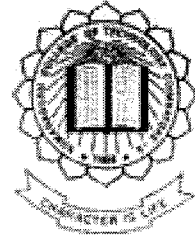
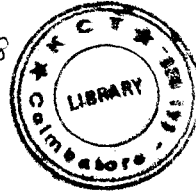




P-2298



SECURED THIRD PARTY DISTRIBUTION

By

P.SRI RAM PRABHU
Registration Number: 71205621050

Of

KUMARAGURU COLLEGE OF TECHNOLOGY
COIMBATORE

A PROJECT REPORT

Submitted to the

FACULTY OF INFORMATION AND COMMUNICATION ENGINEERING

In partial fulfillment of the requirements

for the award of the degree

of

MASTER OF COMPUTER APPLICATION

ANNA UNIVERSITY
CHENNAI 600 025

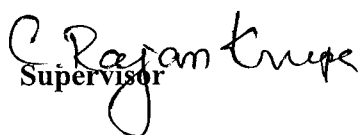
June 2008

KUMARAGURU COLLEGE OF TECHNOLOGY

Coimbatore-641006


DEPARTMENT OF COMPUTER APPLICATION**Bonafide Certificate**

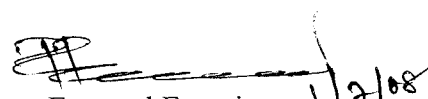
Certified that this project report titled **SECURED THIRD PARTY DISTRIBUTION** is the bonafide work of **Mr.P.Sri Ram Prabhu (Registration Number: 71205621050)** who carried out the research under my supervision. Certified further, that to the best of my Knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.


Supervisor


Head of the Department

Submitted to Project and Viva Examination held on 01-07-2008


Internal Examiner


External Examiner



YOUTH SOFT

30th May 2008

PROJECT COMPLETION CERTIFICATE

This is certify to that Mr.P.Sri Ram Prabhu (Reg.No: 71205621050) doing final year M.C.A in Kumaraguru College of Technology has completed his project entitled “ SECURED THIRD PARTY DISTRIBUTION” Under the guidance of Mr. B. Pradeep Kumar, Senior Software Engineer, in our concern during the period of December 2007 to May 2008.

He has successfully completed the project as per the requirements.

S.KANDHAN PRABHU
Manager, Human Resources

No: 26 G, North Crescent road Off GN Chetty Road T.Nagar Chennai – 600 017

Phone 91- 44 – 65262677

Email: support@youthsoft.net

Web : <http://www.youthsoft.net>

TABLE OF CONTENTS

| CHAPTER NO | TITLE | PAGE NO |
|------------|---|---------|
| | ABSTRACT | iii |
| | LIST OF TABLES | vii |
| | LIST OF FIGURES | vii |
| I | INTRODUCTION | |
| | 1.1 ORGANISATION PROFILE | 1 |
| | 1.2 ABSTRACT | 2 |
| | 1.3 PROBLEM DEFINITION | 3 |
| II | SYSTEM ANALYSIS | |
| | 2.1 EXISTING SYSTEM ARCHITECTURE | 4 |
| | 2.2 PROPOSED SYSTEM ARCHITECTURE | 5 |
| | 2.2.1 BASIC CONCEPTS OF PROPOSED SYSTEM | 5 |
| | 2.3 USER INTERFACE REQUIREMENTS | 8 |
| III | DEVELOPMENT ENVIRONMENT | |
| | 3.1 H/W ENVIRONMENT | 9 |
| | 3.2 S/W ENVIRONMENT | 9 |
| IV | SYSTEM DESIGN | |
| | 4.1 DATA MODEL | 10 |
| | 4.1.1 ER DIAGRAM | 10 |
| | 4.1.2 DATA DICTIONARY | 11 |
| | 4.1.2.1 SUBJECT TABLE | 11 |
| | 4.1.2.2 POLICY TABLE | 11 |
| | 4.2 PROCESS MODEL | 12 |
| | 4.2.1 USE CASE DIAGRAM | 12 |
| | 4.2.1.1 CLIENT USECASE | 12 |

| | |
|---|----|
| 4.2.1.2 PUBLISHER USECASE | 13 |
| 4.2.1.3 CLIENT USECASE | 14 |
| 4.2.2 DATA FLOW DIAGRAM(DFD) | 15 |
| 4.2.2.1 CREATE AND VALIDATE SUBJECTS | 15 |
| 4.2.2.2 EXECUTE QUERY BY SUBJECTS | 15 |
| 4.2.2.3 POLICY DETAILS | 16 |
| V ARCHITECTURAL DETAILS | |
| 5.1 PROGRAM DESIGN LANGUAGE | 17 |
| 5.1.1 VB.NET | 17 |
| 5.1.2 .NET FRAMEWORK | 17 |
| 5.1.3 SQL SERVER. | 19 |
| 5.1.4 BASIC CONCEPTS OF XML. | 19 |
| 5.1.5 ARCHITECTURE OF THIRD PARTY DISTRIBUTION. | 20 |
| VI TESTING | |
| 6.1 TESTING METHOLODIGIES | 26 |
| VII PERFORMANCE AND LIMITATIONS | |
| 7.1 IMPLEMENTATION AND MAINTENANCE. | 28 |
| 7.2 IMPLEMENTATION PROCEDURE. | 29 |
| 7. 2.1. SYSTEM MAINTENANCE | 30 |
| 7.3 FUTURE ENHANCEMENTS | 32 |
| VIII CONCLUSION | 33 |
| IX APPENDICES | |
| 8.1 SAMPLE SCREENS | 34 |
| X REFERENCES | 55 |

LIST OF TABLES

| <u>TABLE DESCRIPTION</u> | <u>PAGE NO</u> |
|---------------------------------|-----------------------|
| 4.1.2.1 SUBJECT TABLE | 11 |
| 4.1.2.2 POLICY TABLE | 11 |

LIST OF FIGURES

| <u>FIGURE DESCRIPTION</u> | <u>PAGE NO</u> |
|---------------------------------------|-----------------------|
| 4.1.1 ER DIAGRAM | 10 |
| 4.2.1.1 CLIENT USECASE | 12 |
| 4.2.1.2 PUBLISHER USECASE | 13 |
| 4.2.1.3 CLIENT USECASE | 14 |
| 4.2.2.1 CREATE AND VALIDATE SUBJECTS | 15 |
| 4.2.2.2 EXECUTE QUERY BY SUBJECTS | 15 |
| 4.2.2.3 POLICY DETAILS | 16 |
| 5.1.4.1 XML DESIGN | 20 |
| 5.1.5.1 SYSTEM ARCHITECTURE | 22 |
| 5.1.5.2 SUBJECT-PUBLISHER INTERACTION | 25 |

I. INTRODUCTION

1.1 ORGANISATION PROFILE

Youth Soft – a new generation software company that understands business and the bottom line. We have chosen a strategic global model combining the best of onshore software development to deliver premium quality services and products to our clients at affordable cost.

