cipher**

The *Cipher* class is a part of the *javax.crypto* package and represents a cryptographic cipher, either symmetric or asymmetric. By specifying an algorithm name, a cipher mode and a padding scheme and if the cipher should be used for encryption or decryption, the object is ready to use. Encryption and decryption is made by invoking one of the methods of the Cipher object [8].

**SecureRandom**

The *SecureRandom* class is a part of the *java.security* package and provides a cryptographically strong pseudo-random number generator (PRNG). A cryptographically strong pseudo-random number minimally complies with the statistical random number generator tests specified in FIPS 140-2, Security Requirements for Cryptographic Modules [26], section 4.9.1. Additionally, SecureRandom must produce non-deterministic output and therefore it is required that the seed material is unpredictable and that the output of SecureRandom are cryptographically strong sequences as described in RFC 1750 [27]: Randomness Recommendations for Security. The algorithm that SUN delivers is called SHA1PRNG which is a pseudo-random number generation (PRNG) algorithm. This implementation follows the IEEE P1363
SignedObject

The SignedObject class is a also part of the java.security package. It can be used for the purpose of creating authentic runtime objects whose integrity cannot be compromised without being detected. For verification to succeed, the specified public key must be the public key corresponding to the private key used to generate the SignedObject. For flexibility reasons the constructor and verify methods allow for customized signature engines, which can implement signature algorithms that are not installed formally as part of a crypto provider. However, it is crucial that the programmer writing the verifier code is aware of what Signature engine is being used, as its own implementation of the verify method is invoked to verify a signature. In other words, a malicious Signature may choose to always return true on verification in an attempt to bypass a security check.

2.3 Distributed Systems

A distributed system is one in which components located at networked computers communicate and coordinate their actions only by passing messages. The Internet is an example of a very large distributed system. A distributed system that uses the Internet can be described in layers (see figure 2.4).

![Applications and services](#)

<table>
<thead>
<tr>
<th>Applications and services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java RMI and CORBA</td>
</tr>
<tr>
<td>UDP and TCP</td>
</tr>
</tbody>
</table>

**Figure 2.4. Communication layers**

The application is on the top and communicates with the middleware which can be represented by for example Java RMI or CORBA. The middleware commu-
nicates with the TCP or the UDP protocol. There are different mechanisms that enable communication between processes in a distributed system. The application program interface to UDP provides a message passing abstraction — the simplest form of interprocess communication. In the Java and UNIX APIs, the sender specifies the destination using a socket — an indirect reference to a particular process used by the destination process at the destination computer. The application program interface to TCP provides the abstraction of a two-way stream between pairs of processes. The information communicated consists of a stream of data items with no message boundaries.

Software that provides a programming model above the basic building blocks of processes and message passing is called middleware. The middleware is a software layer that provides a programming abstraction as well as as masking the heterogeneity of the underlying networks, hardware, operating system and programming languages. An example of this is CORBA. Java RMI is another example, but this middleware only supports the Java programming language [29].

2.3.1 Java RMI

Java RMI is designed to make communication between two Java programs, running on separate Java Virtual Machines, as much like making a method call inside a single process as possible.

RMI relies on two similar types of objects that are automatically generated by the RMI Compiler: Stubs and skeletons. A stub is a client-side object that represents a server object inside the JVM of a client. It implements the same methods as the server object, maintains a socket connection and is responsible for marshalling and de-marshalling data on the client side [30]. A skeleton is a server-side object responsible for maintaining network connections and de-marshalling data on the server side. See figure 2.5 for a basic RMI call with a stub and a skeleton.

Serialization is a process in which a set of object instances that contain references to each other are converted into a linear stream of bytes. Such a stream can be sent through a socket, stored in a file, or simply be manipulated as a stream of data. Serialization is used by Java RMI to pass objects between JVMs, as arguments in a method invocation or as return values from a method invocation. Every class that implements the Serializable interface\(^8\) can be sent as a parameter or as a return value.

Java RMI can be used to pass not only data between JVMs, but also code. In a distributed system that has been running for quite a while, it might be desirable to add new functions to objects that are passed as return values from a server to a client. With Java RMI this is possible. Code which is contained in an object can be replaced with new code at any time. This feature in Java RMI facilitates updates of applications [30].

\(^8\)To make a class implement the Serializable interface the programmer simply adds “implements Serializable” after the declaration of a class. This is allowed for most classes.
2.3 Distributed Systems

Figure 2.5. A method is invoked on the server via Java RMI

2.3.2 CORBA

CORBA is designed to integrate different applications, possibly written in different programming languages and running on different machines, into a single distributed application. CORBA relies on stubs and skeletons in the same way as Java RMI does. But to be able to use the stubs and skeletons automatically, all interfaces must be specified in a special CORBA language, called IDL. The IDL interfaces must then be mapped to the programming languages that are used. IDL has a rich set of data types, ways to define value objects and exceptions, and ways to define methods. IDL does not specify an implementation, it is only used to define how two applications shall communicate [30]. CORBA uses the IIOP protocol to send objects over the network [30].

2.3.3 Java RMI versus CORBA

CORBA is a good choice when multiple programming languages are used, which often is the case when integrating a new application into an already existing system. Java RMI can only be used in an environment that includes applications implemented in Java.

When building a complex system using Java, Java RMI is easier to use. The
amount of code that has to be written when using Java RMI is significantly smaller than corresponding code that has to be written when using CORBA. Because Java RMI uses serialization, it is easy to evolve and maintain a Java RMI application. Furthermore, Java RMI automatically manages distributed garbage collection, there is no such thing in CORBA.

If most of the code in a distributed system is implemented in Java, but a small part of the code is implemented in some other language, it might seem as CORBA is the only solution. This is not true, CORBA can be used on top of Java RMI. This approach will allow the programmer to use Java RMI as usual when only Java applications are communicating. When a java application shall communicate with an application written in another language the RMI/HIOP can be used. Using RMI/HIOP is similar to using Java RMI, but there are however some limitations. The most important restriction concerns Java RMI interfaces that pass objects, either as arguments or as return values. Only the most recent implementation of CORBA can be used to pass objects, and the only other language that can be used is C++. If no objects are passed there are no major restrictions when using RMI/HIOP in conjunction with Java RMI [30].

2.4 Markup Languages

Many types of hardware exists, for example PCs, Sparc stations and Macintoshes. Different operating systems run on these systems. When differences among software are added with differences among hardware and operating systems, there are thousands of possible output types.

Generalized markup provides machine-readable style and format markup that is not specific to any one machine. It imposes a general structure on documents and it handles all documents according to that structure [31].

2.4.1 SGML

Standard Generalized Markup Language\footnote{ISO 8879:1986} (SGML) is a standard set of rules for defining document types by their structures and for marking them up so that machines can recognize and process documents by those structures [31]. SGML constitutes a generic document meta language, a language for describing languages rather than a language for representing documents [32]. As a meta language, it defines syntax rules for document elements but does not define any specific elements [33].

Using the rules of SGML an infinite set of specific markup languages can be created, according to individual document needs [31]. A specific markup language has a specific vocabulary (labels for elements and attributes) and a declared syntax (grammar defining the hierarchy and other features).

Each markup language implemented using SGML, is defined in a Document Type Definition (DTD). The DTD is not always included with the document in-
2.4 Markup Languages

Extensible Markup Language (XML) is a fully compatible simplified subset of SGML that removes a large number of the more complex and sometimes less-used features which make SGML difficult and expensive to implement [35]. XML is defined as an application profile of SGML. This means that any fully conformant SGML system will be able to read XML documents. However, using and understanding XML documents does not require a system that is capable of understanding the full generality of SGML. XML is, roughly speaking, a restricted form of SGML [36]. A document which follows the syntax rules of XML is also following the syntax of SGML [34].

Like SGML (and unlike HTML), XML does not have a fixed set of elements that are always supposed to work. It allows developers to define the elements they need, as they need them [35]. Anyone is free to mark up data in any way using the language, even if others are doing it in completely different ways. This is why it is called Extensible Markup Language [34]. Like an SGML document, an XML document is valid if there is a DTD associated with it and if the document complies with that DTD [37]. There are several XML applications, for example the Extensible Stylesheet Language XSL (described below) and the HTML replacement language XHTML [33].

All XML documents are trees. There are no particular restrictions on how nodes are ordered, how nodes connect and to which other nodes they connect. There is however, always one root node which represent the document itself [35]. For example, if the document is a book the root element is book. If the document is an article, the root element is article. Below is an example that shows a sample
XML document that contains data about a purchase order [38].

```xml
<?xml version="1.0"?>
<purchaseOrder orderDate="1999-10-20">
  <shipTo country="US">
    <name>Alice Smith</name>
    <street>123 Maple Street</street>
    <city>Mill Valley</city>
    <state>CA</state>
    <zip>90952</zip>
  </shipTo>
  <billTo country="US">
    <name>Robert Smith</name>
    <street>8 Oak Avenue</street>
    <city>Old Town</city>
    <state>PA</state>
    <zip>95819</zip>
  </billTo>
  <comment>Hurry, my lawn is going wild!</comment>
  <items>
    <item partNum="872-AA">
      <productName>Lawnmower</productName>
      <quantity>1</quantity>
      <USPrice>148.95</USPrice>
      <comment>Confirm this is electric</comment>
    </item>
    <item partNum="926-AA">
      <productName>Baby Monitor</productName>
      <quantity>1</quantity>
      <USPrice>39.98</USPrice>
    </item>
    <shipDate>1999-05-21</shipDate>
  </items>
</purchaseOrder>
```

The Extensible Style Language (XSL) is a style-sheet language for XML. In simple terms, a style sheet contains instructions that tell a processor (such as a Web browser, print composition engine, or document reader) how to translate the logical structure of a source document into a presentational structure. Because there is no underlying semantic to augment for XML, XSL specifies how each element should be presented and what the element is. XSL defines not only a language for expressing style sheets, but also a vocabulary of “formatting objects” that have the necessary base semantics [39]. XML is popular for many reasons, some of the most important reasons are summarized below.
2.4 Markup Languages

**Easy Data Exchange**

Data and markup are stored as text that can be configured. XML editors can be used to edit XML documents, but if something goes wrong documents can be edited directly because it is all just text. The data is not encoded in some way that has been patented or copyrighted, so it is accessible. In addition, when markup languages are standardized, many different people can use them [37].

**Customizing Markup Languages**

Customized languages can be created using XML. If a number of people agree on a markup language, customized browsers or applications can be created, that handle that language. Hundreds of such languages already are being standardized now. Furthermore, if a markup language is created using XML, extensions to this language can be added easily [37].

**Self-Describing Data**

The data in XML documents are self-describing. Based on the names given to each XML element, it is possible to figure out what the document describes. XML documents are, to a large extent, self-documenting [37].

**Structured and Integrated Data**

XML lets users specify not only data, but also the structure of the data and how various elements are integrated into other elements. The emphasis on correctness of documents is strong in XML. In HTML a Web author can (and frequently does) write sloppy HTML, knowing that the Web browser will take care of any syntax problems. In XML, browsers are supposed to check documents; if there is a problem they are not supposed to proceed any further [37].
Chapter 3

Method

This section describes the methods and techniques used for achieving the purpose of project. The purpose is described in section 1.2, it is divided into four sub-goals.

3.1 Main Features Model

A Main Features Model draft and a questionnaire covering fundamental design issues was developed and sent to a representative of the Louvre Museum in Paris, France and to a representative of the National Gallery in London, UK.

A meeting was arranged with a local museum in Linköping, to discuss the appropriate properties of and the need for ARTSY.

3.2 Security Analysis

To analyze the security threats in the context of ARTSY, literature in the areas of cryptography, distributed systems and computer security was studied. To further assemble relevant information Internet articles were read. To locate relevant Internet articles the Google search engine\(^1\) was used. Some keywords served as the starting point for these searches. The most important keywords are listed below.

- Key Establishment Protocols
- Key Management Techniques
- Authentication
- Digital Signatures

The method used to analyze key establishment protocols chosen was the *ad hoc and practical analysis* [1].

\(^1\)http://www.google.com (31st January 2003)
Viiveke Fääk, assistant professor and a computer security expert at the Department of Electrical Engineering at Linköping University was consulted. An ARTSY security model was developed based on the results from the security analysis and oral feedback on this model was given from professor Fääk.

3.3 Software Requirements Specification

A Software Requirements Specification was developed and structured according to the IEEE830-1998 standard\(^2\).

3.4 Prototype

An ARTSY prototype was designed using the JOOM model [40]. The method used for implementing the prototype was inspired by the eXtreme Programming [41] paradigm. The following practices\(^3\) were adopted from eXtreme Programming:

- Continuous Integration
- Iterative Development
- Spike Solutions

Java was used as the programming language and GNU EMACS\(^4\) was used as the programming environment. JavaDoc\(^5\) generated documentation was used as the starting point for the Users Guide.

---

Chapter 4

Result

This chapter presents the results of the project. The purpose of the project is divided into four parts (see section 1.2), this chapter is therefore divided into four sections. In chapter 5 the results as well as the methods used for achieving them are discussed.

4.1 Main Features Model

The questionnaire that was sent to two museums was answered by one museum. The attached information about ARTSY had not been read by the contact at the museum and the answer to the questionnaire was therefore not of any use. The meeting that was arranged with a local museum in Linköping, was cancelled by the museum the day before the meeting. The stated reason was lack of time. Below is a description of the Main Features Model that was developed.

\[
\begin{align*}
\text{RC} & \quad \rightarrow \quad \text{CP} \\
\text{TS} & \quad \rightarrow \quad \text{RC} \\
\text{TS} & \quad \rightarrow \quad \text{CP}
\end{align*}
\]

Figure 4.1. An RC orders a Content Representation
ARTSY is based on three types of components: Content Providers (CPs), Transaction Servers (TSs) and Reproduction Clients (RCs). A CP offers Content Representations and an RC reproduces them. A TS mediates transactions of Content Representations. In figure 4.1 one RC starts a transaction by ordering a Content Representation. The TS registers the transaction and informs the CP which then transmits the Content Representation to the RC.

There can be many CPs, TSs and RCs. A CP is bound to one TS in the sense that it only communicates with one TS. An RC is independent and is not bound to any TS, it communicates with different TSs at different times. Figure 4.2 gives an overview of a small ARTSY.

![Diagram of ARTSY](image)

**Figure 4.2. A small ARTSY**

The main task of ARTSY is to permit secure transactions over an insecure media, such as the Internet. The transaction procedure normally includes three steps where the actual transaction of a Content Representation constitutes the last one:

1. Information Request
2. Tender Submission
3. Transaction
4.1.1 Information Request

During an Information Request an RC requests information from a CP on some Content that the RC might want to reproduce. When the CP receives the request it responds by sending an Information Response back to the RC, containing the requested information. Information in this case means meta data that describes the Content. If an image is to be transmitted the meta data contains a description of the image, for example its resolution, its physical size, its color depth etcetera. Figure 4.4 shows a Content Meta Data file from an Information Response that describes an image of Mona Lisa by Leonardo Da Vinci. This image has two Content Representations, one in jpeg format and one in tiff format. Apart from being encoded differently these two representations also have different resolutions.

![Diagram](image)

Figure 4.3. Information Request

The TS connected to the CP is not involved in the Information Request. It is therefore not possible to specify the transaction cost in the Information Response.

The purpose of the Information Request is to provide an RC with relevant Content information, making it possible for the RC to decide whether to continue or not with the transaction. Critical Content information is for example the available encoding formats. There is no point in ordering a Content Representation that cannot be reproduced since its format is not supported. The Information Response informs the RC what it can order. Apart from the available encoding formats the response might also contain information about alternative representations of the Content. An alternative representation of a color image could for example be a gray scale image. Another alternative representation of an image could be a sub area of the image. The Information Request is optional and an RC does not have to issue an Information Request before inviting a Tender.

The messages that are exchanged (see figure 4.3) are listed below:

1. The RC sends an Information Request to the CP.

2. The CP responds to this request by sending an Information Response back
to the RC.

CMD:
CP-ID: cp1
CONTENT:
ID: 8
TITLE: Portrait of Mona Lisa (1479 – 1550)
ARTIST: Leonardo Da Vinci
DESCRIPTION:
YEAR:
START: 1503
END: 1506

TECHNIQUE: Painted in oil
SIZE:
WIDTH: 530
HEIGHT: 770

CONTENT-REPRESENTATION:
ID: 1
DESCRIPTION:
TECHNIQUE: Photograph – direct reflecting light
FORMAT: jpeg
RESOLUTION:
X: 512
Y: 384

CONTENT-REPRESENTATION:
ID: 2
DESCRIPTION:
TECHNIQUE: Photograph – direct reflecting light
FORMAT: tiff
RESOLUTION:
X: 6400
Y: 4800

Figure 4.4. An Information Response describing an image
4.1.2 Tender Submission

During a Tender Submission a Tender is invited and submitted. Three components are involved in the procedure: An RC that invites the Tender, a CP that submits a pre-Tender and eventually the TS connected to the CP that creates the Tender based on the pre-Tender. The Tender Invitation contains a request for a specific Content Representation. It specifies the requested encoding format and it can also specify other things. If an image is to be transmitted the Tender Invitation could for example declare that only a sub area of the image shall be transmitted, or a Content Representation with a reduced color depth.

![Diagram of Tender Submission](image_url)

**Figure 4.5.** Tender Submission

The purpose of the Tender Submission is to provide an RC with a Tender. The Tender contains a price for the Content Representation requested in the Tender Invitation. This price is the total price expressed in the currency of the RC. The total price is the sum of the fee that the TS would charge for managing the transaction and the fee that the CP would charge for transmitting and letting the RC reproduce the Content Representation. Given this information the RC decides whether to order the Content Representation or not. If the RC decides to make an Order, the RC must use the Tender as a starting point for the Order. Figure 4.6 shows an example of a Tender.

The messages that are exchanged (see figure 4.5) are listed below:

1. The RC sends a Tender Invitation to the CP.
2. The CP compiles a pre-Tender and submits it to the TS.
3. The TS creates a Tender and submits it to the RC.
TENDER:
TENDER-ID: 225
PRE-TENDER-ID: 213
TENDER-INVITATION-ID: 240
VALID-UNTIL: Monday, February 3, 2003 10:52:03 AM MET
CONTENT:
   ID: 8
   TITLE: Portrait of Mona Lisa (1479 - 1550)
   ARTIST: Leonardo Da Vinci
CONTENT-REPRESENTATION:
   ID: 1
   DESCRIPTION:
   TECHNIQUE: Photograph - direct reflecting light
   FORMAT: jpeg
   RESOLUTION:
   X: 512
   Y: 384

COPIES: 1
RC-ID: rcl
RC-HOST: bullock.isy.liu.se
RC-CURRENCY: sek
CP-ID: cp1
CP-HOST: bullock.isy.liu.se
CP-FEE: 3
CP-CURRENCY: eur
TS-ID: ts
TS-HOST: bullock.isy.liu.se
TS-CURRENCY: sek
TS-FEE: 10
TOTAL-FEE: 37.06
TOTAL-CURRENCY: sek

Figure 4.6. A Tender
4.1.3 **Transaction**

The transaction phase starts when an RC sends an Order to a TS. The Order is based on a Tender which must have been received from a CP before the transaction. When a TS has received an Order the TS sends one Transaction Start message to each client. The CP starts to transmit a Content Representation to the RC when it receives its Transaction Start message.

![Diagram of Transaction - Step 1](image1)

**Figure 4.7. Transaction - Step 1**

When the CP has transmitted the entire Content Representation (or tried to), it sends a feedback message to the TS. The RC correspondingly sends another feedback message to the TS when it has reproduced (or tried to) the entire Content Representation.

![Diagram of Transaction - Step 2](image2)

**Figure 4.8. Transaction - Step 2**

A feedback message contains one of the following: Ok or Failed. The TS decides the result of the transaction given the feedback messages from the RC and the CP.
The TS then informs the clients of the result by sending a Transaction Result message to each of them.

The result of the transaction result is one of the following: Completed, Cancelled or Corrupted. If the result is Completed the RC can be charged for the transaction. If the result is Cancelled the RC can not be charged for the transaction. A Corrupted transaction must be manually investigated further to decide whether the transaction should be Completed or Cancelled.

How different feedback messages give rise to different transaction results is shown in table 4.1. As long as the RC sends Ok the transaction result will be Completed. If a Failed message is received from the CP and no message has been received from the RC the transaction will be Cancelled. The situation where the RC reports Failed and the CP reports Ok is unlikely to occur. The CP reports Ok when it is sure that the Content Representation has been received correctly by the RC. If the RC reports Failed it means that something happened after the Content Representation was received but before it was reproduced. This must be investigated further before it can be decided whether to charge the RC for the transaction or not. The purpose of the Transaction is to transmit a Content Representation from a CP to an RC.

<table>
<thead>
<tr>
<th>Message from RC</th>
<th>Message from CP</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td>OK</td>
<td>Completed</td>
</tr>
<tr>
<td>OK</td>
<td>FAILED</td>
<td>Completed</td>
</tr>
<tr>
<td>FAILED</td>
<td>OK</td>
<td>Corrupted</td>
</tr>
<tr>
<td>FAILED</td>
<td>FAILED</td>
<td>Cancelled</td>
</tr>
<tr>
<td>-</td>
<td>FAILED</td>
<td>Cancelled</td>
</tr>
</tbody>
</table>

Table 4.1. Transaction result

The messages that are exchanged (see figure 4.7 and 4.8) are listed below:

1. The RC sends a request for the selected content to the TS.
2. The TS sends a Transaction Start message to the RC.
3. The TS sends a Transaction Start message to the CP.
4. The CP sends the selected content to the RC.
5. The CP sends feedback to the TS when the entire Content Representation has been transmitted.
6. The RC sends feedback to the TS when the entire Content Representation has been received.
7. The TS informs the clients of the result of the transaction by sending a Transaction Result message to each of them.
4.2 Security Analysis

The result from the security analysis is a list of relevant security threats and a security protocol that prevents those threats.

4.2.1 Security Threats

There are mainly three entities that can threat the communication in ARTSY. Two inside threats: The RC and the CP, and one outside threat: An adversary that has access to the Internet. The RC becomes a threat if it tries to reproduce Content Representations without getting charged. The CP becomes a threat if it tries to charge RCs for Content Representations that have not been transmitted. An adversary can threat ARTSY either by gaining access to Content Representations or by disturbing the communication between two ARTSY components. Below is a list of possible security attacks.