Youth Soft vision is to achieve global IT services leadership in providing value-added high quality IT solutions to our clients in selected horizontal and vertical segments, by combining technology skills, domain expertise, process focus and a commitment to long-term client relationships. At Youth Soft, we are focused on optimizing our customer's investments information technology. We help customers envision and shape their future around the key drivers of technology, productivity and cost-effectiveness.

The infrastructure includes policies and standards for

- Communication
- Disaster recovery
- Data (Security)
- Network Security
- Physical Security

1.2 ABSTRACT

XML (eXtensible Markup Language) is rapidly becoming a de facto standard for document representation and exchange over the Web. A common requirement for Web applications is thus the need for secure publishing of XML documents. By secure publishing, we mean that the publishing service must ensure some security properties to the data it manages. Third-party architectures for data publishing over the Web are today receiving growing attention, due to their scalability properties and to the ability of efficiently managing a large number of subjects and a great amount of data.

In third-party architecture, there is a distinction between the Owner and the Publisher of information. The Owner is the producer of the information, whereas Publishers are responsible for managing (a portion of) the Owner information and for answering subject queries. A relevant issue in this architecture is how the Owner can ensure a secure publishing of its data, even if the data are managed by a third-party.

Modules in the project,

XML Parser Performs basic xml file reading and writing operations. Policy Configuration- When a subject subscribes to the Owner, it receives back an object called subject policy configuration. Subject Publisher Interaction- For each document the Publisher is entitled to manage, the Owner sends the corresponding security enhanced (SE-xml) document and the corresponding secure structure. Reply Generation- When a subject submits a query to a Publisher, the Publisher first determines the set of nodes that need to be returned to him/her. Subject Verification- A subject upon receiving a reply document and a secure structure can verify the authenticity and the completeness of the corresponding query answer.

The Internet is an open architecture susceptible to various forms of network attacks. A common requirement for web applications is thus the need for secure publishing of data.

1.3 PROBLEM DEFINITION

XML (eXtensible Markup Language) is rapidly becoming a de facto standard for document representation and exchange over the Web. It allows information and services to be encoded with meaningful structure and semantics that computers and humans can understand. It is capable of describing many different kinds of data, and can easily be extended to include user-specified and industry-specified tags.

In third-party architecture, there is a distinction between the Owner and the Publisher of information. The Owner is the producer of the information, whereas Publishers (third-party) are responsible for managing the Owner information and for answering subject (client) queries. A relevant issue in this architecture is how the Owner can ensure a secure publishing of his/her data, even if the data are managed by a third-party. To implement the proposed approach that does not require the publisher to be trusted, mainly focusing most relevant security properties: Authenticity and completeness.

Scope and Need

- XML data encoding is used for data representation in wide area of applications.
- Publishers need not to be trusted with respect to authenticity and completeness properties. At the same time, we can ensure that a subject is able to verify such properties on the answer returned by a Publisher.

II. SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

In existing system used the cryptographic algorithm for this encryption and decryption using one public key and two private key. And also the public key and the private key are same for source and destination so hacking and find the public key is enough to find the private key for third parties. This procedure leads to less security of XML documents while publishing through third parties from the owner.

Existing Methods

The most intuitive solution is that of requiring publishers to be trusted with regard to the considered security properties.

Drawbacks of Existing Methods

- Publishers may publicly release or sell the confidential information of the firm.
- We cannot ensure that data received by the subjects are un-modified and original.
- Large Web-based systems cannot be easily verified to be secure and can be easily penetrated.

2.2 PROPOSED SYSTEM

In proposed system using an algorithm called DES(Data Encryption Standard) and RSA(Ron Rivest, Adi Shamir and Len Adleman)which provides two public and private keys where source and destination can used for encrypt and decrypt the XML documents. This provides grater security than other existing systems. Third party of publisher server in difficult of hacking keys and get original Xml documents.

Proposed Method

- RSA
- DES
- Access Control Policies
- Policy Configuration
- Digital Signature and Hashing

Advantages of Proposed System

- Publishers need not to be trusted, with respect to authenticity and completeness.
- Subjects can access the documents, according to access control policies only.
- Our system can be applied to any decentralized architecture and our solution offers the advantage of being scalable and of reducing the risk that the Owner becomes the bottleneck of the entire system.

2.2.1 BASIC CONCEPTS OF PROPOSED SYSTEM

Ensuring document authenticity means that the subject receiving a document is assured that its contents are actually original from the Owner itself. Ensuring the completeness of the response means that any subject must be able to verify that he or she has received all or portions of the documents that is he/she requested to access, according to the stated access control policies.

Owner-Publisher Interaction

Owner sends the Publisher, in addition to the documents it is entitled to manage, a summary signature for each managed document, generated using a technique based on Merkle hash trees.

Subject-Publisher Interaction

When a subject submits a query to a Publisher, the Publisher also sends him/her, besides the query result, also the signatures of the documents on which the query is performed. In this way, the subject can locally recomputed the same bottom-up hash value signed by the Owner and, by comparing the two values, he/she can verify whether the Publisher has altered the content of the query answer and can be sure of its authenticity.

Access Control Policy

Each subject has some access control policy like some documents or portions of some documents are accessible only to some of the subjects. This is called selective property. Query answers returned by the publisher to a subject are filtered according to the access control policies specified by the owner. Thus, the receiving subject is prevented from accessing information he/she is not allowed to access, being at the same time able to perform the completeness verification.

Owner-Subject Interaction

To make the publisher able to verify which access control policies apply to a subject, the owner returns the subject a policy configuration, that is, a certificate containing information about the access control policies that apply to the subject. The subject policy configuration is signed with the private key of the owner to prevent the

subject from altering its content. Thus, all the information exchanged between the parties of our architecture is completely secure.

- **Implementation of RSA Algorithm**

The RSA algorithm is named after Ron Rivest, Adi Shamir and Len Adleman, The RSA algorithm can be used for both public key encryption and digital signatures. Its security is based on the difficulty of factoring large integers.

1. Generate two large random primes, p and q , of approximately equal size such that their product $n = pq$ is of the required bit length, e.g. 1024 bits.
2. Compute $n = pq$ and $\phi = (p-1)(q-1)$.
3. Choose an integer e , $1 < e < \phi$, such that $\gcd(e, \phi) = 1$.
4. Compute the secret exponent d , $1 < d < \phi$, such that $ed = 1 \pmod{\phi}$.
5. The public key is (n, e) and the private key is (n, d) . The values of p , q , and ϕ should also be kept secret.

- **Implementation of DES Algorithm**

The **Data Encryption Standard (DES)** is a cipher (a method for encrypting information) selected as an official Federal Information Processing Standard (FIPS) for the United States. DES consequently came under intense academic scrutiny which motivated the modern understanding of block ciphers and their cryptanalysis.

Double-DES not same as some other single-DES use, but have meet-in-the-middle attack. This attack occurs whenever using a cipher twice. To avoid this kind of attacks we have used Triple-DES with Three-Keys ($C = E_{K3}[D_{K2}[E_{K1}[P]]]$).

2.3 USER INTERFACE REQUIREMENTS.

XML Parser

Perform basic xml file reading and writing operations. We can parse the xml tree through each and every node and their attributes.

Policy Configuration

When a subject subscribes to the Owner, it receives back an object called subject policy configuration, providing information on the access control policies that the subject satisfies.

Subject Publisher Interaction

For each document the Publisher is entitled to manage, the Owner sends the corresponding security enhanced (SE-xml) document and the corresponding secure structure, along with the access control policies and policy configuration for each element (node) of xml tree.

Reply Generation

When a subject submits a query to a Publisher, the Publisher first determines the set of nodes that need to be returned to him/her. Such nodes are determined by evaluating the query on the SE-XML version of the requested document(s) and by pruning from the set of nodes returned by the evaluation, those nodes corresponding to portions for which subject does not possess appropriate authorizations.

Subject Verification

A subject upon receiving a reply document and a secure structure can verify the authenticity and the completeness of the corresponding query answer.

III. DEVELOPMENT ENVIRONMENT

3.1 HARDWARE REQUIREMENTS

| | | |
|-----------------|---|--------------------|
| HARD DISK | : | 80 GB |
| MAIN MEMORY | : | 512 MB RAM |
| PROCESSOR | : | PENTIUM IV 2.4 GHz |
| FDD | : | 1.44 MB |
| CD-DRIVE | : | 52X SAMSUNG |
| KEYBOARD | : | 105 KEYS |
| MONITOR | : | SVGA OR VGA |
| POINTING DEVICE | : | OPTICAL MOUSE |

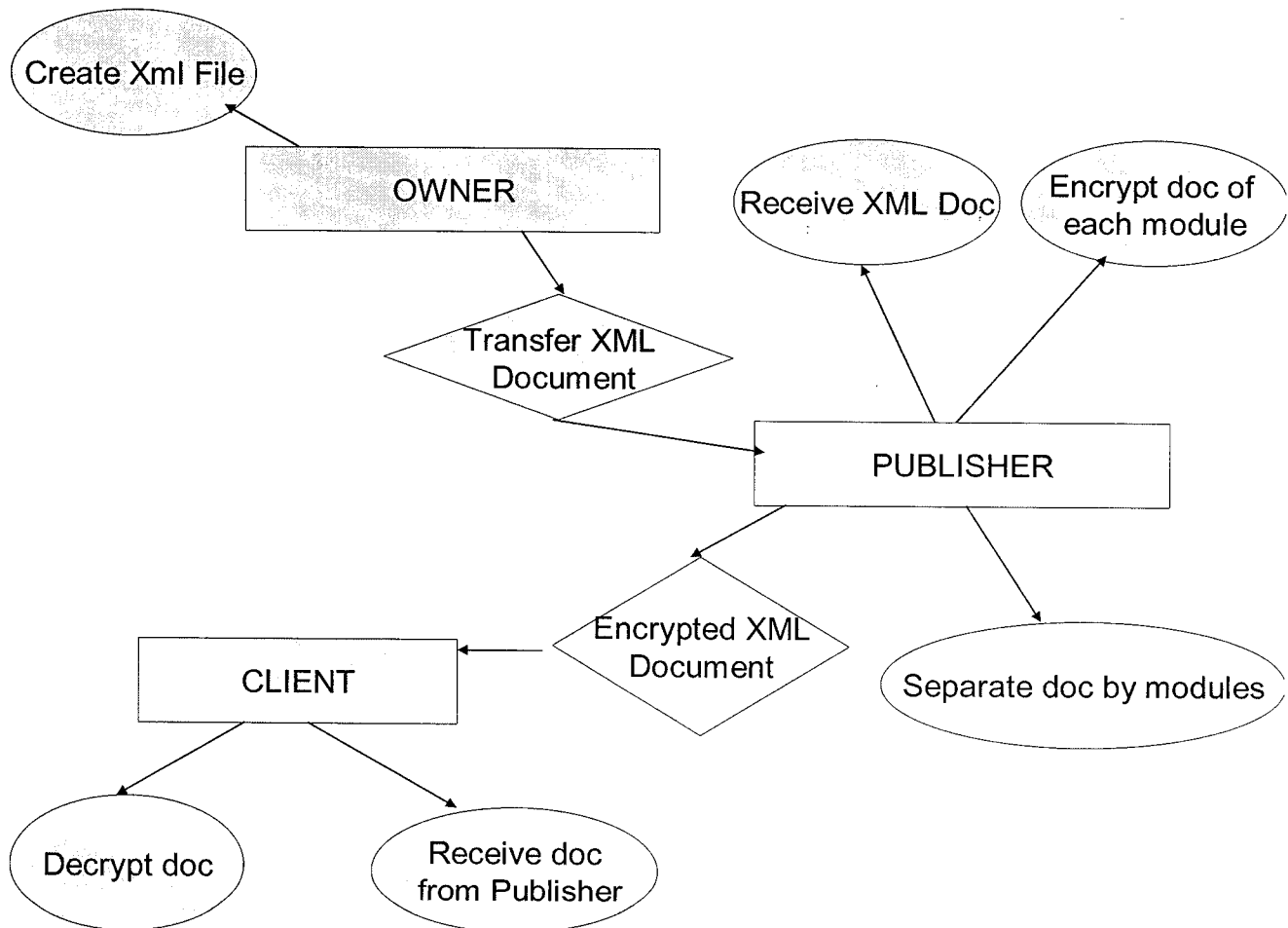
3.2 SOFTWARE REQUIREMENTS

| | | |
|------------------|---|--------------------|
| OPERATING SYSTEM | : | WINDOWS XP |
| LANGUAGE | : | VB.NET 2003 |
| DATABASE | : | MY SQL SERVER 2003 |