Interruption

1. An adversary could perform a Denial Of Service attack against an ARTSY component and thereby blocking the network for the component.

Interception

1. By eavesdropping an unencrypted Content transmission an adversary could gain access to a Content Representation.

2. The RC software does not store a received content transmission locally. Personnel at an RC could however eavesdrop a Content transmission, record it, store it and then use it to make illicit copies.

Modification

1. The RC software could be modified so that Content Representations that have been received are stored. This would make it possible to reproduce a Content Representation an unlimited number of times.

2. The RC software could be modified so that it always sends the feedback message FAILED, even if Content Representations are reproduced properly. This means that the RC never is charged for making reproductions.

3. The CP software could be modified so that it always sends the feedback message OK, even if the transmission of a Content Representation fails.

4. An adversary could perform a man in the middle attack by changing or reusing an Order message from an RC so that the RC is charged for a Content Representation that it has not ordered.
Fabrication

1. An adversary could use the identity of an RC and send Orders. The RC will then be charged for Content Representation it has not ordered.

2. An adversary could use the identity of a TS and send Transaction Start messages. This will make a CP transmit a Content Representation that the adversary could access.

3. An RC could make an Order and then claim that it has not.

4. An adversary could eavesdrop an Order and the replay it. Each time the Order is replayed the RC that made the real Order will be charged again.

An additional threat is that a Reproduction could be copied after the Reproduction process is completed. A physical image could for example be copied after it has been printed.

4.2.2 Security Protocol

As stated in section 4.2.1 ARTSY is threatened by interruption, interception, modification and fabrication attacks. To prevent these attacks the following services are required: Availability, confidentiality, integrity and authenticity. The security protocol of ARTSY provides all services except interruption by using several security mechanisms. Two types of encryption keys are used in ARTSY: Asymmetric keys that are used to sign and encrypt messages, and symmetric session keys that are used to encrypt Content Representations. Each type of key uses its own key establishment protocol, the protocols are presented below.

Most of the security mechanisms are used on messages. Different parts of ARTSY messages are encrypted differently. By modifying the Main Features Model and splitting the RC software into two separate modules the security in the system is further enhanced. Some parties might want to participate in ARTSY without using encryption. ARTSY allows this by using different levels of security. The division of the RC component into two modules and the security level concept is described further below.

ARTSY Messages

ARTSY components communicates principally by passing messages to each other. An ARTSY Message has the following structure:

\[
\text{Information} | \text{[Extra]} | \text{[Signature]}
\]

Message

\text{Signature} (within brackets) is used in some Messages. The parts without brackets are used in all messages. \text{Signature} is the digital signature of the sender and is used for authentication in all secure Messages. \text{Information} contains information
4.2 Security Analysis

about the message, among other things the message type. Some message types that are used are: Transaction Feedback, Transaction Start and Order. Other data that is included as Information is for example pre-Tender documents and Tender documents. Information can also contain a reference to a file or a document. Information in a Transaction Start message could look like this:

```
TRANSACTION_START t300 4E6F772069732074
TS,tokenid session_key
```

Extra is anything that needs to go unencrypted. This might be necessary when some encryption modes are used or for other implementation specific reasons.

Security Mechanisms

The following security mechanisms are used in ARTSY:

- **Asymmetric cryptography**: All messages are encrypted using asymmetric keys — this provides confidentiality of messages.
- **Digital signatures**: All messages are signed and verified — this provides integrity and authenticity of messages.
- **Symmetric cryptography**: Session keys are used when encrypting Content Representation — this provides confidentiality of Content Representation.
- **Public Key Registry**: A public key registry is used to store and make public keys available for download.

The messages that are sent during an Information Request are not encrypted nor signed. All other messages are encrypted using the public key of the receiver. All entities is in possession of two asymmetric key pairs. One pair is used for encryption/decryption and the other pair is used for creating and verifying digital signatures. When sending a message an entity uses the public encryption key of the receiver to encrypt the Information part of the message and its own private signature key to sign the Information and the Extra parts.

```
Sender + Asym[Information]Receiver Public + Asym[Hash(data)]Sender Private
```

When a message is received the receiver verifies the message with the public signature key of the sender and decrypts the Information part of the message using its own private decryption key. Figure 4.9 shows how the security mechanisms are used when an Order message is sent from an RC to a TS.

It is suggested that every client must sign a contract before it can participate in ARTSY. Such a contract should cover security aspects that cannot be implemented in software. It should for example explicitly forbid an RC from making copies
of posters that have been produced using ARTSY. The security protocol cannot protect ARTSY from interruption attacks. ARTSY depends on the hosts that it resides on. By protecting all hosts from interruption attacks, ARTSY will also be protected.

**Asymmetric Key Establishment Protocol**

All ARTSY messages are secured using asymmetric keys. These keys must be distributed before they can be used. A key pre-distribution transport protocol is used to distribute asymmetric keys. The protocol requires a trusted Public Key Registry. One of the TSs in ARTSY must act as a Trusted Party and must host this Public Key Registry. Each Component that connects to ARTSY creates its own public-private key pair at installation. The Public key of the Component is then delivered to Trusted Party along with the unique id of the Component. Each public key is signed by the Trusted Party. When a component needs to send a secure message to another component it contacts the Trusted Party to acquire the public key of the receiver. The public key of the Trusted Party is distributed at installation. Figure 4.10 shows how an RC and a CP acquire each others public keys when a Tender Invitation is sent.
Instead of using a Public Key Registry, certificates and a Certificate Authority can be used. If certificates are used the public key of a component is included in a certificate. All components would then have their own certificate. The CA would maintain a certificate database and make all certificates available to all components. The CA could be an integrated part of ARTSY or a separate CA, for example VeriSign\(^1\), could be engaged.

**Symmetric Key Establishment Protocol**

To establish a session key a dynamic key transport protocol based on asymmetric techniques is used. The session key is generated by the TS using a Pseudo Random Number Generator that is fed with a truly random seed. The session key is then sent to an RC and a CP in the *Information* part of a Transaction Start message. Since it is transmitted as *Information* the session key is encrypted with the public key of the receiver. The message is also signed by the TS so that the receiver can verify the origin of the message. When the Transaction Start messages are received by the CP and the RC the session key is established.

**The Inner and Outer Module of an RC**

The RC component is divided into two modules: The RC outer module (RCo) and the RC inner module (RCi). The RCo handles the network traffic and user interaction while the RCi manages the production of reproductions. The RCi is located inside the reproduction unit (for example a printer), while the RCo is located on a

\(^1\)http://www.verisign.com
host computer just as the CP and TS components. Locating the RCi inside of the reproduction unit makes it possible to transmit encrypted data all the way from the CP to the reproduction unit. Sensitive data will thus stay encrypted as long as it is possible. Decryption will take place inside of the reproduction unit and not on the host computer where the RCo is located. If the RC would not be divided into two modules, the decryption would have to be managed by the host computer. If data would be decrypted on the computer rather on the reproduction unit, it would have to be transmitted from the computer to the reproduction unit unencrypted, without protection. Such a transmission could be recorded, manipulated and re-used in order to produce illicit reproductions.

**Each Tender is Only Used Once**

Each Tender can only be used for making one Order. Tenders cannot be re-used to make multiple Orders of the same Content Representation. This prevents a man in the middle attack (modification attack number four, see section 4.2.1).

**Security Levels**

Each ARTSY component chooses a security level to support. There are two levels to choose from: The *secure security level* and the *insecure security level*. The security protocol, described in section 4.2.2, is used for components that support the secure security level. At the insecure security level no security at all is used (as in the original Main Features Model, see section 4.1). All components that support the secure security level automatically also supports the insecure. Those components can thus communicate with all other components. Components that only support the insecure security level, on the other hand, cannot communicate with components that require full security. A component that supports the secure security level can choose to temporary degrade to the insecure security level to be able to communicate with components that only support this level. All TSs must support the secure security level. It is only possible to charge money for a transaction if all involved components support the secure security level. Table 4.2 summarizes the consequences of using different security levels.
4.3 Software Requirements Specification

A complete version of the Software Requirements Specification can be found in appendix A.

4.4 Prototype

An ARTSY prototype has been implemented using Java. The source code of the
The prototype can be downloaded from http://www.isy.liu.se/cvl/ScOut/Masters/PaperInfo/artsy2003.html. A detailed documentation of the prototype can be found in appendix B. Figure 4.11 shows a screenshot of the prototype.

The prototype meets the prototype requirements specified in the ARTSY Software Requirements Specification (SRS), see appendix A. The SRS presents two sets of requirements: a full set of requirements that apply to a full scale ARTSY implementation and a reduced set of requirements that apply to an ARTSY prototype implementation. The latter is a subset of the former. The most important consequences of the reduced set of requirements for the prototype implementation compared with the full set of requirements for the full scale implementation are presented below.

1. The prototype does not implement or use a public key registry or a certificate authority.
2. The prototype does not support multiple security levels.
3. The prototype does not support more than one TS.
4. The RGi module is not inserted into a printer.

![Diagram](image-url)

**Figure 4.12.** Remote method invocation during a Tender Submission

The prototype utilizes Java RMI technology to enable communication between its components. Each component acts as both a Java RMI server and a Java RMI client. This is necessary since two-way communication is needed between the components. Figure 4.12 illustrates how remote methods are invoked during
a Tender Submission\textsuperscript{2}. XML documents are used to store textual information in messages that are transmitted over the network.

4.4.1 Dependencies

The Java 2 Platform, Standard Edition 1.4.1 (J2SE) was used to create the prototype. J2SE must be available on the computer(s) where the prototype is to be used. Two third party java packages must also be available: The Bouncy Castle package, used for RSA encryption, and the JDOM package, used for generating and parsing XML documents. JDOM version 8 and Bouncy Castle version 1.15 is used in the prototype. minSQL is used as the DBMS. minSQL has been tested on the Sun Solaris platform (SunOS release 5.8). The authors of minSQL claim that it works with the OS/2, Win9x and WinNT platforms as well. This has not been verified. In the prototype minSQL version 3.0 is used. The minSQL source code is not available for download from http://www.isy.liu.se/cvl/ScOut/Masters/PaperInfo/artsy2003.html since it requires a personal registration.

If the prototype is used on a Sun Solaris platform it uses the Solaris system file /dev/random as input seed when generating encryption keys. The prototype can be used on other platforms as well, but it will then use the less secure built-in pseudo random number generator in Java.

4.4.2 Code Structure

The prototype Java source code is divided into five packages:

- artsy.rco
- artsy.rci
- artsy.cp
- artsy.ts
- artsy.common

ARTSY is based on four different types of components: The RCo, the RCi, the CP and the TS. The first four packages contain component-specific classes for the four component types. artsy.rco contains classes needed by the RCo component and so forth. The last package, artsy.common, contains common classes (shared classes) that are needed by all components. Some of the classes in the artsy.common package are superclasses with corresponding subclasses in the other packages. Other classes in artsy.common are data classes that must be known to all components since they are passed between the components in order to exchange information. Below is a concise description of the most important classes.

\textsuperscript{2} "Network" is the name of the method that is invoked remotely
Common Classes

The most important classes contained in the *artsy.common* are:

Message
This data class holds all the data that is needed in an ARTSY message. The class is used when messages are passed over the network. There are no subclasses of this class.

MessageHandler
This superclass contains generic methods for message handling. All components have their own MessageHandler, a subclass of this class. The local MessageHandler subclass at each component handles incoming as well as outgoing messages. Incoming messages are received from the Network class and interpreted. Depending on the message type different methods are then called. The contents of an incoming message are often passed over to the local UserInterface subclass for viewing. Outgoing messages are created by the MessageHandler subclass and then passed on to the Network class.

Network
This class handles all network accesses. Each component create one instance of this class. There are no subclasses. All outgoing messages are transmitted from the local instance of this class to a remote one. Incoming messages are correspondingly received from a remote Network instance and handled by the local Network instance.

Security
This superclass includes generic methods for encryption and decryption. All components have their own Security class, a subclass of this class. All incoming messages are processed by the local Security subclass before they are passed to the local MessageHandler subclass. All outgoing messages are processed by the local Security subclass before they are transmitted over the network by the Network class. The main tasks of the Security subclasses are to decide how to encrypt/decrypt different messages and then decrypt incoming messages and encrypt outgoing messages.

UserInterface
This superclass includes generic methods for displaying information and for receiving instructions from the user. All components have their own UserInterface class, a subclass of this class. The user interface is totally separated from the rest of the code so that minimal effort is needed to replace it. Depending on how this class is implemented the user interface can be either graphical or textual. The current implementation provides a rather simple graphical user interface.
4.4 Prototype

Component-Specific Classes

The structures of the classes in the component-specific packages (artsy.rco, artsy.rei, artsy.cp and artsy.ts) are quite similar. All components have own subclasses of the MessageHandler, Security and UserInterface superclasses as described in section 4.4.2. Besides these subclasses all components also have a main class. The main method of the main class is the first method to be executed when a component is started. It instantiates all fixed classes\(^3\) that the component will use and then terminates (and thereby puts the main class into an inactive state).

In addition to the classes described above every package also contain some extra classes that are not needed in all packages. Among the extra classes are for example classes for handling database accesses, for creating specific XML documents and for printing documents.

4.4.3 Security

Messages are either secure or insecure. All secure messages are encrypted and signed in the same way. Insecure messages are never encrypted nor signed. Only one of the two key establishment protocols is implemented in the prototype: The symmetric key establishment protocol. All public keys that will by used by any Component (for asymmetric encryption) must be available locally at the Component in order for the prototype to work.

Secure Messages

A secure message is encrypted and signed by the sender and verified and decrypted by the receiver. An ARTSY message is structured in the following way:

\[
\text{Message} \begin{array}{c}
\text{Information} \\
\text{[Extra]} \\
\text{[Signature]}
\end{array}
\]

There are two Java classes that form a message: Message and Information. The Message class is described in section 4.4.2. Every instance of the Message class contains one instance of the Information class. The Extra part is represented as attributes in the Message class. When an ARTSY message is secured the Information object contained in a Message object is encrypted. The entire Message object is then signed and a SignedObject (see section 2.2.2, page 25) is thus produced. The SignedObject is then passed over the network. The implemented procedure complies with the security protocol described in section 4.2.2: The Information part is encrypted, the Extra part is not encrypted, the entire message is signed and the Signature part is sent along with the message. In section 4.2.2 the Information is signed before it is encrypted but in the prototype the Information is signed after it is encrypted.

\(^3\)Non-data classes which are created once and not terminated until the component is shut down.
Confidentiality is granted by using RSA encryption. The Information object is encrypted and decrypted using the RSA algorithm in CBC mode and with PKCS#1 padding. The RSA algorithm used is the one provided by Bouncy Castle (the org.bouncycastle.crypto.engines.RSAEngine class is used). It is used via the Java Sun Provider architecture. Originally it is only possible to use this algorithm in the ECB mode. The ARTSY Security class (described in section 4.4.2) does however implement CBC chaining (on top of the Bouncy Castle RSA algorithm) so that each block that is encrypted is chained with the previous encrypted block. The CBC chaining requires an initialization vector (IV). A new IV is created before the encryption of each Message. The IV is created by using the SecureRandom class (described in section 2.2.2, page 24) that uses the SHA-1 algorithm. The SecureRandom is not seeded by ARTSY, it seeds itself.

Authentication is granted by using the SHA-1 hash algorithm. Each Message is signed using the SignedObject class. The hash algorithm used is SHA-1 and the encryption algorithm used is RSA. The input key is the private key of the signing Component.

The private public key pair used in ARTSY is generated using the java.security.KeyPairGenerator. The algorithm used is the key pair generation algorithm described in PKCS#1. The length of the RSA keys generated is 1024 bits. The prime numbers are generated using the class SecureRandom. If the prototype is used with the Sun Solaris platform, SecureRandom is fed with a seed that is generated from the system file /dev/random.

Secure Content Representations

Content Representations are encrypted symmetrically. Confidentiality is granted by using DES encryption. The java.crypto.Cipher class is used with the SUN JCE implementation of the DES algorithm using 64 bits block size in ECB mode and with PKCS5 Padding. DES is implemented as described in FIPS PUB 46-2.

To generate the session keys the java.security.KeyFactory is used with the SUN JCE DES algorithm and a keysize of 56 bits. The java.security.KeyFactory is initialized using the class SecureRandom. If the prototype is used with the Sun Solaris platform, SecureRandom is fed with a seed that is generated from the system file /dev/random

Symmetric Key Establishment Protocol

The symmetric key establishment protocol is implemented in the prototype. Session keys are created by the TS and transmitted to the clients in two messages: The CP Transaction Start message and the RC Transaction Start message. In the information part of the RC Transaction Start message there is an enclosed message intended for the RCI. The session key is included in this message which is not readable to the RCo.
4.4 Prototype

4.4.4 Flowcharts of the Code

This section illustrates the main features of ARTSY using flowcharts. The flowcharts only include objects of such classes that are unique to ARTSY (except for the CipherInputStream class in figure 4.17). The symbols used in the flowcharts are explained below.

- **Boxes** Boxes represent instances of classes (objects). There are only one object of each box type at each ARTSY Component. These objects are the main building blocks in the prototype.

- **Ellipses** Ellipses represent instances of classes (objects). There can be more than one object of each ellipse type at each ARTSY Component.

- **Numbers** The flow starts at number one and then continues with number two and so forth.

- **One way arrows** One way arrows indicate that the control is transferred from the object where the arrow starts to the object where the arrow ends.

- **Two way arrows** Two way arrows indicate that the object where the arrow was encountered invokes a method on the object on the other side of the arrow. The control is returned to the invoker when the second object has finished processing the invocation.

**Information Request**

![Flowchart of Information Request]

A user starts an Information Request via the UserInterfaceRC class (see figure 4.13). Arrow number 4 shows that the SecurityRCo class is called, even though
the Information Request is not encrypted. SecurityRCo keeps track of the messages that shall be encrypted and those that shall not. All messages that are sent from the RCo are processed by the SecurityRCo class. At arrow number 8 the user at the CP is notified that an Information Request has been received. A Content Meta Data (CMD) file is fetched from the database and sent to the RCo. The CMD is displayed at arrow 16 after it has been parsed and formatted by the XMLParser class.

**Tender Submission**

![Tender Submission Diagram](image)

**Figure 4.14. Tender Submission**

A user at the RC starts a Tender Submission via the UserInterfaceRC class (see figure 4.14). At arrow 3 the TenderInvitationFactory class is called. This class creates a Tender Invitation document and stores it in the database (arrow 4). The Tender Invitation is put in a message at arrow 5 and sent to the Network class. At arrow 7 the Tender Invitation is encrypted and signed. When the Tender Invitation has been received at the CP it is decrypted (arrow 9), parsed (arrow 13) and used for creating a pre-Tender document. The pre-Tender document is stored in the database (arrow 14) and sent to the TS after it has been encrypted (arrow 17).
The TS receives the pre-Tender, parses it (arrow 23) and uses it to create a Tender document. The Tender document is then stored in the database (arrow 25). The Tender is sent to the RC (arrow 29) and parsed (arrow 33) before it is displayed (arrow 34).

**Transaction Part 1**

Figure 4.15. Transaction part 1: Transaction Start messages are received

A user at the RC starts a Transaction by entering a Tender id (see figure 4.15). The Tender id is sent to the MessageHandlerRCo which acquires the corresponding Tender from the database (arrow 3) and parses it (arrow 4) to check that the Tender is valid. If it is, an Order is sent to the TS (arrow 7). The TS also checks that the Tender is valid by acquiring it from its database (arrow 11) and parsing it (arrow 12). A symmetric session key is generated by the KeyFactory class (arrow 13). Transaction Start messages are sent to the RC and the CP (arrow A). The Transaction Start messages are decrypted at each Component (arrow B) and sent to the MessageHandler classes (arrow C).
**Transaction Part 2**

![Diagram](image)

**Figure 4.16.** Transaction part 2: Ready To Transmit messages are received

In Transaction Part 1 the Transaction Start messages were received by the RCo and the CP. The RCo received two Transaction Start messages in one. One Transaction Start message that was intended for the RCi. The RCi Transaction Start message was included in the RC Transaction Start message and was encrypted with the public key of the RCi. The session key was included in the RCi Transaction Start message (unreadable to the RCo). This is where Transaction Part 2 starts. The RCo forwards the attached Transaction Start message to the RCi in figure 4.16 (arrow E). The RCi decrypts it (arrow F) and stores the received session key.

The next thing that happens is that the CP attaches the Content Representation to a RemoteStream (arrow 1, 2 and 3) and registers the RemoteStream at the RMIRegistry. The Content Representation is now available for download and the CP sends a Ready To Transmit message including the name of the RemoteStream (arrow 6). The Ready To Transmit message is received at the RCo and a LocalStream is created and bound to the RemoteStream at the CP (arrow 9 and 10). The
4.4 Prototype

RCo creates a new Ready To Transmit message that contains the name of the RCo RemoteStream. This message is sent to the RCI (arrow 13). The RCI receives the messages and creates a LocalStream which it binds it to the RemoteStream at the RCo. A BasicPrint object is created and bound to the LocalStream. Everything is now set up and the printing of the Content Representation can start.

**Transaction part 3**

![Diagram showing the flow of messages between RCo, RCI, and CP](image)

*Figure 4.17: Transaction part 3: An image is printed*

In all the other parts messages were the only kind of data that was sent between Components. In Transaction part 3 no messages are sent and no new objects are created. Each LocalStream is connected to a RemoteStream at another Component. The Transaction part 3 starts at the Basic Print object (see figure 4.17). The Basic Print objects calls the CipherInputStream (arrow 1). The CipherInputStream calls the LocalStream (arrow 2) which calls the RemoteStream at the RCo (arrow 4). This goes on until arrow 7, where the FileInputStream is reached at the CP. The FileInputStream reads some bytes from the Content Representation (arrow 8) that is stored as a file and returns them to the calling CipherInputStream (arrow 9). The CipherInputStream at the CP encrypts the received bytes with the session key received in Transaction part 1 and returns the encrypted bytes to the calling RemoteStream (arrow 10). The encrypted bytes are returned all the way to the CipherInputStream at the RCI (arrow 11 to 14) where the bytes are decrypted.
using the received session key in Transaction part 2. The bytes are passed to the BasicPrint which sends them to the printer (arrow 16). This procedure is repeated until the entire Content Representation has been read and sent to the printer.
Chapter 5

Discussion

This chapter contains a discussion of the results presented in chapter 4 and the methods presented in chapter 3. The chapter is divided into four sections that correspond to the four parts of the purpose (presented in section 1.2).