IV SYSTEM DESIGN

4.1 DATA MODEL

4.1.1 ER DIAGRAM



4.1.2. DATA DICTIONARY

4.1.2.1 SUBJECT TABLE



P-2298

| FIELD NAME | DATA TYPE | SIZE | KEY |
|--------------|-----------|------|-------------|
| SUBJECT ID | VARCHAR | 25 | NOT NULL |
| SUBJECT NAME | VARCHAR | 25 | NOT NULL |
| PRIVATE KEY | VARCHAR | 100 | PRIMARY KEY |
| PASSWORD STR | VARCHAR | 100 | NOT NULL |
| POLICY ID | VARCHAR | 25 | FOREIGN KEY |

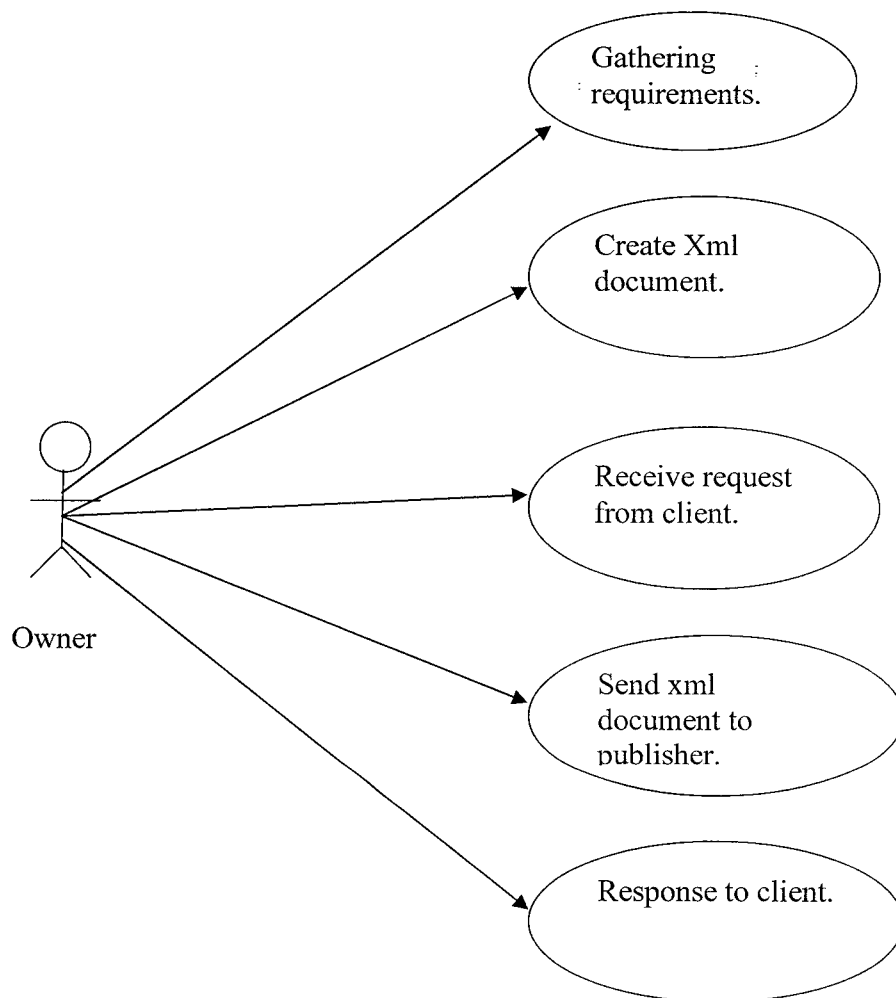
4.1.2.2 POLICY TABLE

| FIELD NAME | DATA TYPE | SIZE | KEY |
|-------------|-----------|------|----------|
| POLICY ID | VARCHAR | 25 | NOT NULL |
| POLICY NAME | VARCHAR | 25 | NOT NULL |