5.1 Main Features Model

Some design decisions were made before the Master Thesis project started, those are not discussed below. A description of the initial conditions can be found in section 1.1.

5.1.1 Method

As can be seen in the section 4, ARTSY was received by the contacted museums, with moderate interest. The ARTSY project as a whole, is at an early stage. The ARTSY prototype has showed that it is technically feasible to implement ARTSY, but it is far from being a finished product that could enter the commercial market. If the prototype were to be transformed into a commercial product, a substantial amount of time would have to be invested in designing a better user interface and in making ARTSY more user friendly. It is important to point out that the ARTSY prototype and a possible commercial ARTSY product are two different programs with totally different exteriors. It is also important to understand the difficulty in engaging possible future customers when there is not even a finished prototype to show to those customers.

The IT maturity at organizations that do not primarily work with computers varies. Low IT maturity does not make a museum less interesting as a subject of an interview, but it could be one of the reasons why it is hard to get the museum interested. It can be difficult to imagine the benefits of a system like ARTSY with limited IT skills.

The ARTSY that was designed in this project is flexible and focus has been put on the techniques that enables ARTSY: security, communication and main features.
A potential user of the system can only interact with the user interface and the Content Meta Data files. The user interface of the prototype is not dependent of the rest of the system and is therefore easy to change. Since all documents in ARTSY are written in XML the information put in the documents is also easy to modify. The conclusion is that even if feedback had been given from potential customers, this would probably have had little effect on the prototype. But ARTSY could not possibly be turned into a product without getting feedback from potential customers.

It was never a primary goal of the project to carry out a market survey, but the proposed system could possibly have been better suited for the future users if some feedback had been achieved during its development.

5.1.2 Result

Every CP that is connected to ARTSY is bound to a specific TS. Every RC that is connected to ARTSY is independent and communicates with different TSs at different times. It would perhaps seem more natural to bind both of the clients and not just one of them. The reason for not binding both of the clients to a TS, is to avoid situations when two clients that are bound to different TSs wish to communicate with each other. If such a situation should arise, two TSs would be involved in the communication instead of one. One of them would have to be selected as the TS responsible for the security. If both TSs were to be responsible at the same time new problems could arise, problems that could be avoided. Suppose for example that there are two TSs responsible for a transaction. When the transaction is over a controversy between the clients comes up, it could be about the price or something else. Both of the clients then consults their own TS for guidance. If the two TSs now tell different things, it is very difficult to investigate what really happened. To avoid situations like this, it is reasonable to select one responsible TS for each transaction. It is also reasonable to do this according to some predictable algorithm. The clients must know, at all times, which TS to consult if problems occur. One predictable algorithm that could be used is to always select the TS that is connected to the CP as the responsible TS. If a problem occurs both the CP and the RC could then consult the TS connected to the CP for guidance. This solution does however make the connection between the RC and its TS pointless. The TS connected to the RC would never be selected as the responsible TS. Removing this connection and thus making the RC independent would not change the system. This is the solution that has been chosen for ARTSY.

ARTSY has three main features: The Information Request, the Tender Submission and the Transaction. They are discussed below.

Information Request

The TS is not involved in the Information Request. In a large system where many CPs are connected to one TS, that TS might become a bottleneck. It is therefore desirable to exclude the TS from as much communication as possible. Since the TS
is not involved it is not possible to define the transaction cost in the Information Response.

The Information Request is optional in the sense that it is possible to start a Tender Submission without having performed an Information Request. In order to start a Tender Submission (to invite a Tender) an id is however needed and the most convenient way of getting an id that can be used is to perform an Information Request. In a future expansion of ARTSY the Information Request as it is currently implemented needs to be replaced by a more flexible and more appealing solution. An adequate enhancement would for example be to let users do all browsing and gather all necessary information through a conventional web browser. If this was possible there would be no need for the current internal Information Request feature. The WWW and standard web techniques provide more efficient and more flexible tools for information visualization than the current built-in solution. Expandability was one of the reasons for making the Information Request optional.

**Tender Submission**

All Tender Submissions go through the TS. Among other things that are specified in a Tender is the price for the transaction. This price is the sum of the fee that the CP charges and the fee that the TS charges. It might seem unnecessary to let all Tender Submissions go through the TS, but unfortunately it is not. If the fee that the TS charges would be predictable it would be. The TS does however have the right to change the fee it charges at any time, so it cannot be predicted.

**Transaction**

An RC cannot order a Content Representation without first having invited a Tender. The Tender Submission is made mandatory because the Tender is an important document. It specifies not only the price of the transaction but also the content of the transaction. If a controversy comes up, for example about the price, the Tender decides who is right. It is therefore of great importance that no Order can be issued without a Tender and that all Tenders are stored until their respective transaction is completed and paid for.

If Content Representations were sent from a CP to an RC via a TS the transmissions could be monitored by the TS. This would make it easier to avoid controversies. In the current implementation the TS is expected to solve possible controversies by means of feedback messages sent from the RC and the CP. The reason for not sending Content Representations via a TS is to enhance the performance of the system. Sending Content Representations via a third party instead of directly from the sender to the receiver would double the transport time. Content Representations are typically represented by relatively large files, network bandwidth is typically small when transmitting over long distances. It is therefore important to minimize the network traffic.
5.2 Security Analysis

5.2.1 Method

The ad hoc and practical analysis method is the most commonly used, but the least satisfying. This method was used in ARTSY because the other methods are very difficult to use. A method that still is a practical method but better than the method used in designing ARTSY is called reducibility from hard problems. This technique consists of proving that any successful protocol attack leads directly to the ability to solve a well-studied reference problem, itself considered computationally infeasible given current knowledge and an adversary with bounded resources. This method is much more difficult than the ad hoc method and relies on the fact that all possible attacks have been taken into account.

5.2.2 Result

The security protocol provides services that will prevent Interception, Modification and Fabrication attacks. An Interruption attack is not prevented in ARTSY. A typical Interruption attack is a DOS-attack, and there are no general ways to prevent such attacks. The best way to protect a system from a DOS-attack is to install a firewall, an anti-virus program and to take similar security actions that has to do with protecting the hosts on which ARTSY resides (see section 2.1.3). This security issue is not a part of this project. The non-repudiation service cannot be provided if the software is altered. But as long as it is intact non-repudiation will be provided.

Threats Prevented by the Security Protocol

The interception attacks described in section 4.2.1 are discussed below.

1. Interception attack number one is solved by encrypting Content Representations with a symmetric session key. This way the eavesdropping adversary will get an encrypted Content Transmission that will take approximately 200 days to decrypt, if the adversary has the skills needed to find the key and do the encryption.

2. Interception attack number two is also solved by encrypting the Content Representation. The RC will have the same problems as the adversary above.

If Content Representations were to be transmitted unencrypted between the RC and the Reproduction Unit the RC would be able to eavesdrop this transmission, record it and use it to make illicit copies. This threat was prevented by dividing the RC component into the RCo and the RCi module. Locating the RCi inside of the reproduction unit makes it possible to transmit encrypted data all the way from the CP to the reproduction unit. If the RC eavesdrops this transmission it will only be able to record an encrypted version of the Content Representation.
Another interception attack could be made if session keys were to be transferred unencrypted from a TS to a CP and an RCo. An adversary would then be able to eavesdrop a Transaction Start message and get an unencrypted session key which could be used for decrypting the Content Representation being sent in the Content Transmission. This attack is however prevented by encrypting the session key.

If the Tender Invitation, pre-Tender, Tender and Order messages would not be encrypted an adversary could get information about a forthcoming Content Transmission and then eavesdrop it. This would make it possible for the adversary to wait for a valuable Content Representation and then spend the time needed to decrypt it. All messages, except the Information Request, are however encrypted and this makes it difficult for an adversary to know when to try to decrypt a Content Representation that is transmitted. An adversary will probably not spend some 200 days trying to decrypt a Content Representation that may be of little value. The modification attacks described in section 4.2.1 are discussed below.

1. Modification attack number one is solved by dividing the RC into two modules. All Content Representations that are stored are encrypted and must be decrypted before they can be reproduced. The RCo does not have access to the session keys that are needed to decrypt them.

2. Modification attack number two is solved by providing Integrity of messages. The TS only accepts feedback messages from the RCI. Since all Messages are signed the TS will discard Feedback messages that are sent from the RCo. The RCo does not have the private key of the RCI and can consequently not use it to sign messages.

3. Modification attack number three is more difficult to prevent than the other. It should be required that each CP party signs a contract before connecting to ARTSY. This contract should explicitly forbid the CP party from modifying the CP software. Such a contract would however not guarantee that the software would not be modified and this is why the TS must keep track of all transaction results and try to detect patterns that indicate that something strange is going on.

4. Modification attack number four is prevented by providing Confidentiality of the Order. The Order is encrypted and the adversary cannot know how to insert a fake reference. Even if he succeeds the TS will discover that the Message has been changed when trying to verify the signature. The TS will not proceed with the Order.

The fabrication attacks described in section 4.2.1 are discussed below.

1. Modification attack number one is prevented by providing Authentication of Order messages. An adversary cannot sign a message with the private key of an RC. A TS that receives an Order where the signature is unverifiable will not proceed with the Order.
2. Modification attack number two is prevented by providing Authentication of all Messages. An adversary cannot pretend to be a TS and send Transaction Start messages. A CP that receives a Transaction Start message where the signature is unverifiable, will not proceed with a transaction.

3. Modification attack number three is prevented by providing Authentication of Order messages. An RC cannot deny an Order since all Orders are signed with the private key of an RC.

4. Modification attack number four is prevented by restricting a Tender from being used more than once. If an Order Message is replayed the TS will discard it.

An RCo could store a Transaction Start message and a Content Transmission. By re-sending these two messages the RCo could then make the RCI reproduce a Content Reproduction that already has been reproduced. This is prevented by letting the RCI keeping track of all Transaction Start messages that it has received. If a Transaction Start message is received a second time the RCI will discard it.

**Security Mechanisms**

ARTSY Messages are encrypted and signed using an asymmetric algorithm. The reason for choosing asymmetric cryptography is that public keys do not have to be secret (see section 2.1.2). If Messages would be encrypted using a symmetric algorithm each Component would have to share a unique secret key with each Component it might need to communicate with, this is called \( n^2 \) key distribution problem.

Any length of an input to a hash algorithm will yield the same length of the output. By using a hash algorithm before producing a digital signature, the length of the signature is limited to the length of the hash value. This has three advantages: It takes less time to sign a hash value of a message than to sign the entire message, less bandwidth is needed when the digital signature is kept small and an adversary cannot use the digital signature in a plaintext attack.

Content Transmissions are the only transmissions where ARTSY messages are not used. An asymmetric algorithm is used to encrypt messages and as stated in section 2.1.2 (page 9) symmetric algorithms are better suited for bulk encryption than asymmetric since they are 100 to 1000 times faster. A typical ARTSY Content Representation is generally a few hundred megabytes large. Encrypting a file this large would take a considerable amount of time if the asymmetric algorithm would be used. This is why symmetric encryption is used for Content Transmissions. Session keys are used to ensure better security (see section 2.1.2).

**Asymmetric Key Establishment Protocol**

The keys used to encrypt messages are established using a Public Key Registry. Certificates could also be used, but the Public Key Registry is chosen because it is easier to implement and still provides the same functionality as certificates do.
5.2 Security Analysis

A Trusted Party (one of the TSs) is trusted to certify each public key and make it available to each component. Since all public keys are signed by the Trusted Party a man in the middle attack cannot be performed (see section 2.1.2).

Symmetric Key Establishment Protocol

The symmetric key establishment protocol provides confidentiality and authentication of the keys established. The protocol provides key authentication since a TS encrypts each session key that is sent in a way so that only the two participating components can decrypt it. Key freshness is guaranteed since the TS always creates a new key for each transaction. A CP cannot use an old key, the RC would then not be able to decrypt the Content Representation (since it will try to decrypt it using a recently received key from the TS). Key control is provided since the TS creates a key and neither the CP nor the RC can predict the key that will be used. The TS uses a truly random seed to create the session key. Non-repudiation (section 2.1.1) is provided, an RC cannot deny having made an Order since all Orders are signed.

A key agreement protocol is a good choice when two parties have not met before and when no previous key material has been distributed. The parties agree on which key to use and then use it. When it is time to distribute session keys there are already keys established for sending messages.

Security Levels

The reasons for providing two different security levels in ARTSY are summarized in the tables 5.2.2 and 5.2.2. The tables show the pros and cons for a client that chooses one of the levels.
<table>
<thead>
<tr>
<th>Level</th>
<th>CP consequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insecure</td>
<td>+ Installation does not require physical meeting with personnel from the Trusted Party. + Requires less computer power than the secure level since no messages are encrypted or authenticated. - No security at all. - It is not possible to charge money from an RC that orders a Content Representation.</td>
</tr>
<tr>
<td>Secure</td>
<td>+ All Content Representations are well protected since all data that is transmitted is encrypted and all transmissions are authenticated. + It is possible to charge money from an RC that orders a Content Representation if that RC supports the secure security level. - The installation requires a physical meeting with personnel from the Trusted Party. - RCs that only supports the insecure security level cannot order Content Representations unless the CP explicitly accepts this. - Requires more computer power than the insecure security level.</td>
</tr>
</tbody>
</table>

**Table 5.1. Consequences for a CP when choosing a security level**

<table>
<thead>
<tr>
<th>Level</th>
<th>RC consequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insecure</td>
<td>+ Installation does not require physical meeting with personnel from the Trusted Party. + Requires less computer power than the secure level since no messages are encrypted or authenticated. - It is not possible to order a Content Representation from a CP that charges money. - Somebody could make orders using the name of the RC (but no orders that involve money). - A CP that uses the secure security level may refuse a transaction that does not involve money</td>
</tr>
<tr>
<td>Secure</td>
<td>+ It is possible to order Content Representations from a CP that charges money. - The installation requires a physical meeting with personnel from the Trusted Party. - Requires more computer power than the insecure security level.</td>
</tr>
</tbody>
</table>

**Table 5.2. Consequences for an RC when choosing a security level**
5.3 Software Requirements Specification

5.3.1 Method

The IEEE830-1998 was chosen as the template for the Software Requirements Specification. The reason for choosing IEEE830-1998 was that it is a well-known and often used standard. If the structure of a document is recognized it is reasonably easier to understand and use the document.

5.3.2 Result

The SRS was based on the Main Feature Model and the results from the Security Analysis. More than 200 separate requirements are specified in SRS. It may seem like a lot of requirements, almost half of them are however non-essential requirements. Requirements that are non-essential are specified with the intention to serve as design guidelines, not as strict requirements. The ambition has been to create a detailed SRS and to provide as much help as possible to future implementors. If an implementor comes up with a solution to a problem that seems better than the solution suggested by some non-essential requirements, then the implementors should choose the new solution (and thus breaking the non-essential requirements).

Some parts of ARTSY are better described than other. Among the things that have not been prioritized are the Browser Clients and the Value Adding Agents. Both of these components enhances ARTSY but are not needed to use the system. A BC would first and foremost enhance the user experience while a VAA would add some new functionality. In the SRS the critical features are prioritized. The procedure where a Content Representations is transmitted from a CP to a RC is rather well described. More than 20 requirements are specified in the transaction section. Three requirements that concern BCs are specified and none of those are essential. It would have been better if all parts would have been investigated and described completely, but when working according to a time schedule with limited time, this is not always an option.

The SRS requires that ARTSY uses one Trusted Party. An alternative would be to suggest some sort of Chain of Trust (see 2.1.2), that would allow for better scalability. Using only one Trusted Party may become insufficient if the system grows to gigantic proportions. The system is also more vulnerable if it depends on one Trusted Party only. Clearly, a Chain of Trust would be more flexible and more reliable. Using only one Trusted Party is still the solution presented in the SRS. The reason for this is that this is the least complicated solution and it is likely that it will be sufficient, even if the system becomes quite large.

The SRS requires that at least two levels of security are supported. It would perhaps seem better to simply require all involved parties to employ maximum security at all times. That would make the system less complex but still secure. The reason for requiring at least one insecure level, is that it is probable that a substantial number of potential ARTSY customer would like to distribute digital content in their holding, without charging for it. Requiring maximum security
on all file transfers would make the system more complicated than necessary, for such customers. The extra effort that maximum security requires may make those customers choose another way of distributing files.

In a Tender, that is sent from a CP via a TS to an RC, an approximate price is specified. It is called an approximate price since it is not guaranteed that this price exactly is the same price that will be charged for the transaction. The reason for not being able to specify the exact price is that exchange rates fluctuates over time. A price that is valid day one may be invalid day five. In most cases the approximate price can be considered an exact price. Exchange rate fluctuations are often not noticeably over a few days. Fluctuations can however be noticeable, even over a few days, when trading with less-stable currencies. Just providing an approximate price in the Tender might seem like an unsatisfactory solution. The exact price can however always be calculated, since the price that the CP charges and the price that TS charges also are included in the Tender (as well as the currencies that they use).

**Prototype Requirements**

A reduced set of requirements are specified for a prototype implementation, in the SRS. The purpose of removing some requirements was to make the implementation task practicable in the time available. The SRS requires that the most critical parts of ARTSY are implemented in a prototype. This includes how different messages are encrypted and how different component communicates with each other. The prototype must encrypt and decrypt data using real cryptographic algorithms. New session keys must be generated before a transaction can take place. It is required that the prototype utilizes a DBMS and that important documents are stored.

The prototype does however not need to support different security levels. It is sufficient to support the full security level. It is not technically interesting to show that it is possible to transfer messages without protection if it can be showed that it is possible to transfer messages with protection. The prototype does not have to support more than one TS and it does not have to use certificates nor a Public Key Registry. In a full scale implementation this would not be acceptable, but for demonstration purposes such simplifications can be accepted.

At the beginning of the master thesis project, the ambition was to insert the RCi module into a printer. This might have been possible with support from Hewlett-Packard, the manufacturer of the printer. Hewlett-Packard was contacted, but they were unfortunately not able to give the support needed. When this became clear the requirement on inserting the RCi into a printer was skipped.

### 5.4 Prototype

#### 5.4.1 Method

The JOOM model was used for designing the prototype. JOOM is a structured, object-oriented approach that focuses on facilitating error detection and on pro-
producing re-usable designs. Relations between objects are emphasized as well as the modularity of the design. A design that is created using JOOM is easy to translate to object oriented code, such as Java code.

The method for implementing the prototype was inspired by the eXtreme Pro-
gramming paradigm (XP). XP is a fairly new paradigm that has become popular lately. It is designed to work with projects comprising of small teams (2 – 10 programmers) that are not sharply constrained by the existing computing environment, and where a reasonable job of executing tests can be done in a fraction of a day. Considering this, XP must be regarded a suitable choice for this project.

XP was based on observations of what made computer programming faster and what made it slower. One of the promises of XP is that is promotes productivity.

XP projects unanimously report greater programmer productivity when compared to other projects. XP was not adopted to its full extent in the project. In order to do so a considerably large amount of time would have to be invested to fully understand all of the concepts. Other things were simply prioritized.

5.4.2 Result

XML

XML was chosen to represent information. A conceivable alternative would have been to implement an own way of representing information. If information would have been represented directly by Java objects, the code would have been less complex and thus easier to understand: Fewer conversions would have been needed and messages could have been used more directly. The advantage of using XML is that the data objects themselves are easier to understand. If the whole system goes down it is at least possible to understand data from stored objects since the data consists of plain text. Moreover, XML is a standardized format. If a future implementation is to interact with other software the adaption will be considerably easier (than if an own, special format would have been used). Using XML has other advantages as well, it is for example easy to extend XML documents. If it turns out that additional information must be transmitted, in addition to what was initially planned, a minimal adaption of the system is required. The XML structure is extensible from the start. An own plain text based format could be implemented, a format that was designed to be extensible. The reason for not doing so is that such a special format would have so much in common with XML that it simply is better to go for XML instead, which is standardized. Going for XML also has another advantage: Numerous Java classes for managing XML documents have been implemented already. Instead of writing own code that manages an own format existing code can be used.

Java

Java was chosen as the programming language for the implementation of the proto-
type. As can be concluded from section 2.2.2, Java has a handful of network related
characteristics that renders it a suitable language for the distributed ARTSY environment. One of the initial requirements on the thesis project was to produce a prototype implementation that runs on multiple platforms. Java is by nature platform independent. A program implemented in Java can be executed on every platform to which the Java interpreter has been ported. Multiplatform systems can, however, be implemented without using Java. It would for example be possible to use C++ (which is far more efficient when it comes execution speed) and compile the source code on several different platforms. If platform independence had been the only consideration this might have been an acceptable solution.

Another advantage with Java is that it comes with a wide range of class libraries. A Java programmer is free to use any of the classes provided in the SDK. The included classes provides methods for producing random numbers, for encrypting and decrypting objects, for sharing objects over a network and many other things. The methods mentioned are all needed in an ARTSY prototype implementation. If such methods have not been directly available it would have been necessary to either implement them from scratch or to try to locate implementations that already have been made. In either case productivity would have been affected in a negative way.

Security is a critical aspect of a distributed system. In Java, there are no ways for programs to take harmful actions towards the system, since all Java programs are executed within the Java sandbox. If all programs were free from bugs, this would not be so important. It is however hard, if not impossible, to guarantee that a program is entirely free from bugs. With Java it can be guarantee that a program does not harm the system, no matter what the program actually does and how well it is implemented. This guarantee, is an important advantage of Java. Many other interpreted languages share this advantage with Java.

Java is a simple language. It is relatively easy to learn, just as C/C++. But unlike the C/C++ programmer, a Java programmer does not have to bother about memory management but can focus more on the actual implementation tasks.

A useful tool included in the Java SDK is the JavaDoc tool. JavaDoc automatically generates standardized documentation of Java code. Third-party software may provide similar functionality for other programming languages but JavaDoc has an advantage over third-party software, simply by the fact that it is delivered along with the Java SDK. JavaDoc has become established as the standard Java documentation tool and JavaDoc-generated documentation is be recognized by almost every Java programmer, since the Java API itself is documented on the Internet using JavaDoc.

The biggest disadvantage of Java is its execution speed. Java may be a good performer compared to other interpreted languages but it can never compete with compiled languages such as C++. In ARTSY execution speed is not critical. ARTSY is not a real-time system that require real-time responses to events. When the prototype implementation was tested network transportation times was always longer than the corresponding encryption/decryption times. As long as the network transportation represents the greatest delay in ARTSY, execution time is not really an issue.
5.4 Prototype

Design Choices

There are two reasons why a Content Representation is not transmitted as a standard ARTSY Message. All Messages are always encrypted in the same way; using asymmetric encryption. First, asymmetric encryption is comparatively slow when encrypting substantial amounts of data (see section 2.1.2). Second, if an entire Content Representation is encapsulated in one single Message, the entire Content Representation must be read before it can be sent by the CP. The RCo then has to receive the entire Content Representation before it can forward it to the RCi. The RCi must correspondingly wait until it has received the entire Content Representation before it can start reproducing it. The entire file must be received twice before the reproduction process can start and this will delay the process. An example: A Content Representation of 20 MB is to be transmitted in a Message. The transmission time is 5 minutes. Since the entire Content Representation has to be received first by the RCo and then by the RCi before it can be reproduced it will take about 10 minutes before the reproduction process can start. In the prototype Content Representations are streamed, the reproduction process starts as soon as the first (small) part of the Content Representation has been received. After 5 minutes the entire Content Representation has been received at the RCi and the reproduction process is likely to be almost completed.

Security

The implementation of the result from the security analysis is discussed below:

Security Protocol

The key length of the asymmetric algorithm is 1024 bits while the key length of the symmetric algorithm is 56 bits. An asymmetric key of 256 bits is comparably strong to a symmetric key of 56 bits. Since the asymmetric key is both easier to attack and more valuable to protect, a much stronger asymmetric key was chosen (see section 2.1.2 page 13). If an adversary breaks a symmetric key he can only get access to one Content Representation in one transmission. If an adversary breaks an asymmetric key he can cause severe problems. If the private key of the Trusted Party is compromised all other components must be reinstalled with a new public key of the Trusted Party and the Public Key Registry must be cleared and refilled. It is therefore more important to have a strong asymmetric key than a strong symmetric key.

A key length of 56 bits was chosen for the symmetric keys. Each symmetric key is used only once and does not need the same level of protection as the asymmetric keys. Using an average PC it takes 208 days to break a 56 bits symmetric key (see table 2.1). If a stronger encryption would have been chosen, the performance of the system would have been affected in a negative way. 56 bits was considered a reasonable trade-off between security and performance.
Secure Messages
The RSA algorithm was chosen to provide confidentiality of messages. RSA is the most popular asymmetric algorithm. It has been used in many applications and it has withstood the test of many adversaries (see section 2.1.2). It is also a reversible algorithm and can thus be used both for encryption and signing. Both encryption and signing are needed in ARTSY.

RSA is used with CBC mode when the Information part of a message is encrypted. ECB mode was not chosen since patterns in plaintext can yield patterns in the ciphertext when using ECB (see section 2.1.2). If the same message is encrypted twice using the CBC mode it does not yield the same ciphertext. The error recovery is worse when using the CBC mode than when using the ECB mode. If it is a small error or a big error, it does not matter when the message is signed. If a message is received with errors, the verification will fail. The TCP protocol does however guarantee that erroneous messages are retransmitted.

Neither the CFB mode nor the OFB mode can be used with an asymmetric algorithm since the cipher needs to be initialized with the same key irrespective of if it will be used for encryption or decryption (see section 2.1.2).

The initialization vector (IV) is filled with random numbers each time an encryption of a message is about to start. This is done to prevent patterns in the plaintext from yielding patterns in the ciphertext. The SecureRandom class is used but it is not fed with a truly random seed. It is not necessary to do this since the IV does not have to be random, only different from time to time, to prevent plaintext attacks. The IV is not encrypted, it is not needed (see section 2.1.2). The integrity of the IV is protected which prevents a faulty decryption, in case the IV would be changed during the transmission.

When making signatures the hash value of SHA-1 is computed for the message. The hash value is then signed using the RSA algorithm together with the private key of the sender. The SHA-1 algorithm was chosen rather than the MD5 because of it is more popular and because it is more resistant to brute-force attacks (see section 2.1.2).

In section 4.2.2 the Information is signed before it is encrypted but in the prototype the Information is signed after it is encrypted. This forces the id of the sender to be put in the Extra part of the Message since the id of the sender is needed to verify a Message. The id of the sender is thus sent unencrypted but this does not compromise the security of ARTSY, but enables the use of the SignedObject class.

Secure Content Representations
The DES algorithm was chosen because it has been used for more than 20 years and no easy way to attack it has been found yet. The greatest weakness of DES is that it only can be used with 56 bits keys, in five years a $1000 computer could find the secret key of a ciphertext encrypted with DES in only two days (see section 2.1.2, page 13). Triple DES is a better choice,
but it would slow down the encryption. Triple DES would probably be a better choice in a commercial version of ARTSY — but again performance must be weighed against security before making the choice. It would be very simple to change from DES to Triple DES in the current implementation, only two lines of code would have to be replaced. AES is a new standard that replaces DES and it is said to be more secure than DES. But since it has been used for only a short period of time, it is not recommended that it is used in ARTSY for encrypting Content Representations yet. The one-time pad (see section 2.1.2) is the perfect encryption scheme but was not used for session keys due to some limitations. The pad can only be used once and must have the same size as the plaintext that shall be encrypted. Since the session keys only are used once the first limitation is not a problem. But to transfer a one-time pad the same size as the Content Representation would require a lot of extra bandwidth.

If an adversary has the plaintext and ciphertext for several messages, he can start to compile a code book without knowing the key. Since the key is only used once the adversary can only get information about a key from one single transmission. If an image is transmitted the data is not predictable (only meta-information in the beginning of the file, that describes the encoding format of the picture has, can be predicted).

It is of great importance that there is no way for an adversary to be able to generate the same session keys as a TS does, even if the adversary has access to the source code that is used to generate the keys. By feeding the SecureRandom class with a truly random seed from the /dev/random system file this is made very hard (see section 2.1.2). Different values are returned each time this file is read.

The Key Establishment Protocols

Only the symmetric key establishment protocol was implemented. The asymmetric key establishment protocol was not implemented due to lack of time. The asymmetric key establishment protocol is not difficult to implement, all techniques needed already exist in the prototype. The asymmetric key establishment protocol would be put in to practice by a Public Key Registry and methods in each component for automatically fetching public keys from the registry. Code needed for signing and verifying the public keys already exist, a database manager for storage of the keys already exist, and the Java RMI framework facilitates an implementation of code for making calls from the components to the database.

Java RMI

By choosing a middleware solution for the communication between objects in ARTSY, the implementation was made easier. The two main candidates when using Java are Java RMI and CORBA. The choice fell on Java RMI. Java RMI provides distributed garbage collection on remote server objects. This makes the
programming simpler since the programmer does not have to dispose of obsolete objects. This is done by Java RMI. Garbage collection does not exist in CORBA.

Any object that is sent over the network must be declared using IDL when using CORBA. When using Java RMI any Java object that implements the Serializable interface can be passed as a parameter between processes.

The greatest disadvantage of Java RMI is that it only can be used for connecting applications written in Java. The ARTSY prototype is written entirely in Java, so this is not a problem. If the prototype is extended in the future and modules written in other programming languages will be used, it is still possible to use Java RMI. RMI over IIOP can be used to make Java applications communicate with applications written in other programming languages.
Chapter 6

Conclusion

An ARTSY Main Features Model was developed. Three main features are described in the model: The Information Request, the Tender Submission and the Transaction. The purpose of the Information Request is to provide a Reproduction Client with information about what a Content Provider offers. The purpose of a Tender Submission is to inform a Reproduction Client of the cost of a particular reproduction. The purpose of the Transaction is to perform a transmission and a reproduction.

An analysis of the security threats in the ARTSY context was performed and a security model was developed. This model was approved by a computer security expert; assistant professor Viiveke Fäk at Linköping university. The security analysis shows that ARTSY is subject to the following threats: Interception, modification, interruption and fabrication attacks. The services that are required are: Confidentiality, integrity and authenticity. The mechanisms that were chosen for the security model to provide these services are: Asymmetric cryptology, digital signatures, symmetric cryptology and a public key registry.

A Software Requirements Specification was developed. It was based on the Main Features Model and the security model. The SRS is not targeted at reproduction systems of images only. It contains extra directives for image reproduction systems but it should be possible to use the SRS for an arbitrary type of reproduction system. The SRS contains an alternative reduced set of requirements that can be used for an ARTSY prototype implementation. The network security of the system specified in the SRS is sufficient. Local security, such as access control and security of the operating system, needs to be investigated further. It is not possible to protect ARTSY against all security threats using software only. The hosts on which the system resides must be protected also. Contracts should be used to regulated some security issues. This is not described in the SRS. Another thing that is not described in the SRS is how to design the graphical user interfaces.

A prototype of ARTSY was implemented using the Java programming language. The prototype uses XML to manage information and Java RMI to enable remote communication between its components. Using XML is better than using an own
standard. 1024 bits RSA is used for asymmetric encryption and 56 bits DES is used for symmetric encryption. In a secure system where cryptographic algorithms are used it is important that they easily can be replaced if needed, in the prototype this can be done without great effort. The database manager system miniSQL is used for database accesses. The prototype does not use a public key registry or certification authority, but it does meet the prototype requirements specified in the SRS. It was built as a platform independent system and it has been tested and proven to be operational on the Sun Solaris platform as well as the Win32 platform.

6.1 Recommendations

Before ARTSY can be used as a commercial system it is recommended that the following tasks are accomplished:

- Carry out a market survey
- Develop a full-scale implementation
- Establish security policies for ARTSY

Market Survey

Contact with potential users of ARTSY has unfortunately been very limited so far in this project. Before the project is continued the needs of potential users must be investigated. Museums, libraries, archives and/or print workshops should be contacted and interviewed. An ARTSY prototype (possibly the current implementation) should be installed and tested at potential users that act as both RCs or CPs. A wish/must list of desired features should be established after the testing period is over. It must also be investigated if it is relevant to proceed with the project from a financial point of view. A cost estimate should be established.

Develop a full-scale implementation

Whether the current prototype implementation should serve as the starting point for a full-scale implementation or not, should to be investigated. If the current prototype is further developed, it has to be extended in (at least) the following ways:

- The graphical user interface must be enhanced
- A Public Key Registry must be implemented (certificates could alternatively be used)
- Access control for the DBMS must be implemented
- The private encryption keys must be stored in a safe way so that they cannot easily be compromised
6.1 Recommendations

- The RCi module should be implemented into the Reproduction Unit if this is feasible
- A Browser Client interface should be implemented
- Additional features according to needs of potential users probably has to be implemented

Irrespective of if the current implementation is used when developing the full-scale implementation or not, the full-scale implementation should comply with the full set of requirements formulated in the ARTSY Software Requirements Specification.

Establish Polices

Some of the requirements of ARTSY cannot be implemented in software, these requirements has to be realized in other ways. The hardware that is used for ARTSY must be protected from adversaries. How the hardware shall be protected by each party in ARTSY must be regulated. How secure transaction of money can be performed should be investigated. Each client that connects to ARTSY should sign a contract that forbids the Client to alter the software in any way, this contract must be formulated.
Bibliography


APPENDIX A
ARTSY
Software Requirement Specification
Contents

1 Introduction .............................................. 5
1.1 Purpose .............................................. 5
1.2 Scope .................................................. 5
1.3 Vocabulary ............................................. 5
1.3.1 Abbreviations ....................................... 8
1.4 Overview ............................................... 9

2 Overall description ....................................... 9
2.1 Product Perspective .................................... 10
2.2 Product Functions ....................................... 11
2.2.1 Information Request ................................. 12
2.2.2 Tender Submission ................................. 12
2.2.3 Transaction ......................................... 14
2.3 Assumptions and Dependencies ......................... 14
2.4 Apportioning of Requirements ......................... 16

3 Specific Requirements .................................. 16
3.1 External Interfaces ..................................... 17
3.1.1 User Interfaces ..................................... 17
3.1.2 Software Interfaces ................................ 18
3.1.3 Communications Interfaces ......................... 19
3.2 System Architecture ................................... 20
3.3 System Features ......................................... 20
3.3.1 Information Request ................................. 20
3.3.2 Tender Submission .................................. 22
3.3.3 Transaction ......................................... 24
3.4 System Security ......................................... 29
3.4.1 Security Levels ..................................... 29
3.4.2 Security Mechanisms ............................... 32
3.4.3 Public Key Registry ............................... 33
3.5 System Administration ................................ 35
3.5.1 Reproduction Client ............................... 35
3.5.2 Content Provider ................................... 36
3.5.3 Transaction Server .................................. 36
3.6 Required Data Files ................................... 37
3.6.1 Content Meta Data ................................. 37
3.6.2 Tender Invitation ................................... 38
3.6.3 pre-Tender ......................................... 39
3.6.4 Tender ............................................... 40
3.6.5 Price List .......................................... 41
3.6.6 Reproduction Capabilities ......................... 41
3.6.7 RCi-log .............................................. 41
3.7 Logical Database Requirements ....................... 42
3.7.1 Reproduction Client ............................... 42

3
3.7.2 Content Provider .............................................. 43
3.7.3 Transaction Server ............................................ 45
3.8 Software System Attributes ................................... 46
  3.8.1 Security ...................................................... 46
  3.8.2 Portability .................................................... 47

A Use-cases ............................................................. 47
  A.1 Definition ....................................................... 47
  A.2 Actors ............................................................ 47
  A.3 Use-cases ........................................................ 48
    A.3.1 Request Information about Content .................... 48
    A.3.2 Invite Tender .............................................. 48
    A.3.3 Order Content ............................................. 49
    A.3.4 Investigate Corrupt Transaction ....................... 49
    A.3.5 Add Content .............................................. 50
    A.3.6 Add CP ..................................................... 50
    A.3.7 Revoke Public Key ......................................... 50

B Messages sent in ARTSY ........................................... 52
1 Introduction

1.1 Purpose

The purpose of this document is to define requirements for *A Reproduction Transaction System* (ARTSY). The intention is to facilitate the development of such a system. The intended audience is potential developers of ARTSY and potential users of ARTSY.

1.2 Scope

The product to be produced is called ARTSY (A Reproduction Transaction System). Its main task is to permit secure transactions and reproductions of digital content over an insecure media such as the Internet. This involves ensuring that if x copies of some file are sent, only x copies can be reproduced. Files that have been transmitted to an Reproduction Client (RC) cannot be stored locally and reproduced later. Transmission and reproduction of Content Representation files are handled by ARTSY, as well as fees associated with the transactions. ARTSY does however not, handle transaction of money and it does not guarantee that Content Providers (CPs) get paid for the Content they offer.

Virtually any digital computer file can be considered as Content. The ARTSY project however, focuses primarily on transactions and reproductions of digital images, for example digital representations of art. The requirements specified within this document are nevertheless formulated in a general way, so that a system built on the basis of this specification is not bound to support digital images only.

ARTSY can be used to connect museums, libraries and archives in possession of digital content with print workshops, poster shops and museum shops. Another application area is record companies in possession of music and record shops able to make records.

The goals of ARTSY are to decentralize the distribution of digital content via the Internet and to make more reproductions available locally at more sites. ARTSY also aims at opening a new distribution channel for Content Providers, and thus generating more money for both Content Providers and Reproduction Clients. An important detail is that content is delivered directly from a Content Provider to a Reproduction Client. ARTSY offers a cheap way of distributing reproductions since the number of links between the provider and the end customer are reduced to a minimum.

1.3 Vocabulary

This section describes the meaning of some important and frequently used terms. It should be noted that the terms explained have a special meaning within this document and that this meaning in some cases differs from the commonly used meaning of the same terms.
1 INTRODUCTION

Original
An Original is a physical artifact. A painting can be referred to as an original. A Reproduction of a painting can never be called an original.

Content
Content is an abstract representation of a physical object (Original), it is represented by a set of one or more Content Representations.

Content Representation
A Content Representation is a digital representation of some Content. A picture could for example be represented by a jpeg file and a bitmap file. Content Representations are either permanently stored at a Content Provider or dynamically generated when needed.

Content Provider
A Content Provider is a set of devices capable of storing and transmitting Content Representations. The term includes both the hardware and software needed to accomplish this and the personnel needed to administrate the system. A Content Provider has the legal rights to offer content to a buyer. A museum that has produced digital versions of some paintings could for example act as a Content Provider.

Reproduction
A reproduction is a physical manifestation of a Content Representation. An example: A museum produces an Content Representation of a painting using a digital camera or a scanner. The Content Representation is then used to produce a printed poster. The printed poster is a Reproduction.

Reproduction Client
A Reproduction Client is a set of devices capable of receiving Content Representations and making Reproductions. The Reproduction Client uses a Reproduction Unit to do make Reproductions. The term Reproduction Client includes the Reproduction Unit, the components needed to receive a Transaction and the personnel needed to administrate this system. A print workshop could for example act as a Reproduction Client.

Reproduction Unit
A reproduction unit is the hardware and software needed to reproduce Content Representations. Every Reproduction Unit is a part of a Reproduction Client. A printer can for example be a Reproduction Unit.

Transaction
A transaction is a procedure in which a Content Representation is transmitted from a Content Provider to a Reproduction Client. A Transaction Server is involved in the procedure to ensure that the Content Representation is protected if that is needed.

Transaction Server
A Transaction Server mediates Transactions. Its most important task is
to ensure that a Content Representation that is transmitted reaches its right destination and that it is impossible for an outsider to record the transmission and make illicit copies of the Content Representation.

Value Adding Agent
A Value Adding Agent is a piece of software that can be used to enhance Content Representations. The Reproduction Client pays for the enhancement. A Value Adding Agent can also act as a sponsor by adding advertisement to the Content Representation. In the latter case the Value Adding reduces the cost for the Reproduction Client instead of increasing it.

Value Adding Agent Provider
Each Value Adding Agent is accessible through a Value Adding Agent Provider only. A Value Adding Agent Provider is a server that provides one or more Value Adding Agents.

Component
An ARTSY Component is the hard- and software of a Reproduction Client, a Content Provider, a Transaction Server or a Value Adding Agent Provider.

Party
An ARTSY Party includes an ARTSY component and the personnel and users that interact with and administrates the component.

Client
An ARTSY Client is either a Reproduction Client or a Content Provider. The difference between Client and Component is that the Transaction Server is a Component but not a Client.

Trusted Party
A Trusted Party provides ARTSY Components with public encryption keys by maintaining a public database.

Image
An ARTSY Image is an abstract representation of a physical image. In an ARTSY implementation intended for images only (ARTSYi), the term Image is synonymous to Content.

Image Representation
An ARTSY Image Representation is a digital representation of an Image: For example a jpeg file or a bitmap file (see figure 1). A black and white jpeg image and a color image could be two different representations of the same Image. An Image Representation can either be stored permanently at a Content Provider or it can be generated dynamically when needed. For example, a customer could request a certain part of an Image Representation. This part does not exist as a separate file at the Content Provider but can be generated from another Image Representation. The
requested part becomes a new Image Representation when it has been generated.

![Diagram of original image and its representations](image.png)

Figure 1: Original, Image and Image Representation

### 1.3.1 Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARTSY</td>
<td>A Reproduction Transaction SYstem</td>
</tr>
<tr>
<td>ARTSYi</td>
<td>A Reproduction Transaction SYstem for images</td>
</tr>
<tr>
<td>BC</td>
<td>Browser Client</td>
</tr>
<tr>
<td>CA</td>
<td>Certificate Authority</td>
</tr>
<tr>
<td>CP</td>
<td>Content Provider</td>
</tr>
<tr>
<td>RC</td>
<td>Reproduction Client</td>
</tr>
<tr>
<td>RCi</td>
<td>Reproduction Client inner module</td>
</tr>
<tr>
<td>RCo</td>
<td>Reproduction Client outer module</td>
</tr>
<tr>
<td>TS</td>
<td>Transaction Server</td>
</tr>
<tr>
<td>VAA</td>
<td>Value Adding Agent</td>
</tr>
</tbody>
</table>
1.4 Overview

This document is structured according to the IEEE830-1998, which is a recommended practice for software requirements specifications\(^1\). The reader is recommended to read section 2 first, to get an overview of ARTSY. Section 3 contains detailed requirements on ARTSY and should be read by developers and by readers that take a special interest in the system. Appendix A contains use-cases that exemplifies the main features of ARTSY.

2 Overall description

ARTSY is based on three types of components: Content Providers (CPs), Transaction Servers (TSs) and Reproduction Clients (RCs). A CP offers Content Representations and an RC reproduces them. A TS mediates transactions of Content Representations. In figure 2 one RC starts a transaction by ordering a Content Representation. The TS registers the transaction and informs the CP which then transmits the Content Representation to the RC.

\[\text{Figure 2: An RC orders an Image Representation}\]

ARTSY permits secure transactions over an insecure media, such as the Internet. This involves ensuring that if x copies are sent only x copies can be reproduced. There can be many CPs, TSs and RCs. A CP is bound to one TS in the sense that it only communicates with one TS. An RC is independent and is not bound to any TS, it communicates with different TSs at different times.

The RC component is divided into two modules\(^2\): The RC outer module (RCo) and the RC inner module (RCi). The RCo handles the network traffic and user interaction while the RCi manages the production of reproductions. The reason for dividing the RC into two separate modules is to enhance security. The RCi is located inside the reproduction unit (for example a printer), while

\[\text{\(^1\)http://www.ieee.org/portal/index.jsp}\]
\[\text{\(^2\)This is not a strict requirement, but it is recommended}\]
the RCo is located on a host computer just as the CP and TS components are. Locating the RCI inside of the reproduction unit makes it possible to transmit encrypted data all the way from the CP to the reproduction unit. Sensitive data will thus stay encrypted as long as it is possible. Decryption will take place inside of the reproduction unit and not on the host computer where the RCo is located. If the RC would not be divided into two modules, the decryption would have to be managed by the host computer. If data would be decrypted on the computer rather than the reproduction unit, it would have to be transmitted from the computer to the reproduction unit unencrypted, without protection. Such a transmission could be recorded, manipulated and re-used in order to produce illicit reproductions.