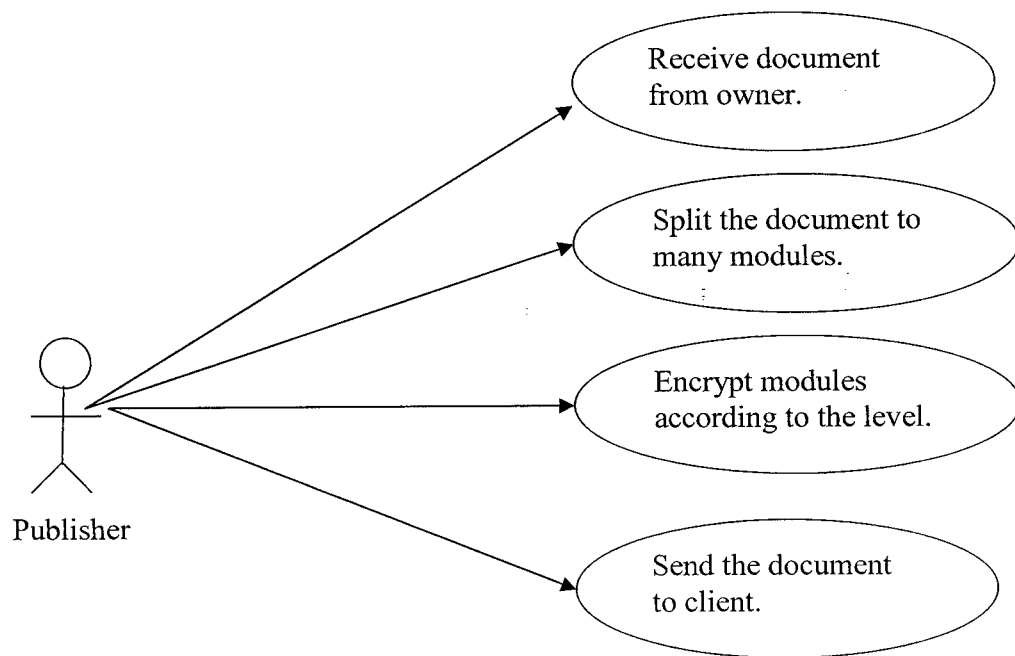
4.2 PROCESS MODEL

4.2.1 USECASE DIAGRAM

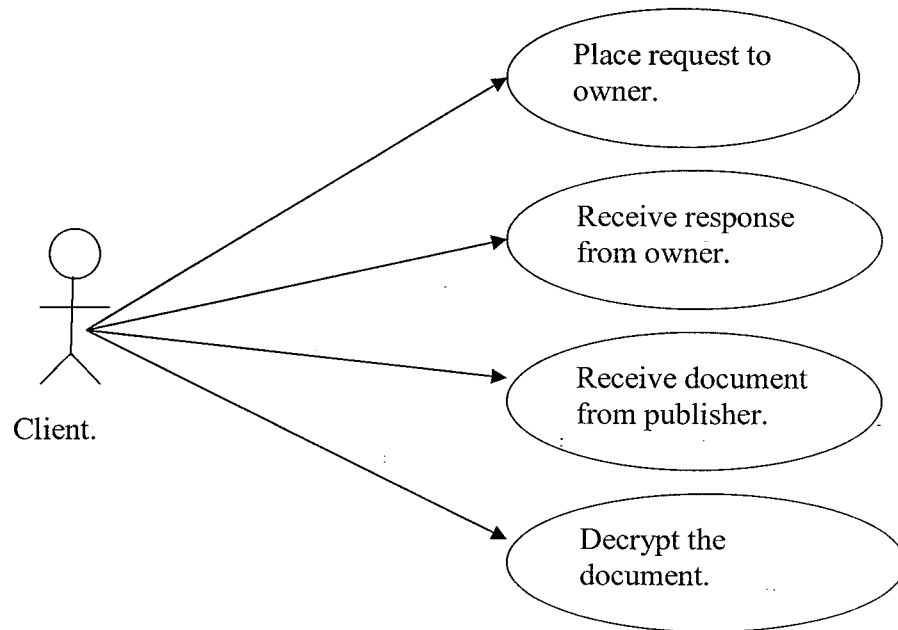
4.2.1.1 CLIENT USECASE:-



4.2.1.2 PUBLISHER USECASE:-

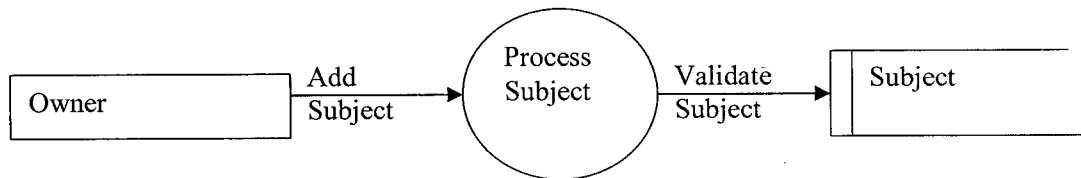


4.2.1.3 CLIENT USECASE:-

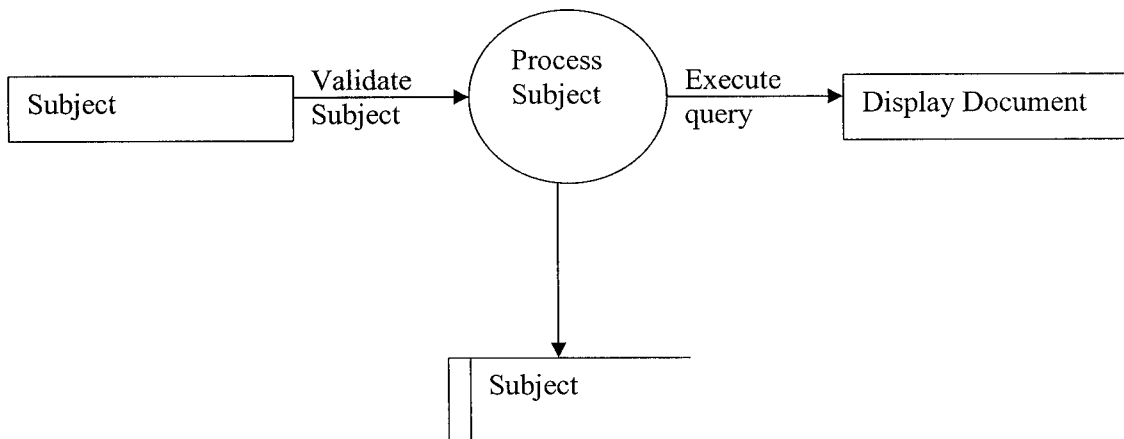


4.2.2 DATA FLOW DIAGRAM

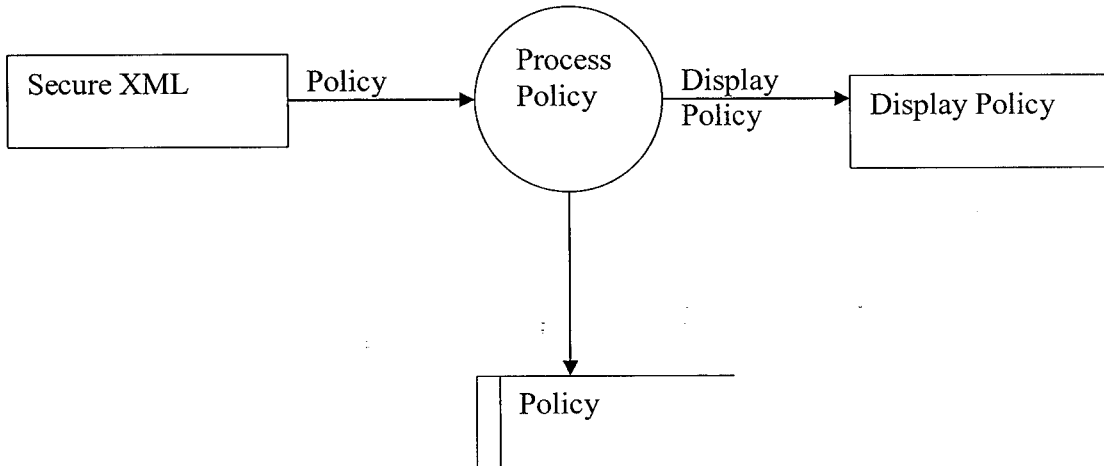
4.2.2.1 Create and validate subjects



4.2.2.2 Execute query by subjects



4.2.2.3 Policy details



V Architectural Details

5.1 Program Design Language

5.1.1 VB.NET

- **Working With Forms and Controls**

Forms allow us to work visually with controls and other items from the tool box. In VB.NET forms are based on the System.Windows.Forms namespace and the form class is System.Windows.Forms.Form.

A control is an object that can be drawn on to the form to enable or enhance user interaction with the application.

- **Working With Menus**

In this project, we used to control for three main menus namely Owner, Publisher, and Subject.

Owner menu has sub menus through which we can access the forms of Policy Master, Subject Configuration etc as sub menus in which we can enter the details and store in the database.

Likewise all other menus are tied up with their corresponding forms through sub menus.

5.1.2. NET FRAMEWORK:

- The .NET Framework is the infrastructure for the new Microsoft .NET Platform.
- The .NET Framework is a common environment for building, deploying, and running Web applications and Web services.

- The .NET Framework contains a common language runtime and common class libraries –like ADO .NET, ASP.NET and Windows Forms to provide advanced standard services that can be integrated into a variety of computer systems.
- The .NET Framework provides a feature-rich application environment, simplified development and easy integration between a numbers of different developments languages.

- **COMMON LANGUAGE RUNTIME:**

One of the design goals of .NET Framework was to unify the runtime engines so that all developers could work with a set of runtime services. The .NET Framework's solution is called the Common Language Runtime (CLR). The CLR provides capabilities such as memory management, security, and robust error handling to any language that work with the .NET Framework.

- **.NET CLASS LIBRARIES:**

The .Net Framework provides many classes that help developers re-use code. The .Net class libraries contain code for programming topics such as threading, file I/O, database support, XML parsing, and data structures, such as stacks and queues, this entire class library is available to any programming languages that support the .NET Framework.

Because all languages now support the same runtime, they can reuse any class that works with the .NET Framework. This means that any functionality available to one language will also be available to any other .NET language.

5.1.3. SQL SERVER:

STRUCTURED QUERY LANGUAGE (SQL):

SQL (pronounced SEQUEL) is the programming language that defines and manipulates the database. SQL databases are relational databases, which mean data is stored in a set of simple relations. A database can have one or more tables. Each table has columns and rows. Oracle stores each row of a database table containing data for less than 256 columns as one or more row pieces. A table that has an employee database, for example, can have a column called employee number and each row in that column is an employee's employee number.

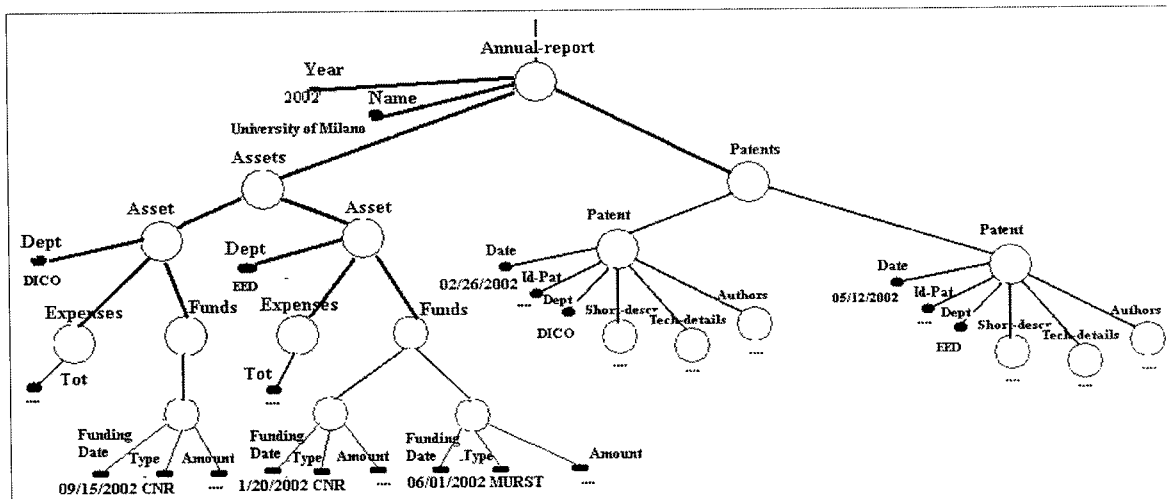
Data can be defined and manipulated in a table with SQL statements. SQL's data definition language (DDL) statements are used to define data. DDL statements include statements for creating and altering databases and tables.

Update, delete, or retrieve a data in a table are done by SQL's data manipulation language (DML). DML statements include statements to alter and fetch data. The most common SQL statement is the `SELECT` statement, which retrieves data from the database.

In addition to SQL statements, the Oracle server has a procedural language called PL/SQL. PL/SQL enables programmers to program SQL statements. It lets user control the flow of a SQL program, use variables, and write error-handling procedures.

5.1.4. Basic Concepts of XML

Building blocks of XML documents are nested, tagged elements. Each tagged element as zero or more sub elements, zero or more attributes, and may contain textual information (data content). Elements can be nested at any depth in the document structure. Attributes can be of different types, allowing one to specify element identifiers (attributes of type ID), additional information about the element (e.g., attributes of type CDATA containing textual information), or links to other elements of the document (attributes of type URI(s)).



5.1.4.1 XML Design.

Based on the above definition, an XML document can be represented as a graph. In the graph representation adopted, white nodes represent elements, whereas black nodes represent attributes. The graph representation contains edges representing the element-attribute and the element-subelement relationships, and link edges, representing links between elements. Solid lines represent edges, whereas dashed lines represent link edges.