ARTSY optionally supports interactions with Value Adding Agents (VAA). A VAA is a separate component that works similar to a filter for Content Representations. A Content Representation can be sent from a CP to an RC via a VAA that enhances it, if the RC wishes so. A VAA could for example convert a color image into a gray scale image, or adjust the contrast in a desired way. A VAA can charge a fee for enhancing Content Representations. A sponsor can also act as a VAA, in this case the VAA pays part of the cost for transmitting the Content Representation and in exchange gets to change the Content Representation in some way, for example by putting a logo type on it. Several VAA's can be used in sequence during one transaction (see figure 3).

![Figure 3: Two VAAs enhance a Content Representation.](image)

The VAA itself is a small program that is able of refining Content Representations in some way. Several VAA programs may be located together, at a VAA-provider. Every VAA-provider is connected to one TS. Putting many VAA's together under the control of one TS, makes charging for VAA services easier. If for example four VAA's are used in one transaction and every VAA takes a fee for its services, the money can be payed to the TS, which then can distribute the money to the four VAA's. The RC does not have to pay four small amounts to the four different VAA's. Figure 4 gives an overview of a small ARTSY.

### 2.1 Product Perspective

ARTSY is a self-contained system, apart from a database management system (DBMS) ARTSY does not require any third party software in order to work. ARTSY itself does not include a Browser Client (BC). A BC is a browser program that can be used to browse Content at various sites before ordering it.
ARTSY does however, allow interactions with separate third party Browser Clients via a special external interface, see section 3.1.2.

2.2 Product Functions

ARTSY is intended for transmitting Content securely from a CP to an RC. The transaction procedure normally includes three separate phases where the actual transaction of Content constitutes the last one:

1. Information Request
2. Tender Submission
3. Transaction

A list with all messages that are sent during all three phases are presented in Appendix B. Appendix A contains use-cases that exemplifies all important functions in ARTSY.
2.2.1 Information Request

An Information Request is sent from an RC to a CP (see figure 5). Its purpose is to provide the RC with relevant Content information, making it possible for the RC to decide whether to continue or not with the transaction procedure. The Information Response provides the RC with a list of available encoding formats and other Content information. If the Content is an image this information can for example include the resolution and the color depth of the image. Figure 6 shows an Information Response that describes an image: Mona Lisa by Leonardo Da Vinci. This Image has two Image Representations, one in jpeg format and one in tiff format. These Image Representations also have different resolutions.

2.2.2 Tender Submission

When the RC has received an Information Response it can invite a Tender for a Content Representation encoded in a format that the RC can reproduce (see figure 7). The requested Content Representation is specified in the Tender Invitation and Content specific constraints can also be specified. If an image is to be transmitted such constraints can for example define a sub area of the image. Other constraints such as reduction of color depth or resolution might also be defined. In figure 8 a Tender Invitation is showed that invites a Tender on Mona Lisa, Image Representation number 1. This Tender Invitation does not define any constraint on the Image, it invites a Tender on an Image Representation that already exists.

A Tender specifies a Content Representation and a total transaction price. Figure 9 shows the Tender received after the Tender Invitation was transmitted in figure 8.

The Tender Invitation is first sent to the CP which on reception submits a pre-Tender to the TS. The pre-Tender will be a part of the Tender and includes for example the fee that the CP will charge for transmitting the Content Representation. The TS makes a Tender based on the pre-Tender and adds the fee
CMD:
CP-ID: cp1

<table>
<thead>
<tr>
<th>IMAGE:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID: 8</td>
</tr>
<tr>
<td>TITLE: Portrait of Mona Lisa (1479 - 1550)</td>
</tr>
<tr>
<td>ARTIST: Leonardo Da Vinci</td>
</tr>
<tr>
<td>DESCRIPTION:</td>
</tr>
<tr>
<td>YEAR:</td>
</tr>
<tr>
<td>START: 1503</td>
</tr>
<tr>
<td>END: 1506</td>
</tr>
<tr>
<td>TECHNIQUE: Painted in oil</td>
</tr>
<tr>
<td>SIZE:</td>
</tr>
<tr>
<td>WIDTH: 530</td>
</tr>
<tr>
<td>HEIGHT: 770</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IMAGE-REPRESENTATION:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID: 1</td>
</tr>
<tr>
<td>DESCRIPTION:</td>
</tr>
<tr>
<td>TECHNIQUE: Photograph - direct reflecting light</td>
</tr>
<tr>
<td>FORMAT: jpeg</td>
</tr>
<tr>
<td>RESOLUTION:</td>
</tr>
<tr>
<td>X: 512</td>
</tr>
<tr>
<td>Y: 384</td>
</tr>
<tr>
<td>PRICE-CLASS: 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IMAGE-REPRESENTATION:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID: 2</td>
</tr>
<tr>
<td>DESCRIPTION:</td>
</tr>
<tr>
<td>TECHNIQUE: Photograph - direct reflecting light</td>
</tr>
<tr>
<td>FORMAT: tiff</td>
</tr>
<tr>
<td>RESOLUTION:</td>
</tr>
<tr>
<td>X: 6400</td>
</tr>
<tr>
<td>Y: 4800</td>
</tr>
<tr>
<td>PRICE-CLASS: 1</td>
</tr>
</tbody>
</table>

Figure 6: An Information Response describing an Image
TENDER-INVITATION:
TENDER-INVITATION-ID: 239
IMAGE-REPRESENTATION-ID: 1
COPIES: 1
RC-CURRENCY: sek
RC-ID: rcl
RC-HOST: bullock.isy.liu.se
CP-ID: cp1
CP-HOST: bancroft.isy.liu.se

Figure 8: A Tender Invitation on Image Representation number 1

that the TS will charge for managing the transaction.

2.2.3 Transaction

When a Tender has been received the RC can initiate a transaction. An order message including the id of the Tender is sent to the TS (see figure 10). The TS starts the transaction by informing the RC and the CP participating in the transaction. During the transaction the Content Representation is encrypted and transmitted from a CP to an RC. At the RC the Content Representation is decrypted and reproduced. The TS has the overall responsibility for the transaction and is expected to solve possible controversies between the participating parties.

2.3 Assumptions and Dependencies

In order for ARTSY to work, every computer acting as a host for an ARTSY component (such as a TS, RC or a CP) must have a reliable Internet connection. Any two ARTSY components must be able to communicate with each other
2.3 Assumptions and Dependencies

TENDER:
TENDER-ID: 225
PRE-TENDER-ID: 213
TENDER-INVITATION-ID: 240
VALID-UNTIL: Monday, February 3, 2003 10:52:03 AM MET
IMAGE:
ID: 0
TITLE: Portrait of Mona Lisa (1479 - 1550)
ARTIST: Leonardo Da Vinci
IMAGE-REPRESENTATION:
ID: 1
DESCRIPTION:
TECHNIQUE: Photograph - direct reflecting light
FORMAT: jpeg
RESOLUTION:
X: 512
Y: 384
PRICE-CLASS: 2
COPIES: 1
RC-ID: rc1
RC-HOST: bullock.isy.liu.se
RC-CURRENCY: sek
CP-ID: cp1
CP-HOST: bullock.isy.liu.se
CP-FEE: 3
CP-CURRENCY: eur
TS-ID: ts
TS-HOST: bullock.isy.liu.se
TS-CURRENCY: sek
TS-FEE: 10
TOTAL-FEE: 37.06
TOTAL-CURRENCY: sek

Figure 9: A Tender
using the Internet. ARTSY also depends on the function of the DBMSs and the reproduction units that it uses.

2.4 Apportioning of Requirements

Some system design issues are not discussed in this document. The focus is on the functional requirements on the system. Human-Computer Interaction (HCI) aspects are not treated at all. It is recommended that this SRS is complemented with additional requirements that cover HCI aspects.

The BC component is not considered as a part of ARTSY. No requirements have therefore been defined for the BC. It is recommended that a possible integration of a BC into the system is examined further.

3 Specific Requirements

This section contains all software requirements. The requirements are divided into three classes: Essential requirements, conditional requirements and optional requirements. Essential requirements must be fulfilled, the software will not be acceptable otherwise. Conditional requirements enhances the software product but would not make it unacceptable if they are absent. Optional requirements may or may not be worthwhile and should be considered as design decision guidelines.

Some special requirements are declared for an ARTSY implementation intended for images. Those requirements are separated from the other and are called ARTSYi requirements. The ARTSYi requirements are also divided into essential, conditional and optional requirements.
3.1 External Interfaces

Conditions for a Prototype Implementation

This document presents two sets of requirements; a full set of requirements that apply to a full scale ARTSY implementation and a reduced set of requirements that apply to an ARTSY prototype implementation. The latter is a subset of the former.

The requirements specified in this document are either marked with a (p) or not marked at all. Those that are marked forms the reduced set of requirements that apply to the prototype implementation. The full set of requirements includes all requirements in this document.

To summarize the most important differences between the full implementation and the prototype implementation some general constraints for the prototype are presented below. The basic philosophy when putting constraints on the prototype was to give the highest priority to those functions that are vital to ARTSY and that are technically important. Functions that are of interest when demonstrating the system and that are relatively easy to meet has also been given a high priority. The following constraints apply to the ARTSY prototype:

1. The prototype does not have to implement a Public Key Registry.
2. Certificates does not have to be used in the prototype.
3. The prototype does not have to support multiple security levels. The secure security level (see section 3.4.1) can be used for all components.
4. The prototype does not have to support more than one TS.
5. The RCi module does not have to be inserted into a printer.

3.1 External Interfaces

This section contains specific requirements for the user interfaces, the software interfaces and the communications interfaces.

3.1.1 User Interfaces

All components have a user interface that manages instructions from and displays information to a user. Only a few fundamental requirements for these interfaces are presented below. No HCI related requirements are defined. It is recommended that this SRS is complemented with additional requirements that cover HCI aspects.

Essential Requirements

E-1.1.1 (p) All RCs shall implement a user interface through which a user at the RC can issue an Information Request.
E-1.1.2 (p) All RCs shall implement a user interface through which a user at the RC can request a Tender Invitation.

E-1.1.3 (p) All RCs shall implement a user interface through which a user at the RC can request an Order.

E-1.1.4 All CPs shall implement a user interface through which a user at the CP can add new Content Representations and Content Meta Data Files.

E-1.1.5 All TSs shall implement a user interface through which a user at the TS can find information about every Transaction and every Tender created.

Optional Requirements

O-1.1.1 All VAA-providers shall implement a user interface through which a user at the TS can add new VAA.s.

O-1.1.2 All VAA-providers shall implement a user interface through which an owner of a VAA can get information about all the transactions the VAA has participated in.

3.1.2 Software Interfaces

All components within ARTSY store information at different times. Tender Invitations are stored at the CP, Tenders are stored at the RC and the TS as long as they are valid, all components store logs of old transactions etcetera. A DBMS (database management system) is used to ease the administration of the information storage. How databases shall be used are further described in section 3.6 and section 3.7.

To issue an Information Request an RC must know the id and the location of a CP and the id of a Content file. This data can be entered manually by a user at the RC. The data can however also be entered automatically by an external program, by a BC. The BC allows the user at the RC to browse images at a BC and make an Information Request from the BC. The BC sends the Content Id to the RC where the user confirms the Information Request before it is sent to the CP (see figure 11). The BC is not a part of ARTSY, but ARTSY includes support to interact with third-party BCs.

Essential Requirements

E-1.2.1 All ARTSY components shall use a DBMS to store and access required data files\(^3\).

E-1.2.2 Each DBMS used in ARTSY shall only be accessible by the component that the DBMS belongs to.

\(^3\)See section 3.6
3.1 External Interfaces

E-1.2.3 The RC shall include an external interface for interactions with remote BCs.

E-1.2.4 The external BC interface at the RC shall include functionality to receive an id number of a CP along with an id number of a Content Representation.

E-1.2.5 The external BC interface at the RC shall include functionality to prepare an Information Request using data which has been received from a BC.

E-1.2.6 An RC shall only be able to issue an Information Request if a user at the RC confirms this. In other words, all Information Requests must be confirmed by a user locally.

3.1.3 Communications Interfaces

In order for ARTSY to operate flawlessly it must use a network that implements a reliable transport protocol. The sending side must know if a sent message was delivered or not. TCP/IP is an example of a protocol that can be used.
3 SPECIFIC REQUIREMENTS

Essential Requirements

E-1.3.1 (p) ARTSY shall use a network that implements a transport protocol that uses an automatic repeat request algorithm (ARQ)\textsuperscript{4}.

3.2 System Architecture

Essential Requirements

E-2.1.1 (p) Each CP shall always communicate with the same TS.

E-2.1.2 (p) Each RC shall be able to communicate with any TS.

Optional Requirements

O-2.1.1 Each VAA-provider shall always communicate with the same TS.

O-2.1.2 Each VAA shall reside on a VAA-provider.

O-2.1.3 Each VAA shall only be able to communicate with the VAA-provider it resides on.

3.3 System Features

The main features of the system are described in detail below.

3.3.1 Information Request

During an Information Request an RC requests information from a CP, on some Content that the RC might want to reproduce. Information in this case means meta data that describes the Content. If an image is to be transmitted the meta data contains a description of the image, for example its resolution, its physical size etcetera. The TS connected to the CP is not involved in the Information Request. In a large system where many CPs are connected with a TS, the TS might become a bottleneck. Since the TS is not involved it is not possible to specify the transaction cost in the Information Response.

Purpose of feature The purpose of the Information Request is to provide an RC with relevant Content information, making it possible for the RC to decide whether to continue or not with the transaction. Critical Content information is for example the available encoding formats. There is no point in ordering Content that cannot be reproduced since its format is not supported.

The Information Response informs the RC what it can order. Apart from the available encoding formats the response also contains information about alternative representations of the Content. An alternative representation of a color image could for example be a gray scale image. Another alternative representation of an image could be a sub area of the image.

\textsuperscript{4}TCP meets this requirement
3.3 System Features

Stimulus/Response sequence An RC sends an information request to a CP and the CP answers this request (see figure 12).

![Diagram of Stimulus/Response sequence](image)

**Figure 12: Information request**

**Messages** The messages that are sent are:

1. Information Request

2. Information Response

**Essential Requirements**

E-3.1.1 (p) An RC shall send an Information Request to a CP when a user at the RC requests this.

E-3.1.2 (p) A CP shall respond to an Information Request from an RC. The response shall include the Content Meta Data file belonging to the requested Content.

E-3.1.3 (p) When an RC has received an Information Response, the user at the RC shall be notified.

**Optional Requirements**

O-3.1.1 Every unique Information Response an RC receives shall be stored at the RC for at least 45 days.

O-3.1.2 If an RC does not receive a response to an Information Request within one minute the request shall be considered unsuccessful, the user shall then be notified and questioned to try again.
3.3.2 Tender Submission

During a Tender Submission a Tender is invited and submitted. Three components are involved in the procedure: An RC that invites the Tender, a CP that submits a pre-Tender and eventually the TS connected to the CP that creates the Tender based on the pre-Tender. The Tender Invitation contains a request for a specific Content Representation. It specifies the requested encoding format and it can also specify other things. If an Image Representation is to be transmitted the Tender Invitation could for example declare that only a sub area of the Image Representation shall be transmitted, or an Image Representation with a reduced color depth.

Purpose of feature The purpose of the Tender Submission is to provide an RC with a Tender. The Tender contains an approximative price for the Content defined in the Tender Invitation. The price is an approximation of the total price expressed in the currency of the RC. The total price is the sum of the fee that the TS charges for managing the transaction, the fee that the CP charges for transmitting and letting the RC reproduce the requested Content Representation and the possible fees charged by any participating VAAAs. Given this information the RC decides whether to order the Content Representation or not. If the RC decides to make an Order, the RC must use the Tender as a starting point for the Order.

Stimulus/Response sequence An RC sends a Tender Invitation to a CP, the CP creates a pre-Tender and submits it to its TS, the TS creates a Tender based on the pre-Tender and submits the Tender to the RC (see figure 13).

```
| RC | 1 |
|------------------|
|                |
|                |
|                |
|                |
| 3              |
|                |
|                |
| 2              |
|                |
|                |
|                |
|                |
|                |
|                |
| TS             |
```

Figure 13: Tender Submission

5 Possibly more than three if VAAs are employed
3.3 System Features

Messages  The messages that normally⁶ are sent are:

1. Tender Invitation
2. pre-Tender
3. Tender

Essential Requirements

E-3.2.1 (p)  An RC shall send a Tender Invitation to a CP when a user at the RC requests this.

E-3.2.2 (p)  It shall be possible to use an Information Response as a starting point for a Tender Invitation.

E-3.2.3 (p)  When a CP receives a Tender Invitation from an RC, the CP shall check if the Content Representation requested in the Tender Invitation is available at the CP.

E-3.2.4 (p)  If a CP finds that the Content Representation demanded in an Information Request is unavailable at the CP, the CP shall inform the RC that the demanded Content Representation is unavailable.

E-3.2.5 When a CP receives a Tender Invitation from an RC, the CP shall compare the security level stated for the RC in the Tender with its own security level (see section 3.4.1).

E-3.2.6 If a CP finds that the security level stated in a Tender Invitation is lower than the lowest allowed security level for the CP, the CP shall send an Error Message to the RC that sent the Tender Invitation.

E-3.2.7 If a CP finds that the security level stated in a Tender Invitation is greater than or equal to the lowest allowed security level for the CP and that the requested Content Representation is available at the CP, then the CP shall create a pre-Tender and submit it to the TS connected to the CP.

E-3.2.8 If a TS receives a pre-Tender from a CP and the forthcoming transaction involves money, then the TS shall examine the security level defined in the pre-Tender.

E-3.2.9 If a TS finds that the security level defined in a pre-Tender is below the lowest allowed security level for transactions involving money and if the forthcoming transaction involves money then the TS shall send an Error Message to both the RC and the CP.

⁶If no error(s) occur.
E-3.2.10 If a TS receives a pre-Tender from a CP and the forthcoming transaction does not involve money then the TS shall create a Tender based on the pre-Tender and then submit this Tender to the RC that invited it.

E-3.2.11 If a TS finds that the security level defined in a pre-Tender is greater than or equal to the lowest allowed security level for transactions involving money then the TS shall create a Tender based on the pre-Tender and then submit this Tender to the RC that invited it.

E-3.2.12 A Tender which has been sent from a TS to an RC shall be stored at the TS as long as it is valid.

E-3.2.13 A Tender which has been used to make an Order shall be stored at the TS for at least 45 days.

E-3.2.14 If an RC is informed by a CP that a demanded Content Representation is unavailable at the CP the RC shall notify the user at the RC.

Optional Requirements

O-3.2.1 If an RC does not receive a Tender within one minute after a Tender Invitation has been sent the invitation shall be considered unsuccessful and the user shall then be notified by the RC and questioned to try again.

O-3.2.2 Before an RC sends a Tender Invitation to a CP, the RC shall compare the requested Content Representation in the Tender Invitation with the Reproduction Capabilities of the RC.

O-3.2.3 If an RC finds that its Reproduction Capabilities are insufficient for reproducing the Content Representation requested in a Tender Invitation, the RC shall issue a warning to the user at the RC and demand a confirmation of the Tender Invitation before it is sent.

O-3.2.4 If a VAA is requested in the Tender Invitation the pre-Tender shall be sent to the VAA-provider before it is sent to the TS.

O-3.2.5 If one or more VAA's are requested in a Tender Invitation the VAA-provider shall add the fees of the requested VAA's to the pre-Tender and send it to the TS.

3.3.3 Transaction

The transaction phase starts when an RC sends an Order to a TS. The Order is based on a Tender which must have been received from a CP before the transaction. When a TS has received an Order the TS sends one Transaction Start Message to each client. The CP starts to transmit a Content Representation to the RC when it receives its Transaction Start Message (see figure 14). The exact Content transmission procedure is not specified within this document. Content
transmission can be managed in more than one way. The most obvious solution is to let the CP send one large message to the RC that contains the entire Content Representation file. This might however not be a practicable solution if the Content Representation file is large. A better solution would be to let the CP inform the RC when it is ready to transmit and then leave the control to the RC. The RC would then request the first part of the (large) Content Representation file, say the first 4 kB. When the RC has finished processing the first part of the file (saved it to disk, sent it to the reproduction unit or something else), the RC would request the next part of the file. This would continue until the entire file would have been transmitted to the RC.

When the CP has transmitted the entire Content Representation, it sends a feedback message to the TS. The RC correspondingly sends another feedback message to the TS when it has received the entire Content Representation.

A feedback message contains one of the following: Ok or Failed. The TS decides the result of the transaction given the feedback messages from the RC and the CP. The TS then informs the clients of the result by sending a Transaction Result Message to each of them (see figure 15).

The result of the transaction result is one of the following: Completed, Cancelled or Corrupted. If the result is Complete the RC can be charged for the transaction. If the result is Cancelled the RC can not be charged for the transaction. A Corrupted transaction must be manually investigated further to decide whether the transaction should be Completed or Cancelled.

An example of how different feedback messages give rise to different transaction results is shown in table 1. As long as the RC sends Ok the transaction result will be Completed. If a Failed message is received from the CP and no message has been received from the RC the transaction will be Cancelled. The situation where the RC reports Failed and the CP reports Ok is unlikely to occur. The CP reports Ok when it is sure that the Content Representation has been received correctly by the RC. If the RC reports Failed it means that something happened after the Content Representation was received but before it was reproduced. This must be investigated further before it can be decided whether to charge the RC for the transaction or not.

**Purpose of feature** The purpose of the transaction is to transmit a Content Representation from a CP to an RC. The transaction phase completes the transaction procedure.

**Stimulus/Response sequence** An RC sends an Order to a TS. The TS sends two Transaction Start Messages, one to the RC and one to the CP specified in the Order. The CP transmits a Content Representation to the RC and the sends a Feedback Message to the RC. The RC receives the Content Representation and then sends a Feedback Message to the TS. Eventually the TS sends a Transaction Result message to the two clients.
3 SPECIFIC REQUIREMENTS

Figure 14: Transaction - Step 1

Messages  The messages that normally\(^7\) are sent are:

1. Order
2. Transaction Start Message
3. Transaction Start Message
4. Content Transmission
5. CP Feedback
6. RC Feedback
7. Transaction Result

Figure 15: Transaction - Step 2

\(^7\text{If no error(s) occur.}\)
Table 1: Transaction result

<table>
<thead>
<tr>
<th>Message from RC</th>
<th>Message from CP</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td>OK</td>
<td>Completed</td>
</tr>
<tr>
<td>OK</td>
<td>FAILED</td>
<td>Completed</td>
</tr>
<tr>
<td>FAILED</td>
<td>OK</td>
<td>Corrupted</td>
</tr>
<tr>
<td>FAILED</td>
<td>FAILED</td>
<td>Cancelled</td>
</tr>
<tr>
<td>-</td>
<td>FAILED</td>
<td>Cancelled</td>
</tr>
</tbody>
</table>

### 3.3 System Features

#### Essential requirements

**E-3.3.1 (p)** An RC shall send an Order to a TS when a user at the RC requests this.

**E-3.3.2 (p)** A Tender shall serve as the starting point for an Order.

**E-3.3.3 (p)** An Order shall contain a reference to the Tender that served as the starting point for the Order.

**E-3.3.4 (p)** It shall be possible to use a Tender as starting point for an Order only once. It shall not be possible to use the same Tender for more than one Order. A Tender is either unused or used and a used Tender cannot serve as a starting point for an Order.

**E-3.3.5 (p)** A Tender that is referred to in an Order shall be stored at the RC for at least 45 days.

**E-3.3.6 (p)** Given an Order a TS shall send one Transaction Start Message to the RC and one Transaction Start Message to the CP.

**E-3.3.7 (p)** A Transaction Start Message which is sent to an RC, shall include a reference to the Tender that served as the starting point for the Order issued by the RC.

**E-3.3.8 (p)** A Transaction Start Message which is sent to a CP, shall include a reference to the pre-Tender that the CP created.

**E-3.3.9 (p)** A Tender that is referred to in a Transaction Start Message shall be stored at the TS for at least 45 days.

**E-3.3.10 (p)** If a TS fails to send a Transaction Start Message to a client involved in a transaction, the TS shall inform the other client that the transaction is cancelled.

**E-3.3.11 (p)** Given a Transaction Start Message a CP shall transmit the Content Representation defined in the Tender, to the RC specified in the Tender.
E-3.3.12 (p) A Content Transmission from a CP to an RC shall include a reference to the Tender Invitation or the Tender that preceded the transmission.

E-3.3.13 (p) When a CP has transmitted a Content Representation, the CP shall send a feedback message to the TS defined in the Tender. This message shall describe if errors occurred or not during the transmission of the Content Representation.

E-3.3.14 (p) When an RC has received a Content Representation, the RC shall send a feedback message to the TS defined in the Tender. This message shall describe if errors occurred or not during the reception of the Content Representation.

E-3.3.15 (p) When a TS has decided the result of a transaction (see an example in table 1), the participating RC and CP shall be informed of the result.

E-3.3.16 (p) An ARTSY component (RC, CP, TS) shall be able to store the result of a transaction only once. It shall not be possible to change a transaction result that has been stored in the database at a component.

Conditional requirements

C-3.3.1 (p) An RC shall send its feedback message to the TS after it has reproduced (or tried to reproduce) a Content Representation. The message shall include information about the reproduction process, it shall describe if errors occurred or not when reproducing the Content.

C-3.3.2 (p) If the system includes an RCI module only the RCI module shall be able to send an RC Feedback message.

C-3.3.3 If a TS does not receive feedback messages from both the CP and the RC within 12 hours after the transaction has been initiated the TS shall send a status request message to the client(s) that has not sent a feedback message.

C-3.3.4 If a client does not respond to a status request message within 10 minutes after a TS has sent it, the transaction shall be terminated and the TS shall make a second entry in the transaction log file describing the course of events. The TS shall also inform the clients that the transaction has been terminated by sending a Transaction Result Message.

Optional requirements

O-3.3.1 The RC shall save the progress of the reproduction process to a file every half second.
3.4 System Security

O-3.3.2 If an RC manages to reproduce only a part of a Content Representation, the percentage share of the Content Representation that was reproduced shall be included in the feedback message to the TS.

O-3.3.3 If one or more VAs participates in a transaction, a separate Transaction Start Message shall be sent from the TS to their VAA-provider.

3.4 System Security

The Internet is an insecure media and critical information must be protected when transmitted over the Internet. Content Providers must be sure that only the intended receiver receives the transmitted Content. CPs and TSs must be sure that an Order comes from the stated sender and not from anybody else.

Different components communicate with each other by sending messages. A message in ARTSY has the following structure:

\[
\text{Message} = \begin{cases} \text{Information} & \text{[Extra]} \text{ [Signature]} \end{cases}
\]

A message contains of up to three different parts. The signature part and the extra part (within brackets) is optional. The part without brackets are mandatory. Information is the information the message carry. The message type is included in the information part of the message. For example: Transaction Feedback, Transaction Start or Order. Other data included in the Information part vary between each message type but it can for example be a file (any of the data files mentioned in section 3.6, for example: pre-Tender or Tender) or a reference to a file of some kind. The information part in a Transaction Start Message could look like this:

\begin{center}
\begin{tabular}{c}
\hline
\text{Information} \\
\text{TRANSACTION\_START} & \text{t-300} & 4E6F772069732074 \\
\text{TS\_tender\_id} & \text{session\_key} \\
\hline
\end{tabular}
\end{center}

Signature is the signature of the sender and is used for authentication. Extra is implementation specific data that must be sent unencrypted.

If the secure security level is used all messages except Information Request and Information Response are secure. The different parts in a message are encrypted differently in a secure message. The information part is encrypted with an asymmetric algorithm using the public key of the receiver. The signature is a signed hash value where the hash value is computed over the the information and the extra parts of the message. The extra part of the message is not encrypted.

3.4.1 Security Levels

Different Content Providers might need different security. One Content Provider might not want any security at all on its messages. Another Content Provider
might want maximum security on all its messages. The reason for a client to choose a low security level is that a high security level has a couple of drawbacks. The installation process is more complex and a more computer power is required.

**Essential requirements**

E-4.1.1 ARTSY shall support at least two different security levels: The insecure security level and the secure security level.

E-4.1.2 (p) At the secure security level all messages except Information Request and Information Response shall be secure.

E-4.1.3 (p) Each component shall support at least one security level. To support a security level the component shall be able to provide all messages with the required security for that security level.

E-4.1.4 Each TS shall specify the lowest security level required for transactions involving money.

E-4.1.5 (p) A TS that is involved in a secure transaction shall generate a new session key for each new transaction.

E-4.1.6 (p) A TS that is involved in a secure transaction shall send the session key to the CP and the RC involved in the transaction. The session key shall be included in the information part of the Transaction Start Message.

E-4.1.7 (p) The session key sent to the CP shall be used for encrypting the Content Representation. The session key sent to RC shall be used to decrypt the Content Representation.

E-4.1.8 (p) Information that is included in a secure message shall be encrypted with the public key of the receiver.

E-4.1.9 (p) The Content Representation that is transmitted during a secure transaction shall be encrypted with a session key by the sending CP.

E-4.1.10 (p) A hash value shall be produced for information and extra included in a secure message by applying a hash algorithm to the unencrypted information and extra.

E-4.1.11 (p) A hash value produced by a hash algorithm shall be encrypted with the private authentication key of the sender and placed in the secure message as a signature.

E-4.1.12 (p) Information that are received as part of a secure message shall be decrypted with the receivers private confidentiality key.

E-4.1.13 (p) The signature of a secure message that has been received shall be decrypted with the public authentication key of the sender of the message.
3.4 System Security

E-4.1.14 (p) A hash value shall be produced for the information and extra that are received as a part of a secure message. This shall be done by applying a hash algorithm to the unencrypted information and extra. The hash value shall then be compared with the decrypted signature included in the secure message. If the values do not match an Error Message shall be sent to the sender of the message.

E-4.1.15 (p) The receiver of a secure message shall use the hash algorithm stated in the certificate of the sender of the message, to produce a hash value for the message.

Conditional requirements The following requirements are applicable if the RC is divided into an RCI and an RCo module.

C-4.1.1 (p) The session key in a secure Transaction Start Message shall be encrypted with the public key of the receiving RCI.

C-4.1.2 (p) The reference to a Tender included in a Transaction Start Message for an RC, shall be encrypted with the public key of the sending TS\(^8\).

C-4.1.3 (p) A secure Transaction Start Message that is designated for a RC shall be sent to the RCo module of the RC.

C-4.1.4 (p) When the RCo module of an RC receives a secure Transaction Start Message it shall forward the encrypted session key and the encrypted transaction reference to the RCI module of the RC.

Optional requirements

O-4.1.1 The security levels shall be numbered from one and up. The security level with the highest security has the highest number.

O-4.1.2 Each security level that is not the highest level, shall be a subset of the level above.

O-4.1.3 The insecure security level shall not provide any security at all. No messages shall be encrypted, no messages shall be authenticated, Content Representations shall be transmitted unencrypted.

O-4.1.4 If a VAA is requested in a secure transaction, two different session keys shall be produced by the TS. The first session key shall be sent to the CP and the VAA-provider in the Transmission Start message and the second session key shall be sent to the VAA-provider and the RC in the transmission start message.

---

\(^8\)This reference will be forwarded to the RCI from the RCo. By encrypting the transaction reference an adversary cannot forward another transaction reference without RCI discovering it. The RCI must be certain of which transaction the session key is for. Otherwise the adversary might send several messages including the same session key but different transaction references and could then send the same encrypted Content Representation and get it printed several times.
O-4.1.5 The VAA shall use the first session key in a Transaction Start Message to decrypt a Content Representation received from a CP.

O-4.1.6 The VAA shall use the second session key in a Transaction Start Message to encrypt Content Representation before sending it to an RC.

O-4.1.7 Each VAA-provider shall support the secure security level.

3.4.2 Security Mechanisms

Essential requirements

E-4.2.1 (p) A TS shall generate session keys in a manner which is unpredictable. If a pseudo random number generator is used to generate the session keys, the input to the generator shall be truly random\(^9\).

E-4.2.2 (p) The algorithm used for encryption and decryption together with the session keys shall have been announced and approved by recognized authorities in computer security\(^10\).

E-4.2.3 (p) All authentication and encryption of information and files in ARTSY shall be performed using an asymmetric algorithm.

E-4.2.4 (p) An asymmetric algorithm used in ARTSY shall have been announced and approved by recognized authorities in computer security\(^11\).

E-4.2.5 (p) A client that has chosen a secure security level shall use a private key to sign and to encrypt secure messages.

E-4.2.6 (p) A hash algorithm used in ARTSY shall be unkeyed.

E-4.2.7 (p) Given a hash value it shall be theoretically proven that it is infeasible to determine the input to the hash algorithm used in ARTSY.

E-4.2.8 (p) It shall be computationally infeasible to find two inputs to the hash algorithm used in ARTSY that have the same hash value\(^12\).

E-4.2.9 (p) The security mechanisms shall be implemented in such a way that ARTSY shall remain secure even if an adversary gets access to the source code of ARTSY. In other words, ARTSY cannot be based on that the algorithms themselves are secret and need to be hidden.

\(^9\)A truly random number comes from something physical. A solution is to maintain a pool of information pertaining to physical parameters, properties, and activity of the system. Input to the pool can be: Time between keystrokes, timing of disk interrupts, the number of page faults, number of network packages arrived. A hardware-based number generator can also be used.

\(^10\)Examples of algorithms that are possible to use at the time of writing: Triple-DES, IDEA.

\(^11\)An algorithm that would be possible to use at the time of writing is the RSA 1024-bit algorithm.

\(^12\)An example of a hash algorithm that meets all requirements is the MD5 algorithm.
3.4 System Security

Conditional requirements  It is vital that the private key of each component never is compromised. A way to protect the private keys is to let each component possess two private keys instead of one. One key would then be used to sign messages (authentication key) and one would be used to decrypt messages (confidentiality key). The advantage of using two keys instead of one is that the confidentiality key never is used for signing and no ciphertext encrypted with the private confidentiality key is ever available for an adversary.

If an algorithm is used with the electronic codebook mode the ciphertext will be vulnerable to a known plaintext attack.

C-4.2.1 The electronic codebook (ECB) mode shall not be used with any encryption algorithm in ARTSY.

C-4.2.2 A client that has chosen a secure security level shall use a private authentication key to sign secure messages.

C-4.2.3 A client that has chosen a secure security level shall use a private confidentiality key to encrypt secure messages.

C-4.2.4 The RC shall possess two private authentication keys and two private confidentiality keys. One pair for the RCi and one pair for the RCo.

C-4.2.5 Each TS shall use private authentication key to sign secure messages.

C-4.2.6 Each TS shall use private confidentiality key to encrypt secure messages.

3.4.3 Public Key Registry

In ARTSY there is one Public Key Registry where all public keys to all components are stored. To authenticate a component the public key is needed and it can be downloaded from the Public Key Registry. Each component in the Public Key Registry has an entry where the unique id, the public key and a signature is stored. The Public Key Registry is hosted at a TS and all public keys in the registry are signed by the TS. When the ARTSY software is installed at a party the public key of the TS hosting the Public Key Registry is installed. The TS hosting the Public Key Registry is called the Trusted Party. There must be a way for a component to revoke a private key that has been compromised. A revoked key is deleted from the Public Key Registry and all components are notified of the revocation. The notification can be solved implicitly if each component must contact the Public Key Server each time they use a public key. Private keys that are compromised are removed from the Public Key Server.

A Certificate Authority (CA) could be used instead of a Public Key Registry. The CA could be a part of ARTSY or an external CA could be used. The essential requirements are formulated so that any of the solutions Public Key Registry and a CA can be used.
Essential requirements

E-4.3.1 One Trusted Party shall be used in ARTSY.

E-4.3.2 The public key belonging to the Trusted Party in ARTSY shall be installed on all components that has chosen the secure security level.

E-4.3.3 A public key shall be stored in an entry in a public database. In the same entry shall also the unique id of the component owning the key and a signature made by the Trusted Party be stored.

E-4.3.4 All ARTSY components that have chosen the secure security level shall be able to access and download all entries belonging to all other ARTSY components that has chosen the secure security level.

E-4.3.5 A signature by the Trusted Party is made by running the information stored in an entry in the public database through a hash algorithm which produces a hash value. The hash value shall be encrypted with the private key of the Trusted Party and put in the same entry.

E-4.3.6 If there are more than one security level requiring public keys the chosen security level shall be stored in each entry in the public database.

E-4.3.7 Each component shall know which hash algorithm each component uses for making signatures.

E-4.3.8 Each component shall know which algorithm each component uses for encryption of Information and authentication of Messages.

E-4.3.9 Each party in ARTSY that has chosen the secure security level shall be able to revoke its public key.

E-4.3.10 Revocation information about all public keys used in ARTSY shall be provided to all components that use the secure security level.

Optional requirements If a CA is used in ARTSY the following requirements can be used.

O-4.3.1 Each party in ARTSY for which CA has generated a certificate shall be able to send a Revocation Request of its own certificate to the CA.

O-4.3.2 If there are more than one security level requiring certificates the chosen security level shall be stated in the certificate.

O-4.3.3 Each component in ARTSY shall be able to automatically download a certificate for any party connected to ARTSY that has a valid certificate.

O-4.3.4 The CA shall maintain status information about a certificate during its validity period, and provide revocation information to all components in ARTSY.
3.5 System Administration

O-4.3.5 In each certificate there shall be an id of the certificate possessor. The id of the party shall be unique to all parties (CP, RC, TS) within ARTSY.

O-4.3.6 In each certificate there shall be a name of the certificate possessor.

O-4.3.7 In each certificate there shall be a time interval during which the certificate is valid.

O-4.3.8 In each certificate it shall be stated the algorithm used by the CA to digitally sign the certificate.

O-4.3.9 In each certificate the CAs digital signature of the certificate shall be included.

O-4.3.10 In each certificate it shall be stated the algorithm(s) used for authentication and encryption of information and files.

O-4.3.11 In each certificate it shall be stated the hash algorithm used by the certificate possessor to make digital signatures.

O-4.3.12 Each certificate shall include the public key of the certificate possessor.

O-4.3.13 Each certificate shall include the public encryption key of the certificate possessor.

O-4.3.14 Each certificate shall include the public authentication key of the certificate possessor.

O-4.3.15 A Certificate issued for an RC shall include both the public key for the RCo and the public key for the RCi.

3.5 System Administration

All components within ARTSY store information at different times. The requirements for the information storage and administration are described below.

3.5.1 Reproduction Client

An RC stores its Reproduction Capabilities. This information is used during the Information Request phase, see section 3.3.1. The Reproduction Capabilities needed to be stored are described in section 3.6.6.

Optional requirements

O-5.1.1 An RC shall store its Reproduction Capabilities.

O-5.1.2 A modification of the Reproduction Capabilities stored at an RC shall not require a software re-installation of the RC software. If the RC is divided into an RCo module and an RCi module, this requirement applies to the RCo module.
O-5.1.3 It shall be possible to update the Reproduction Capabilities at an RC at run-time, without shutting the RC down.

3.5.2 Content Provider

A CP stores Content, Content Meta Data and a Content Price List. The Content Meta Data and the Price List are further described in section 3.6.1 and section 3.6.5.

Essential requirements

E-5.2.1 (p) All Content at a CP shall be described by a Content Meta Data file, stored at the CP.

E-5.2.2 (p) A CP shall store a Content Price List.

E-5.2.3 (p) Content Representations that are referred to in a Price List at a CP shall be available at the CP.

E-5.2.4 (p) Content that is referred to in a Content Meta Data file shall be available at the CP.

Conditional requirements

C-5.2.1 (p) It shall be possible to update the Price List at a CP at run-time, without shutting the CP down.

C-5.2.2 (p) It shall be possible to update a Content Meta Data file at a CP at run-time, without shutting the CP down.

C-5.2.3 It shall be possible to add, remove and update Content at a CP at run-time, without shutting the CP down.

Optional requirements

AO-2.1 A CP shall be able to get a list from the TS of available VAA.s.

3.5.3 Transaction Server

Information about all CPs that are connected to a TS is stored in a database at the TS. The Content of this database is described in section 3.7.3.

Essential requirements

E-5.3.1 It shall be possible to add, remove and modify entries in a CP List at a TS at run-time, without shutting the TS down.
3.6 Required Data Files

This section contains the requirements on the data files that are used. Most of the files are used in communication between different parties. The files that exist are:

- Content Meta Data
- Tender Invitation
- pre-Tender
- Tender
- Price List
- Reproduction Capabilities

The following files are sent in messages: Content Meta Data (Information Response message), Tender Invitation (Tender Invitation message), pre-Tender (pre-Tender message) and Tender (Tender message). Each CP, VAA and TS has one Price List file and each RC has one Reproduction Capabilities file.

3.6.1 Content Meta Data

A Content Meta Data file is created by each CP for each Content that the CP provides. One Original can give rise to different Content Representations. For example: A certain Original is photographed with three different light settings: Normal light, ultra violet and infra red. The three images can be stored as three or more Image Representations. It might be possible to order a certain part of an Image Representation or to get an Image Representation encoded in a certain data format. Each Content Meta Data file describes the Content Representations that are available for some specific Content.

Essential requirements

E-6.2.1 (p) Each Content Meta Data file shall contain the id of the CP that owns the Content.

E-6.2.2 (p) Each Content Meta Data file shall contain a name that is associated with the Content.

E-6.2.3 (p) Each Content Meta Data file shall contain an id that identifies the Content.

E-6.2.4 (p) Each Content Meta Data file shall contain the data formats in which Content Representations are available.
Optional requirements

O-6.2.1 Each Content Meta Data file shall contain the identities of the available VAAs.

O-6.2.2 Each Content Meta Data file shall contain a description of the available VAAs.

Conditional requirements for ARTSYi

C-6.2.1 Each Content Meta Data file shall contain the resolution(s) in pixels of the Image Representation(s).

C-6.2.2 The Content Meta Data file shall contain the color depth(s) of the Image Representation(s).

C-6.2.3 The Content Meta Data file shall contain the resolution(s) in dots per inch for the Image Representation(s).

C-6.2.4 The Content Meta Data file shall contain the height and width of the Original in millimeters.

C-6.2.5 The Content Meta Data file shall contain descriptions of all Content Representations.

Optional requirements for ARTSYi

Oi-6.2.1 Each Content Meta Data file shall contain the following information about the Image:

- The artist that produced the Original.
- The year when the production of the Original was started.
- The year when the production of the Original was completed.
- The technique used by the artist when producing the Original.
- The technique used for acquiring the Content.

3.6.2 Tender Invitation

An RC creates a Tender Invitation to be able to invite a tender. A Tender Invitation is sent from an RC to a CP. A Content Representation is defined in a Tender Invitation and in the most basic case it just consists of a Content Representation id. A Content Representation id represents a Content Representation predefined by the CP. In a more complex case the Tender Invitation defines a particular area of an Image as an Image Representation.
3.6 Required Data Files

Essential requirements

E-6.3.1 (p) The Tender Invitation shall contain an id that uniquely identifies it within the RC.

E-6.3.2 (p) The Tender Invitation shall contain the id of the RC that created it.

E-6.3.3 (p) The Tender Invitation shall contain the id of the CP that will receive the it.

E-6.3.4 (p) The Tender Invitation shall contain an id that identifies the Content that is treated in the Tender Invitation.

E-6.3.5 (p) The Tender Invitation shall contain information that defines the requested Content Representation.

E-6.3.6 (p) The number of requested reproduction copies shall be included in the Tender Invitation.

E-6.3.7 The Tender Invitation shall include the security level of the RC.

Conditional requirements

C-6.3.1 (p) The Tender Invitation shall include the currency that the RC uses.

C-6.3.2 The Tender Invitation shall include the VAAs that shall be used during the Content Transmission.

Conditional requirements for ARTSYi

Ci-6.3.1 It shall be possible to declare a requested resolution, in pixels, of the requested Image Representation in the Tender Invitation.

3.6.3 pre-Tender

A pre-Tender is created by a CP as a response to a Tender Invitation. A pre-Tender is sent from a CP to a TS. The TS creates a Tender from the pre-Tender and sends it to the RC that invited the Tender.