5.1.5. Architecture of Third Party Distribution.

1. Merkle Hash Trees for XML Documents

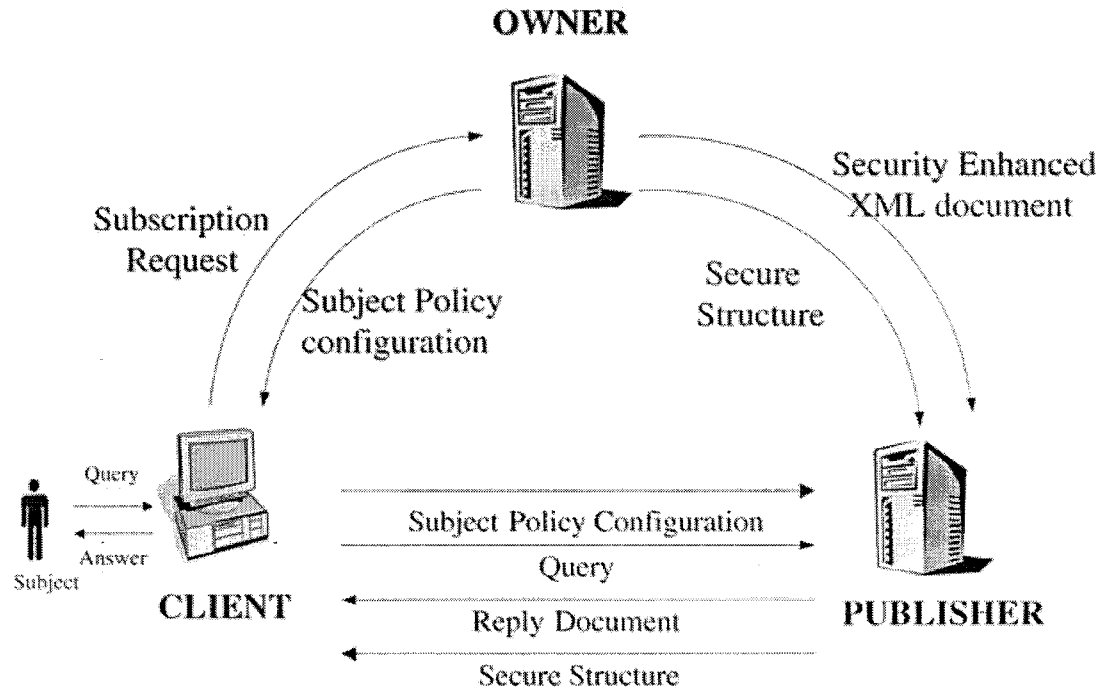
Authenticity is ensured by using the Merkle tree authentication mechanism proposed in and adapting it to XML. The method we propose allows a subject to prove source authenticity as well as the authenticity of both the schema and the content of a document. To accomplish this goal, the idea is to associate a hash value with each node in the graph representation of an XML document. More precisely, the hash value associated with an attribute is obtained by applying a hash function over the concatenation of the attribute value and the attribute name.

By contrast, the hash value associated with an element is the result of the same hash function computed over the concatenation of the element content, the element tag name, and the hash values associated with its children nodes, both attributes and elements. Hash values associated with the nodes of an XML document are computed by the Merkle hash function.

2. Access Control Model for XML Documents

Access control policies are specified by the Owner according to the access control model presented, whose main characteristics are summarized in what follows. In this model, subjects are qualified by means of credentials. A credential is a set of properties concerning a subject that are relevant for security purposes (for example, age, position within an organization). Credentials are encoded using an XML-based language. Access control policies specify conditions on the credentials and properties of the credentials, using an XPath-compliant language.

The access control model provides varying access granularity levels, and can express policies that apply to: 1) all the instances of a DTD/XML Schema, 2) collections of documents not necessarily instances of the same DTD/XML Schema, and 3) selected portions within a document(s), or a link (or a set of links). This set of granularity levels is complemented with the possibility of specifying access control policies based on the document content, in addition to the document structure. Like credentials, access control policies are also encoded using the X-Sec language. We use the term Policy Base (PB) to denote the XML file encoding the access control policies of the Owner.



5.1.5.1 System architecture.

3. SUBJECT-OWNER INTERACTION

When a subject subscribes to the Owner, it receives back an object called subject policy configuration, providing information on the access control policies that the subject satisfies.

4. OWNER-PUBLISHER INTERACTION

For each document the Publisher is entitled to manage, the Owner sends the corresponding security enhanced document and the corresponding secure structure.

Security Enhanced XML Document

The first information the security enhanced document contains is which access control policies apply to the corresponding document. Policy information is specified at the element level since different access control policies can apply to different elements and/or attributes of the same document. The idea is to encode information about the set of policies that apply to a specific element into a string of hexadecimal values, called policy configuration, and to store this string as an additional attribute of the corresponding element within the security enhanced version of the document (this attribute is called PC).

Secure Structure

To prove completeness of XML queries, a subject receives from the Publisher the secure structure of the XML documents on which the query is performed. In this section, we show how the secure structure is generated. The basic idea is to supply the subject with the structure of the XML document on which the query is submitted where, with the term structure, we mean the XML document without data contents, that is, containing only the names of the tags and attributes of the XML document.

The subject is then able to locally perform on the structure all queries whose conditions are against the document structure of the original document. Thus, the subject can match the result with the answer sent by the Publisher. Obviously, in such a basic approach, the subject is able to view the tag and attribute names of the whole document and thus also those referring to portions he/she may not be authorized to access. To overcome this drawback, the secure structure of the XML document is generated by hashing with a standard hash function each tag and attribute name. Since the value returned by the hash function may contain characters not allowed in an XML well-formed document⁷ (e.g., "< ", "= "), during the generation of the secure structure the resulting hash values are encoded into an integer-based representation. Moreover, to be

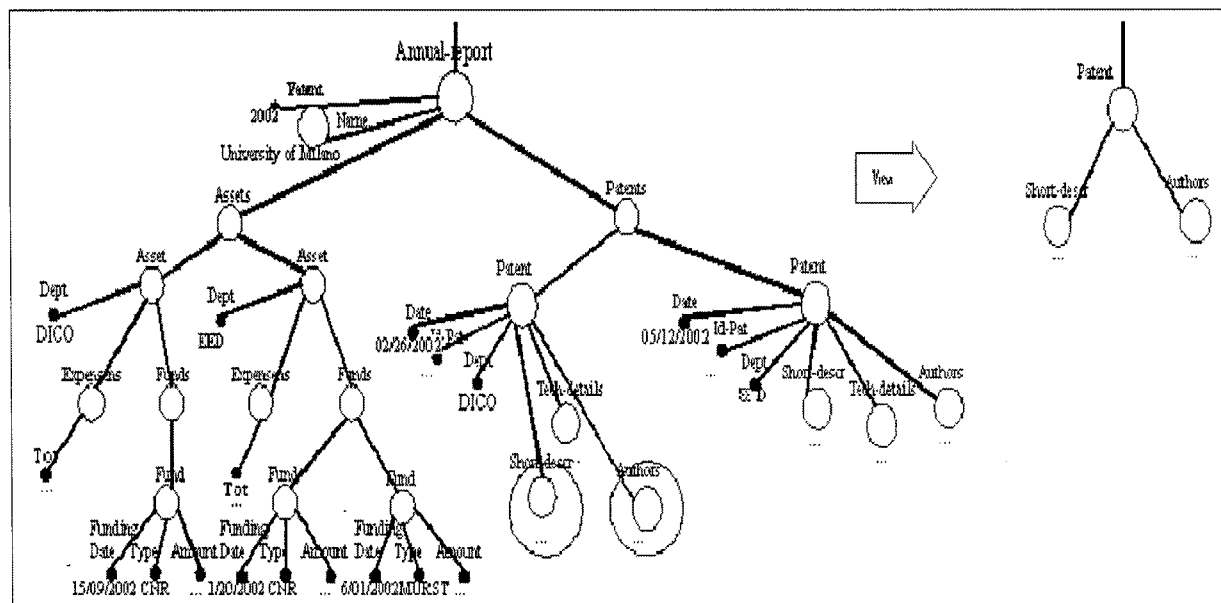
compliant with the XML syntax, we insert symbol “x” as a prefix of the integer before storing it as a tag name. Additionally, to extend the set of queries for which it is possible to prove completeness, we also insert the hashed attribute values of the XML document in the secure structure.