Essential requirements

E-6.4.1 (p) The pre-Tender shall include an id that uniquely identifies it within the CP.

E-6.4.2 (p) The pre-Tender shall include the id of the Tender Invitation that initiated it.

E-6.4.3 (p) The pre-Tender shall include the security level that will be used in the transaction.
3 SPECIFIC REQUIREMENTS

E-6.4.4 (p) The pre-Tender shall include the fee that the CP charges the RC.

E-6.4.5 (p) The Tender Invitation shall include the currency that the CP uses.

E-6.4.6 (p) The pre-Tender shall include the time when the pre-Tender ceases being valid.

E-6.4.7 (p) The pre-Tender shall include all essential information required in the Tender Invitation above (see section 3.6.2).

3.6.4 Tender

A Tender is created by TS and initiated by a pre-Tender from a CP.

Essential requirements

E-6.5.1 (p) The Tender shall include an id that uniquely identifies it within the TS.

E-6.5.2 (p) The Tender shall include the id of the TS that created it.

E-6.5.3 (p) The Tender shall include the id of the CP that provides the Content.

E-6.5.4 (p) The Tender shall include the time interval during which it is valid.

E-6.5.5 (p) The Tender shall include a price for the transaction.

E-6.5.6 (p) The Tender shall include all essential information required in the Tender Invitation above (see section 3.6.2).

Conditional requirements

C-6.5.1 (p) The Tender shall include the fee that the CP charges the RC for transmitting the Content Representation and the currency in which this fee is stated.

C-6.5.2 (p) The Tender shall include the fee that the TS charges the RC for mediating the transaction and the currency in which this fee is stated.

C-6.5.3 (p) The Tender shall include an approximate price for the transaction. This price shall be stated in the currency that RC uses.

Optional requirements

O-6.5.1 The Tender shall include the total price for VAAs that will be used in the transaction.
3.6.5 Price List

Each component that offers a service of some kind and charges money for it, possesses a price list. A price at a CP contains information about Content Representation prices. The price given in each pre-Tender can be derived from the price list and the Tender invitation. A TS can have a quite simple price list and charge the same amount for each transaction. Or it can charge an amount that is relative to the CP fee. Each VAA also has a price list where the price for each available service is stated.

Essential requirements

E-6.6.1 (p) It shall be possible to derive a price for each pre-Tender from the CP price list and the Tender Invitation.

E-6.6.2 (p) It shall be possible to derive a price for each Tender from the pre-Tender and the TS price list.

Optional requirements

O-6.6.1 It shall be possible to derive a price for each pre-Tender from the CP price list, the Tender Invitation and all the price lists of all affected VAA's.

3.6.6 Reproduction Capabilities

Optional requirements

O-6.7.1 The Reproduction Capabilities stored at an RC shall at least declare the supported encoding formats.

3.6.7 RCI-log

An RCI-log is required only if the RCI module exists. The RCI keeps track of the status of the Content Transmission by continuously writing status messages in the RCI-log. If the reproduction unit is shut down the RCI will know, on the next start up, if the reproduction process was finished or not.

Conditional requirements

C-6.8.1 (p) The RCI shall store an id that uniquely identifies the transaction among all transactions within the RC. The id shall be chosen in such a way that it also can be used by the participating TS to uniquely identify the transaction.

C-6.8.2 The RCI shall store the progress of each reproduction (see section 3.3.3).

C-6.8.3 The RCI shall store the number of copies that will be reproduced along with the id that identifies the ongoing transaction.
C-6.8.4 The RCI shall store the number of copies that have been reproduced along with the id that identifies the ongoing transaction.

C-6.8.5 (p) The RCI shall store information that makes it possible to recognize a Transaction Start Message that already has been sent to the RCI\textsuperscript{13}.

3.7 Logical Database Requirements

This section contains the requirements on which data that each ARTSY component must store. When a requirement states that a file shall be stored it does not have to be stored directly in the database. It is sufficient to store a path to the actual file. There are some examples on what tables shall be stored in the database at each component. The data in these tables reflect a snapshot of all databases in a possible ARTSY implementation.

3.7.1 Reproduction Client

These requirements apply to all RCs in ARTSY.

Essential requirements

E-7.1.1 (p) Each Tender shall be uniquely identifiable in the database among all Tenders at the RC.

E-7.1.2 (p) Each Tender Invitation shall be uniquely identifiable in the database among all Tender Invitations at the RC.

E-7.1.3 (p) Each Tender that an RC has received shall be stored in a database at the RC for at least 45 days.

E-7.1.4 (p) Each Tender shall be stored together with information that makes it possible to identify the corresponding Tender Invitation.

E-7.1.5 (p) It shall be possible to identify the Tenders that has been used to make an Order.

E-7.1.6 (p) The status of a Transaction shall be stored in a database.

E-7.1.7 (p) All Transaction status data shall be stored together with information that makes it possible to identify the corresponding Tender.

\textsuperscript{13}This prevents a replay attack. An adversary could catch a Transaction Start Message on its way from the RC to the RCI. The adversary could then send the message to the RCI an unlimited number of times and get an unlimited number of print-outs of the Image Representation. This requirement will prevent an adversary from using an old Transaction Start Message.
Optional requirements

O-7.1.1 (p) Each Tender Invitation shall be stored in a database at the RC for at least 45 days.

O-7.1.2 (p) Each Content Meta Data file that an RC has received shall be stored in a database at the RC for at least 45 days.

O-7.1.3 (p) Each Tender Invitation shall be stored together with information that makes it possible to identify the corresponding Content Meta Data file.

Examples The tables below shows what the tables in the RC database could look like. The bold fields mark primary keys and the italic fields indicate foreign keys.

<table>
<thead>
<tr>
<th>CMD_ID</th>
<th>CONTENT_META_DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>139</td>
<td>c:/cmd/cp10/cmd_5002.xml</td>
</tr>
</tbody>
</table>

Table 2: Content Meta Data (CMD)

<table>
<thead>
<tr>
<th>TENDER_INV_ID</th>
<th>TENDER_INV</th>
<th>CMD_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>c:/ti/ti_11.xml</td>
<td>139</td>
</tr>
</tbody>
</table>

Table 3: Tender Invitation (Ti)

<table>
<thead>
<tr>
<th>TENDER_ID</th>
<th>TENDER</th>
<th>TENDER_INV_ID</th>
<th>USED</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>c:/t/ts2/t_300.xml</td>
<td>11</td>
<td>YES</td>
</tr>
</tbody>
</table>

Table 4: Tender (T)

<table>
<thead>
<tr>
<th>TRANS_ID</th>
<th>TENDER_ID</th>
<th>STATUS</th>
<th>START</th>
<th>END</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>11</td>
<td>processing</td>
<td>2003-02-02</td>
<td>null</td>
</tr>
</tbody>
</table>

Table 5: Transaction

3.7.2 Content Provider

Content Representations are either stored at a CP permanently or generated when needed. An example follows: A Content Representation of the song “Lily Marlene” is stored at a CP in wave format. An RC wants the first three seconds of the song. The CP copies the first three seconds of the stored Content Representation and sends it to the RC. These three seconds forms a new Content Representation of “Lily Marlene”, but this representation is not stored at the
CP, it is generated when it is needed. If the RC would have wanted the whole song in wave format, the stored Content Representation could have been sent directly.

The following requirements apply to all CPs in ARTSY.

**Essential requirements**

E-7.2.1 (p) One Content Meta Data file for each Content shall be stored in a database at the CP.

E-7.2.2 (p) Each Content Meta Data file shall be uniquely identifiable among all Content Meta Data files within the CP.

E-7.2.3 (p) Each Content Representation that is stored at a CP shall be uniquely identifiable among all Content Representations within the CP.

E-7.2.4 (p) Each Content Representation shall be stored together with the id of the corresponding Content in a database at the CP.

E-7.2.5 (p) Each pre-Tender that a CP creates shall be stored in a database at the CP.

E-7.2.6 (p) Each pre-Tender file shall be uniquely identifiable among all pre-Tenders within the CP.

E-7.2.7 (p) The following information about each transaction shall be stored in a database at the CP.

- the date and time when the transaction started
- the date and time when the transaction ended
- the status of the transaction
- information that makes it possible to identify the corresponding pre-Tender

**Examples** The tables below shows what the tables in the CP database could look like. The bold fields mark primary keys and the italic fields indicate foreign keys.

<table>
<thead>
<tr>
<th>CONTENT_ID</th>
<th>CMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>5002</td>
<td>c:/cmd/cmd_5002.xml</td>
</tr>
</tbody>
</table>

Table 6: Content Meta Data
3.7 Logical Database Requirements

<table>
<thead>
<tr>
<th>CONTENT REP_ID</th>
<th>CONTENT REP</th>
<th>CONTENT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>7400</td>
<td>c:/images/dali/time.jpg</td>
<td>5002</td>
</tr>
<tr>
<td>7401</td>
<td>c:/images/dali/time.tif</td>
<td>5002</td>
</tr>
</tbody>
</table>

Table 7: Content Representation

<table>
<thead>
<tr>
<th>pre-T_ID</th>
<th>pre-TENDER</th>
<th>RC</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>c:/preT/preT_100.xml</td>
<td>rc_50</td>
</tr>
</tbody>
</table>

Table 8: pre-Tender

<table>
<thead>
<tr>
<th>TRANS_ID</th>
<th>pre-T_ID</th>
<th>START</th>
<th>END</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>55</td>
<td>100</td>
<td>2002-02-02, 14:22:53</td>
<td>null</td>
<td>transmitting</td>
</tr>
</tbody>
</table>

Table 9: Transaction

3.7.3 Transaction Server

Essential requirements These requirements apply to all TSs in ARTSY.

E-7.3.1 The following information about each CP that is connected to the TS shall be stored in a database at TS:

- The id of the CP.
- The name of the CP.
- The security level of the CP.
- Contact information for the administrator at the CP.

E-7.3.2 (p) Each Tender that a TS created shall be stored in a database at the TS.

E-7.3.3 (p) Each Tender shall be given an id that is unique among all Tenders within the TS.

E-7.3.4 (p) Each Tender shall be stored together with the id of the corresponding pre-Tender.

E-7.3.5 (p) Each Tender shall be stored together with information about whether it is used or not.

E-7.3.6 (p) The following information about each transaction shall be stored in a database at the TS:

- An id that uniquely identifies the transaction within the TS.
- The id of the corresponding Tender.
- The feedback from the CP involved in the transaction.
- The feedback from the RC involved in the transaction.
- The status of the transaction.
3 \textit{SPECIFIC REQUIREMENTS}

\textbf{Conditional requirements} These requirements apply to all TSs in ARTSY.

\textbf{C-7.3.1} The lowest acceptable security level of each CP that is connected to a TS shall be stored in a database at the TS.

The tables below shows what the tables in the TS database could look like. The bold fields mark primary keys and the italic fields indicate foreign keys.

\begin{table}[ht]
\begin{tabular}{|c|c|c|c|c|}
\hline
T\_ID & TENDER & CP & pre-T\_ID & VALID & USED \\
\hline
300 & c:/t/t300.xml & 10 & 100 & 2002-02-22 & YES \\
\hline
\end{tabular}
\caption{Tender}
\end{table}

\begin{table}[ht]
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline
TRANS\_ID & T\_ID & START & END & CP\_MESS & RC\_MESS & STATUS \\
\hline
120 & 300 & 2002-02-02 & null & null & null & started \\
\hline
\end{tabular}
\caption{Transaction}
\end{table}

\begin{table}[ht]
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline
CP\_ID & CP\_NAME & SEC\_LEV & IP & CONTACT\_INFO & ADM \\
\hline
10 & The Swedish museum of art & 4 & 130.1.1.34 & petjo@swenus.se & Jan Jonsson \\
\hline
\end{tabular}
\caption{CP}
\end{table}

\textbf{3.8 Software System Attributes}

This section declares general requirements for the software that has not already been described.

\textbf{3.8.1 Security}

For each component in ARTSY (RC, CP, TS) there is one administrator that is responsible for the availability, reliability and security of the component. To maintain the overall security of ARTSY, only administrators are allowed to add, remove or update information. The identity of an administrator is authenticated using some authentication technique, for example login with user name and password.

\textbf{Conditional requirements}

\textbf{C-8.1.1} Only an administrator shall be able to add, remove or update information stored at an ARTSY component.
C-8.1.2 An administrator shall not be considered an administrator until the identity of the administrator has been authenticated.

C-8.1.3 The identity of an administrator shall be authenticated using a well-known authentication technique that has been approved by authorities in computer security.

3.8.2 Portability
ARTSY is a multi-platform system. Either the system can be easily ported to several computer platforms or the system is made platform-independent from the start.

Conditional requirements

C-8.2.1 (p) ARTSY shall not be designed to be used on one computer platform only.

A Use-cases

The use-cases are written with a specific implementation of ARTSY in mind. The Content Providers are museums with scanned photos of art as content. The Reproduction Clients are print workshops or museum shops with a printer. MuseDomina acts as both the Trusted Party and the TS.

A.1 Definition
A use-case is a scenario that describes how software can be used in a given situation. Use-cases are defined from an actors point-of-view.

A.2 Actors
An actor is a role that people (users) or devices play as they interact with the software. An actor is something that communicates with the system and that is external to the system itself. The following actors exist in ARTSY.

Reproducer: A person that uses an RC to order content.

Provider: A person that manages content at a CP.

Transaction manager: A person that is responsible for and manages the TS.
A.3 Use-cases

A.3.1 Request Information about Content

1 The reproducer has found an Image that she wants to know more about. She pushes the Information Request button and the Information Request window is opened. She inputs the id of the Image and the id of the CP. She pushes the button to send the Information Request. If the ids are invalid the system will ask the reproducer to enter valid ids. If a connection to CP is not possible (due to a network problem or a faulty CP), the RC will tell the reproducer to try later.

2 The requested information is shown on the screen, this information is extracted from the Content Meta Data file that the CP provides for each content. The reproducer can see a description of the image and available formats.

A.3.2 Invite Tender

1 From the Content Meta Data it is possible to invite a Tender. The reproducer pushes the invite tender button in the main window which opens the *Invite tender — choose image* window. She chooses a CP and an image that she wants to know more about.

2 The reproducer pushes the customize button. This will open the *Invite tender — customize image* window. The reproducer sees a thumbnail of the chosen image and gets to choose format and other specifications on the image (for example: size, resolution, color depth). When she has customized the tender invitation she pushes the Invite Tender button. If a connection to CP is not possible (due to a network problem or a faulty CP), the RC will tell the reproducer to try later.

3 When the Tender has arrived to the RC a message pops up on the main screen. The message tells the reproducer that a Tender from the CP has arrived. If this does not happen within one minute the system will ask the reproducer if the Tender Invitation shall be sent again.

4 The reproducer pushes the Tender button on the main menu and the received Tender is shown. The Tender contains information about price, format, how long the Tender is valid and other things. From here the reproducer can go on and make an Order by pushing the Order button.

---

14Invalid input format: The RC discovers that the input format is erroneous. Invalid CP-id: The RC discovers that there is no CP with the inputted id, by trying to contact the CP. Invalid image number: The CP tells the RC that the image does not exist.
A.3.3 Order Content

1 The reproducer pushes the Order button in the main window, the Order image window is shown. She chooses a CP id and a Tender from a list. When a choice has been made the Tender information is shown. A tender that is not valid will be highlighted. Tenders that have been invalid for a long time and never been attached to an Order will be deleted from the RC.

2 The reproducer finds her Tender and it is still valid. She pushes the Order button and a new window pops up and asks her to confirm the Order. She pushes the OK button and the Order is sent to the TS. If a connection to TS is not possible, the reproducer will be told to try to send the Order later, no registration of the Order will be made. If the TS fails to contact the CP the reproducer will be asked to try later.

3 The RC starts receiving the Image Representation within one minute from the time the Order was sent. If the transmission of the Image Representation fails the CP will re-send the Image Representation without an extra fee. If the RC already has started to receive an Image Representation the new Image Representation will be received in parallel. The progress of receiving Image Representations will be displayed to the user.

4 The Image Representation is received and the printer starts printing the Image Representation. If the file is corrupt, the user will be told that the CP will re-send the Image Representation without an extra fee. If another Image Representation is being printed the new Image Representation will be put in a printing queue. This queue is displayed to the reproducer at the RC. If the printer runs out of ink or paper the Image Representation will be reprinted without an extra fee.

A.3.4 Investigate Corrupt Transaction

1 The transaction manager is alerted by the main window at the TS that a corrupt transaction has occurred. She pushes the transaction button which opens the Transactions window. This will by default display the corrupt transaction and all information about it.

2 This information shows that an RC has reported a transmission error but a CP has not and it is the third time in one month this happens, with the same RC involved. The first two times the transaction was repeated without any extra cost or investigation.

3 The transaction manager contacts the RC and the CP to investigate what went wrong with the transaction.
A.3.5 Add Content

1 The provider pushes the content button and the Content window is shown.

2 The provider pushes the add content button and the Add Content window is shown.

3 The provider fills out the form for adding an Image: Name of the Image, artist, available Image Representations, technique, size, resolution, color depth and filenames.

4 The provider pushes the add button and the Image is added to the system. If any field in the submitted form is invalid (for example the size field containing letters), the CP will ask the user to correct the invalid field.

A.3.6 Add CP

1 The transaction manager pushes the add CP button in the main window of TS and the Add CP window is shown.

2 The transaction manager fills out a form with the following data: Name of CP, ip-address, preferred security level and accepted security level.

3 The transaction manager pushes the button add CP and the CP is added to the system.

A.3.7 Revoke Public Key

If the private key has been compromised the public key must be revoked. The Trusted Party shall be contacted to add a new public key to the public database. This function can be performed by the reproducer, the provider or the transaction manager.

1 The reproduction unit has recently dismissed an employee for being unloyal to the company. The unit is worried that the employee has a copy of the private key and may use it to make false Orders in the name of the reproduction unit. The reproducer pushes the revoke public key button on the main menu of RC.

2 The reproducer is asked to confirm that the public key shall be revoked. The reproducer confirms. A new private/public key pair is generated at the Component. The public key is included in the revocation request and so is the secret code that was received on the software installation. The revocation request is encrypted with the public key of the Trusted Party and sent to the Trusted Party.
3 The reproducer gets a message from the Trusted Party that a new public key is stored at the Public Key Registry. This message is received within two days and no transactions or Tender Invitations can be made during this time.
B  Messages sent in ARTSY

<table>
<thead>
<tr>
<th>Message type</th>
<th>Sent from</th>
<th>Sent to</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information Request</td>
<td>RC</td>
<td>CP</td>
<td>To get information about Content</td>
</tr>
<tr>
<td>Information Response</td>
<td>CP</td>
<td>RC</td>
<td>To provide information about Content</td>
</tr>
<tr>
<td>Tender Invitation</td>
<td>RC</td>
<td>CP</td>
<td>To get a Tender on a Content Representation</td>
</tr>
<tr>
<td>pre-Tender</td>
<td>CP</td>
<td>TS</td>
<td>The first step in creating a Tender for an RC</td>
</tr>
<tr>
<td>Tender</td>
<td>TS</td>
<td>RC</td>
<td>The Tender requested in the Tender Invitation</td>
</tr>
<tr>
<td>Order</td>
<td>RC</td>
<td>TS</td>
<td>To start a transaction on the Tender with the attached Tender id</td>
</tr>
<tr>
<td>Transaction Start</td>
<td>TS</td>
<td>CP &amp; RC</td>
<td>Starts a transaction. Includes symmetric keys</td>
</tr>
<tr>
<td>Transaction Result</td>
<td>TS</td>
<td>CP &amp; RC</td>
<td>To inform a client of the result of a transaction</td>
</tr>
<tr>
<td>Error</td>
<td>any</td>
<td>any</td>
<td>To inform a component that something went wrong</td>
</tr>
<tr>
<td>Revocation Request</td>
<td>any</td>
<td>Trusted Party</td>
<td>To revoke a public key</td>
</tr>
</tbody>
</table>

Table 13: Messages sent in ARTSY
APPENDIX B
ARTSY
Prototype Documentation
## Contents

1 Introduction  
2 Start-up Instructions  
   2.1 Downloading Files  
   2.2 Setting the Java Classpath  
   2.3 Setting up the Database  
      2.3.1 Create a Database for Each Component  
      2.3.2 Create Tables for Each Database  
      2.3.3 Put Images in the CP Database  
   2.4 Preparing the Files  
   2.5 Compiling  
   2.6 Executing  
   2.7 Generating documentation  
   2.8 Additional Information  
3 Using ARTSY  
   3.1 Information Request  
   3.2 Tender Submission  
   3.3 Transaction  
   3.4 Additional Information  
4 Code Structure  
   4.1 Common Classes  
   4.2 Component-Specific Classes  
5 Flowcharts of the code  
   5.1 Information Request  
   5.2 Tender Submission  
   5.3 Transaction Part 1  
   5.4 Transaction Part 2  
   5.5 Transaction part 3
CONTENTS
1 Introduction

The purpose of this document is to give an introduction to the ARTSY prototype and to make it possible to set up and configure the ARTSY prototype environment. The document also contains a short guide on how to use the ARTSY prototype. It does not contain a complete documentation of the source code but it should be sufficiently detailed to make it possible to understand the main structure of the code and how the different components of ARTSY interact with each other. A full documentation of the source code can be generated using JavaDoc (See the section 2.7).

Section 2 contains instructions on how to set up, configure and compile the ARTSY environment. Once these instructions have been followed correctly it should be possible to start ARTSY. Section 3 contains a concise guide, with screenshots, on how to use ARTSY once it has been started. Section 4 contains information about how the code is organized into different packages and it also explains the purposes of the most important classes. Section 5 illustrates, using flowcharts, how different classes interact with each other.

The ARTSY prototype meets the prototype requirements that are specified in the ARTSY Software Requirements Specification document (SRS). The SRS contains detailed information about the inner workings of ARTSY and it can be used as a complement to this document.

2 Start-up Instructions

ARTSY is a multi-platform system. Unfortunately, command line instructions are written differently on different platforms. The path separator used in Unix “/” is, for example different from the path separator used on the Win32 platform “\”. In this section the Unix way of writing things have been used as the default in all examples except those that apply only to the Win32 platform. Other platforms, such as the Macintosh, are not described in this document. It should however be easy to adopt these instructions to other platforms.

2.1 Downloading Files

Some preparations need to be done before ARTSY can be started. A database manager must be installed and the ARTSY packages must be stored on the hard drive. In this section it is assumed that an empty prototype/ directory has been created. The name of this directory is irrelevant, any name will do. Download the following files:

Java

The Java 2 Platform, Standard Edition 1.4.1 was used to create ARTSY. It can be downloaded from http://java.sun.com. Install Java anywhere on the harddrive. The JavaDoc tool (see section 2.7) is included in the Java package.
ARTSY
Download artsy.zip from http://www.isy.liu.se/cvl/ScOut/Masters/PaperInfo/artsy2003.html using a web browser. Unzip the file and copy all files that are produced into the prototype/ directory on the harddrive. It is possible to set up ARTSY so that different components run on different computers. The entire artsy/ directory (one of the directories included in the zip file) can be copied to all computers that will be used but it is better to just copy the files that are actually needed. On all computers where a CP component will be located the artsy/cp/ directory must be copied. The file path must not be changed; there has to be an artsy/ directory and the cp/ directory must be located in the artsy/ directory. TS components correspondingly needs the artsy/ts directory. RCo and RCi components need be grouped together. It is possible to use several RCo and RCi components but they must be grouped in pairs. Copy both the artsy/rco and the artsy/rci/ directory to all computers where RCo/RCi components will be located.

miniSQL
The DBMS miniSQL can be found at http://www.hughes.com.au/download/index.shtml. A free registration is required before the download can begin. A license costs $250 but the DBMS can be used for free a limited period of time. In ARTSY version 3.0 is used, but a later version might work. Install the DBMS when it has been downloaded.

miniSQL jdbc driver
The miniSQL jdbc driver msqjdbc-2-0b5.jar can be found in a directory named thirdPartyPackages/ (included in the artsy.zip file). It enables communication between Java applications and miniSQL. Copy the whole directory to the prototype root. At the time of writing it is very difficult to find this driver on the web. It should be located at http://www.imaginary.com/Java but has not been so for at least four months.

JDOM
Since the thirdPartyPackages/ directory already is in the prototype root there is no need to copy the JDOM package once more. jdom.jar enables ARTSY to interpret and create XML-documents. JDOM Beta 8 is used in ARTSY and the package can be found at http://www.jdom.org

Bouncy Castle
The Bouncy Castle package, bcpv-jdk14-115.jar is also located in the thirdPartyPackages/ directory. It provides ARTSY with some encryption algorithms. The package can be found at http://www.bouncycastle.org. The use of the package is not restricted in any way. It is free to use in a commercial product.
2.2 Setting the Java Classpath

In section 2.1 the thirdPartyPackages/ directory was copied to the harddrive into the prototype/ directory. When the prototype is run the Java interpreter must know where to locate these third party packages. This is done by setting the classpath. This is done differently on different operating systems. In a Unix or a Linux environment this is done by executing the following command in a shell:

```bash
setenv CLASSPATH
.: [DIR]/prototype/thirdPartyPackages/msql-jdbc-2-0b6.jar
; [DIR]/prototype/thirdPartyPackages/jdom.jar
; [DIR]/prototype/thirdPartyPackages/bcprov-jdk14-115.jar
```

Where [DIR] is any directory of choice, for example $HOME. In a Windows environment this can be done by executing the following command at the command prompt:

```bash
set CLASSPATH=
.; [DIR]\prototype\thirdPartyPackages\msql-jdbc-2-0b6.jar
; [DIR]\prototype\thirdPartyPackages\jdom.jar
; [DIR]\prototype\thirdPartyPackages\bcprov-jdk14-115.jar
```

Where [DIR] is any directory of choice, for example C:\.

2.3 Setting up the Database

2.3.1 Create a Database for Each Component

Each component needs its own database (except for the RCi). A database is created by executing the command:

```bash
% msql3/bin/msqladmin create [DATABASENAME]
```

Where [DATABASENAME] is the name of the database that shall be created. When setting up a small ARTSY consisting of two RCs one TS and two CPs following commands shall be executed:

```bash
% msql3/bin/msqladmin create RC1
% msql3/bin/msqladmin create RC2
% msql3/bin/msqladmin create CP1
% msql3/bin/msqladmin create CP2
% msql3/bin/msqladmin create TS
```

There can be only one TS in the system and each RC and CP has its own id consisting of the component type and a number.
2.3.2 Create Tables for Each Database

Each database needs a number of tables. Open a database with the following command:

```
% mysql/bin/mysql <name of the database>
```

Each type of component (RC, TS, CP) has a file containing miniSQL instructions for making the necessary tables. These files are called tablesRC.txt, tablesTS.txt and tablesCP.txt and are located in the database/ directory (included in the artsy.zip file).

Open each created database and copy all text from the corresponding instruction file to the mysql program. All tables necessary for the database are now created.

2.3.3 Put Images in the CP Database

When the artsy/ directory was copied, two images and their corresponding Content Meta Data files were copied also (artsy/cp/cmd01.xml and artsy/cp/cmd02.xml). These Content Meta Data files must be added to the database. The following commands adds the two files in the prototype/artsy/cp/ directory to the CP1 database:

```
% javac AddCMD.java
% java AddCMD CP1 cmd01.xml
% java AddCMD CP1 cmd02.xml
```

Now two Content Meta Data files have been added to the CPDatabase. Check that the files were added by opening the database and running the following command:

```
mysql> select * from CMD \g
```

2.4 Preparing the Files

There are two files included in the artsy/ directory that need to be edited manually in order to adapt ARTSY to its environment:

```
artsy/cpMappings.txt
artsy/database.txt
```

`cpMappings.txt` is only needed on the computer where the RCs will be executed. `database.txt` must be available on all computers that are used. By default `cpMappings.txt` contains the following text:

```
cp1 bullock.isy.liu.se
cp2 bancroft.isy.liu.se
```
2.5 Compiling

The file is used by the RCo component to locate a CP given an id. According to the default cpMappings.txt the CP with id cp1 is located at the host bullock.isy.liu.se, and so forth. The information in cpMappings.txt needs to be updated so that it is applicable for its current environment. Replace bullock.isy.liu.se with the host name of the computer where the CP will be located. If a second CP will be used bancraft.isy.liu.se should be replaced as well. If more than two CPs shall be used, more lines must be added to the file. The CPs does not have to be named cp1, cp2, ... cpm. It is just a suggestion. Any name can be used for a CP but it must not include space characters. The database.txt file contains the following text by default:

bullock.isy.liu.se

It is used by all components to locate the database. Consequently this file must be available to all components. Update this file by replacing bullock.isy.liu.se with the name of the host where the database is located.

2.5 Compiling

Before ARTSY can be started the source files must be compiled into class files. If all ARTSY components are to be executed on one computer, or if different components are to be executed on different computers does not matter. The procedure that follows will work in either case. All instructions below must, however, be entered once on every computer that shall be used.

The files can be compiled one at a time, but it is easier to compile all of the files at the same time. Make sure that the current directory is the same directory to which the ARTSY directory was copied, then compile the source files by giving the following instruction:

```
% javac artsy/*/.*.java
```

All source files now have corresponding class files. ARTSY utilizes the Java Remote Method Invocation technology (Java RMI) to exchange information over the network. In order to get the Java RMI parts of ARTSY to work properly, some additional compiling must be done now. From the same directory as above, give the following instruction:

```
% rmic artsy.common.Network
```

The second instruction will result in an error if the first instruction never was entered. Now enter this instruction:

```
% rmic artsy.common.RemoteStream
```

All files that are needed to start ARTSY have now been created.
2.6 Executing

To launch ARTSY four different processes need to be started: The RCo, the RCi, the TS and the CP. All of these processes can run on one host computer or several computers could be used. However, an RCi component needs to be located on the same computer as an RCo component. Make sure that the current directory is the same directory to which the ARTSY directory was copied, then start the RCo component with the following instruction:

```bash
% java -Djava.security.policy=artsy/rco/.java.policy
    -Djava.rmi.server.codebase=file:[DIR]/prototype/
    artsy.rco.RCo [RCo-ID]
```

Where [RCo-ID] is the id that the RCo shall use. By default the ids “rc1” and “rc2” can be used, but no other ids. This is because ARTSY comes with two pre-generated pair of keys. The key pairs of “rc1” and “rc2”. Examine the current directory. All key file names start with the id of a component. This convention must be obeyed if new key pairs are generated. Start the RCi with the following instruction:

```bash
% java -Djava.security.policy=artsy/rci/.java.policy
    -Djava.rmi.server.codebase=file:[DIR]/prototype/
    artsy.rci.RCi [RCi-ID]
```

The RCi ids that can be used by default is “rc1i” and “rc2i”. All RCIs are connected to an RCo. An RCi must to be located on the same computer as an RCo and it must have an id on the form: [RCo-ID]i where [RCo-ID] is the id of the RCo that the RCi is connected to (the RCo that is located on the same computer as the RCi). Start the TS with the following instruction:

```bash
% java -Djava.security.policy=artsy/ts/.java.policy
    -Djava.rmi.server.codebase=file:[DIR]/prototype/
    artsy.ts.TS [TS-ID]
```

The only TS id that can be used by default in the prototype is “ts”. Start the CP with the following instruction:

```bash
% java -Djava.security.policy=artsy/cp/.java.policy
    -Djava.rmi.server.codebase=file:[DIR]/prototype/
    artsy.cp.CP [CP-ID] [TS-ID] [TS-HOST]
```

All CPs are connected to a TS. The last two arguments are the id and the location of TS that this particular CP shall be connected to, for example ts and bancroft.isy.liu.se. The TS that a CP is connected to does not have to be located on the same computer as the CP.

2.7 Generating documentation

Detailed information about what every class in ARTSY does can be generated using the JavaDoc tool (distributed along with the Java package, see section ??). To generate documentation with JavaDoc, enter the following in a shell window:
javadoc -d [TARGET-DIR] [SOURCE-DIR]/prototype/artsy/*/.*.java

Where [TARGET-DIR] is the directory where the documentation will be located and [SOURCE-DIR] is the directory on the level above the prototype/ directory. After the instruction above has been given, documentation will be generated and placed in [TARGET-DIR]. The documentation is in html format, a standard web browser can be used to browse it.

2.8 Additional Information

The KeyGen application can be used to create new encryption/decryption key pairs. Key pairs for the following components are already generated and included in the artsy.zip file: rcl, rcli, rc2i, cp1, cp2, ts. If other ids will be used or if more components are added, new key pairs must be created. When KeyGen is executed it creates the following files:

private.key
public.key

The files that are generated must be renamed so that they match the id of their owner. If for example a new CP with id cp3 shall be used, the files shall be renamed to cp3_public.key and cp3_private.key. The KeyGen application is included in the artsy.zip file in the extras/ directory.

3 Using ARTSY

When the RCo component has been launched from the command prompt (see section 2.6) the window in figure 1 will show up after a short while. Before using any of the buttons in the window make sure that the rest of the components have been started and that they are ready to receive messages. The Content Provider window in figure 3 will show up after the CP has been launched. The CP is ready to receive message when the text “Ready to receive messages” is displayed in the window, like it is in the figure. When the TS has been launched the Transaction Server window in figure 3 will show up. The RCI component does not have a graphical user interface since it is intended that it will be inserted into a printer. The RCI component can be used when “Ready to receive messages” is displayed in the command prompt window from which the RCI was started.

Below instructions are given on how to perform all of the main features of ARTSY. It is assumed that ARTSY has been configured properly before it was launched (see section 2.4).

---

1An alternative solution is to simply rename some of the pre-generated keys
Figure 1: Reproduction Client

Figure 2: Content Provider and Transaction Server
3.1 Information Request

To request information from a CP (using the RCo interface) do the following:

1. Enter a valid image id in the Image Id box in the upper left-hand corner. An image id is valid if an image with that id is available at CP from which the information shall be requested.

2. Enter a valid cp id in the CP Id box next to the Image Id box. All cp ids that are valid are specified in the cpMappings.txt file (located in the artsy/RCo/ directory).

3. Push the Make Request button next to the CP Id box.

After a short while a Content Meta Data file will be displayed in the log window to the right. An example is showed in figure 3.

3.2 Tender Submission

To invite a Tender from a CP (using the RCo interface) do the following:

1. Enter a valid image representation id in the Image Rep Id box below the Image Id box. A valid Image Representation id can be found in the Content Meta Data that was received as a result of the Information Request: Locate an IMAGE-REPRESENTATION: heading (there are one or more of those) and use the id specified on the next line.
2. Enter a valid cp id in the CP Id box next to the Image Rep Id box. If an Image Representation id from the Content Meta Data was used, then the id of the CP that sent the Content Meta Data must be used.

3. Push the Invite Tender button next to the CP Id box.

After a short while a Tender will be displayed in the log window to the right.

### 3.3 Transaction

![Reproduction Center](image)

**Figure 4:** An image has been printed

To invite a Tender from a CP (using the RCo) interface do the following:

1. Enter a valid Tender id in the Tender Id box in the lower left-hand corner. A valid Tender id can be found in the Tender that was received as a result of the Tender submission: Locate the TENDER-ID: heading (there is only one such heading) and use the id specified on the same line.

2. Push the Make Order button next to the Tender Id box.

After a short while information about the reproduction process will be displayed in the log window to the right. The most recent information is displayed at the top. An example is showed in figure 4.

### 3.4 Additional Information

In the upper part of the RCo window there are three tabs. The tabs are labeled Main menu, Image Meta Data Files and Tenders. Until now only the Main
menu tab has been used. The other two tabs can be used to browse through Content Meta Data files (Image Meta Data files) and Tenders (Tenders) that have been received.

Since it is possible to invite a Tender and then wait a few days before making an Order that is based on that Tender, it is convenient to have all Tenders easily available. It is also convenient to have a quick way of accessing old Content Meta Data files. More than one Tender can be invited using one Content Meta Data file. This is why those two tabs have been added. Figure 5 shows an example of the RCo interface where the Tenders tab has been chosen.

4 Code Structure

The Java source code is divided into five packages:

- artsy reco
- artsy rci
- artsy cp
- artsy ts
- artsy common
ARTSY is based on four different types of components: The RCo, the RGi, the CP and the TS. The first four packages contain component-specific classes for the four component types. *artsy reco* contains classes needed by the RCo component and so forth. The last package, *artsy common*, contains common classes (shared classes) that are needed by all components. Some of the classes in the *artsy common* package are superclasses with corresponding subclasses in the other packages. Other classes in *artsy common* are data classes that must be known to all components since they are passed between the components in order to exchange information. Below is a concise description of the most important classes.

### 4.1 Common Classes

The most important classes contained in the *artsy common* are:

**Message**

This data class holds all the data that is needed in an ARTSY message. Instances of this class are the only objects that are passed over the network. There are no subclasses of this class.

**MessageHandler**

This superclass contains generic methods for message handling. All components have their own MessageHandler, a subclass of this class. The MessageHandler handles incoming as well as outgoing messages. Incoming messages are received from the Network class and interpreted. Depending on the message type different methods are then called. The contents of an incoming message are often passed over to the UserInterface class for viewing. Outgoing messages are created by MessageHandler and then passed on to the Network class.

**Network**

This class handles all network accesses. Each component create one instance of this class. There are no subclasses. All outgoing messages are transmitted from the local instance of this class to a remote one. Incoming messages are correspondingly received from a remote Network instance and handled by the local Network instance.

**Security**

This superclass includes generic methods for encryption and decryption. All components have their own Security class, a subclass of this class. All incoming messages are processed by the local Security subclass before they are passed to the local MessageHandler. All outgoing messages are processed by the local Security class before they are transmitted over the network by the Network class. The main tasks of the Security classes are to decrypt incoming messages and to encrypt outgoing messages.

**User Interface**

This superclass includes generic methods for displaying information and
for receiving instructions from the user. All components have their own
UserInterface class, a subclass of this class. The user interface is totally
separated from the rest of the code so that minimal effort is needed to
replace it. Depending on how this class is implemented the user interface
can be either graphical or textual. The current implementation provides
a rather simple graphical user interface.

4.2 Component-Specific Classes

The structures of the classes in the component-specific packages (artsy.reo,
artsy.rcl, artsy.cp and artsy.ts) are quite similar. All components have own
subclasses of the MessageHandler, Security and UserInterface superclasses de-
scribed in section 4.1. All subclasses have names on the following form:

[SUPERCLASS-NAME][SUFFIX]

where [SUFFIX] is the name of the local component, for example “CP”. Besides
these subclasses all components also have a main class. The main classes have
the same name as the component to which they belong. The main class of a
component is the class that is launched from the command prompt. It creates
all other fixed classes\(^2\) that the component will use and then puts itself into an
inactive state.

In addition to the classes described above every package also contain some
extra classes that are not needed in all packages. Among the extra classes are
for example classes for handling database accesses, for creating specific XML
documents and for printing documents. The documentation that can be gen-
erated using Javadoc (see section 2.7) contains detailed descriptions of every
class in ARTSY.

5 Flowcharts of the code

This section illustrates the main features of ARTSY using flowcharts. The
flowcharts only include objects of such classes that are unique to ARTSY (ex-
cept for the CipherInputStream class in figure 10). The symbols used in the
flowcharts are explained below.

- **Boxes** Boxes represent instances of classes (objects). There are only one
  object of each box type at each ARTSY Component. These objects are
  the main building blocks in the prototype.

- **Ellipses** Ellipses represent instances of classes (objects). There can be
  more than one object of each ellipse type at each ARTSY Component.

- **Numbers** The flow starts at number one and then continues with number
two and so forth.

---

\(^2\)Non-data classes which are created once and not terminated until the component is shut
down.
• **One way arrows** One way arrows indicate that the control is transferred from the object where the arrow starts to the object where the arrow ends.

• **Two way arrows** Two way arrows indicate that the object where the arrow was encountered invokes a method on the object on the other side of the arrow. The control is returned to the invoker when the second object has finished processing the invocation.

### 5.1 Information Request

![Information Request Diagram]

Figure 6: Information Request

A user starts an Information Request via the UserInterfaceRC class (see figure 6). Arrow number 4 shows that the SecurityRCo class is called, even though the Information Request is not encrypted. SecurityRCo keeps track of the messages that shall be encrypted and those that shall not. All messages that are sent from the RCo are processed by the SecurityRCo class. At arrow number 8 the user at the CP is notified that an Information Request has been received. A Content Meta Data (CMD) file is fetched from the database and sent to the RCo. The CMD is displayed at arrow 16 after it has been parsed and formatted by the XMLParser class.

### 5.2 Tender Submission

A user at the RC starts a Tender Submission via the UserInterfaceRC class (see figure 7). At arrow 3 the TenderInvitationFactory class is called. This class creates a Tender Invitation document and stores it in the database (arrow 4). The Tender Invitation is put in a message at arrow 5 and sent to the Network class. At arrow 7 the Tender Invitation is encrypted and signed. When the Tender Invitation has been received at the CP it is decrypted (arrow 9),
5.3 Transaction Part 1

parsed (arrow 13) and used for creating a pre-Tender document. The pre-
Tender document is stored in the database (arrow 14) and sent to the TS after
it has been encrypted (arrow 17). The TS receives the pre-Tender, parses it
(arrow 23) and uses it to create a Tender document. The Tender document is
then stored in the database (arrow 25). The Tender is sent to the RC (arrow
29) and parsed (arrow 33) before it is displayed (arrow 34).

5.3 Transaction Part 1

A user at the RC starts a Transaction by entering a Tender id (see figure 8). The
Tender id is sent to the MessageHandlerRCo which acquires the corresponding
Tender from the database (arrow 3) and parses it (arrow 4) to check that the
Tender is valid. If it is, an Order is sent to the TS (arrow 7). The TS also
checks that the Tender is valid by acquiring it from its database (arrow 11) and
parsed it (arrow 12). A symmetric session key is generated by the KeyFactory
class (arrow 13). Transaction Start messages are sent to the RC and the CP
(arrow A). The Transaction Start messages are decrypted at each Component
(arrow B) and sent to the MessageHandler classes (arrow C).
5.4 Transaction Part 2

In Transaction Part 1 the Transaction Start messages were received by the RCo and the CP. The RCo received two Transaction Start messages in one. One Transaction Start message that was intended for the RCi. The RCi Transaction Start message was included in the RC Transaction Start message and was encrypted with the public key of the RCi. The session key was included in the RCi Transaction Start message (unreadable to the RCo). This is where Transaction Part 2 starts. The RCo forwards the attached Transaction Start message to the RCi in figure 9 (arrow E). The RCi decrypts it (arrow F) and stores the received session key.

The next thing that happens is that the CP attaches the Content Representation to a RemoteStream (arrow 1, 2 and 3) and registers the RemoteStream at the RMIRegistry. The Content Representation is now available for download and the CP sends a Ready To Transmit message including the name of the RemoteStream (arrow 6). The Ready To Transmit message is received at the RCo and a LocalStream is created and bound to the RemoteStream at the CP.
5.5 Transaction part 3

In all the other parts messages were the only kind of data that was sent between Components. In Transaction part 3 no messages are sent and no new objects are created. Each LocalStream is connected to a RemoteStream at another Component. The Transaction part 3 starts at the Basic Print object (see figure 10). The Basic Print objects calls the CipherInputStream (arrow 1). The CipherInputStream calls the LocalStream (arrow 2) which calls the RemoteStream at the RCo (arrow 4). This goes on until arrow 7, where the FileInputStream is
reached at the CP. The FileInputStream reads some bytes from the Content Representation (arrow 8) that is stored as a file and returns them to the calling CipherInputStream (arrow 9). The CipherInputStream at the CP encrypts the received bytes with the session key received in Transaction part 1 and returns the encrypted bytes to the calling RemoteStream (arrow 10). The encrypted bytes are returned all the way to the CipherInputStream at the RCI (arrow 11 to 14) where the bytes are decrypted using the received session key in Transaction part 2. The bytes are passed to the BasicPrint which sends them to the printer (arrow 16). This procedure is repeated until the entire Content Representation has been read and sent to the printer.

Figure 10: Transaction part 3: An image is printed
På svenska

Detta dokument hålls tillgängligt på Internet – eller dess framtidiga ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga ansehen eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

In English

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© Sofia Max, Mårten Björk