The node-set returned by evaluating a query on the secure structure could be a superset of the nodes the subject is entitled to see according to the Owner access control policies. Thus, in order to verify the completeness, the subject must also consider the access control policies specified on the document. For this reason, the secure structure contains also the Policy, PC, and PCATTR elements, whose tag name and content are not hashed. Additionally, in order to prevent alterations by the Publisher, the Owner computes the Merkle Signature of the secure structure.

5. SUBJECT-PUBLISHER INTERACTION

When a subject s submits a query to a Publisher, the Publisher first determines the set of nodes that need to be returned to s . Such nodes are determined by evaluating the query on the SE-XML version of the requested document(s) and by pruning from the set of nodes returned by the evaluation, those nodes corresponding to portions for which s does not possess appropriate authorizations.

Information on access control policies that apply to s are transmitted by s to the Publisher when submitting the access request. More precisely, the subject sends the Publisher his/her policy configuration together with the submitted query. Then, the set of nodes to be returned to s is complemented with additional information, which is used by s to authenticate the answer and to verify its completeness. In particular, all the additional information needed to verify the authenticity, as well as the nodes of the requested document(s) to be returned to the subject, are organized into an XML document, called reply document.



5.1.5.2 Subject-Publisher Interaction.

Reply Document

For the sake of simplicity, in the following, we consider only queries on a single document. The methods we present, however, can be easily extended to queries referring to multiple documents. In general, an answer to a query may contain only selected portions of a given XML document.

For instance, consider the XML document, and suppose that an EED professor submits a query q asking for all DICO patents. According to the access control policies, the nodes answering q and for which an EED professor has the necessary authorizations are the Short-descr and Authors elements.

We assume the existence of a function, called View q , which takes as input a query q and the policy configuration of a subject s , and turns the set of nodes answer to q and for which s has the necessary authorizations into a well-formed XML document.

VI. TESTING

Software testing is an important element of S/W quality assurance and represents the ultimate review of specification, design and coding. The increasing visibility of S/W as a system element and the costs associated with an S/W failure are motivating forces for well planned, through testing. Thus a series of testing are performed for the proposed system before the system is ready for user acceptance testing.

The testing of “Secure Third-Party Distribution of Xml Documents” is entering data using the created forms. All the forms are checked for error and corrections were made after receiving feedback from the user. If testing is conducted successfully according to the objectives stated above, it will uncover errors in the software. Also, testing demonstrates that software functions appear to be working according to the specification, that performance requirements appear to have been met.

6.1 TESTING METHODOLOGIES

Testing is a set of activities that can be planned in advance and conducted systematically. Testing is a very important stage of a software include Unit Testing, Integration Testing and Deployment testing.

Unit Testing

Unit testing focuses verification effort on the smallest unit of S/W design i.e., the module. The unit testing is always white-box oriented and the step can be conducted in parallel for modules. In Secure Third-Party Distribution of Xml Documents, unit testing is done to uncover the following errors: The module interfaces are tested to ensure that information flows properly into and out of the program and is equal to the number of arguments in stored procedure checking the parameter and argument attributes matching the stored procedures. Data structure testing is used to ensure that the data stored temporarily maintains integrity.

Integration Testing

Data can be lost across an interface; one module can have inadvertent, adverse effects on another, sub-function, when combined, may not produce the desired major functions, individually acceptable imprecision may be magnified to unacceptable levels; global data structures can present problems. Integration testing is a systematic technique for constructing the program structure while at the same time conducting test to uncover errors associated with interfacing. The objective is to take unit-tested modules and build a program structure that has been dictated by design.

In Third Party Distribution of Xml Documents, the programs in various modules that are interfacing with other modules are tested thoroughly. Here we followed Top-Down integration and modules are integrated by moving downward through the control hierarchy, beginning with the Project related process, then activity related process and report generation process.

Deployment Testing

In deployment testing basically check for hard coded links. For smooth transfer of data from one page to another page in the system, we had to be sure there were no hard coded links. The scope of the objects and data was tested when they were transferred to another place.

Validation Testing

At the end of Integration testing, software is completely assembled as a package, interfacing errors have been uncovered and correction testing begins. For example, In this project subject can give ID and PASSWORD and view a particular encrypted document .here the validation testing check the authority of subjects.

VII. PERFORMANCE AND LIMITATIONS.

7.1. IMPLEMENTATION AND MAINTANANCE

Implementation is the stage of the project where the theoretical design is turned in to a working system. At this stage the main work load, the greatest upheaval and the major impact on the existing system shifts to the user department. If the implementation is not carefully planned a controlled it can cause chaos and confusion.

Implementation includes all those activities that take place to convert from the old system to the new one. The new system may be totally new, replacing an existing manual or automated system or it may be a major modification to an existing system. Proper implementation is essential to provide a reliable system to meet the organization requirements. Successful implementation may not guarantee improvement in the organization using the new system, but improper installation will prevent it.

The process of putting a developed system in actual use is called system implementation. This includes all those activities that take place to convert from the old system to the new system. The system can be implemented only after thorough testing is done and if it is found to be working according to the specifications. The system personnel check the feasibility of the system.

The most crucial stage is achieving a new successful system and giving confidence on the new system for the user that it will work efficiently and effectively. It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the change over.

The most complex system being implemented, the more involved will be the system analyses and the design effort required just for implementation. The system implementation has three main aspects. They are education and training, system testing and change over.

The implementation stage involves following tasks

- Careful planning
- Investigation of system and constraints
- Design of methods to achieve the change over.
- Training of the staff in the change over phase.
- Evaluation of the change over method.

The method of implementation and the time scale to be adopted are found out initially. Next the system is tested properly and the same time users are trained in the new procedures.

7.2 IMPLEMENTATION PROCEDURES

Implementation of software refers to the final installation of the package in its real environment, to the satisfaction of the intended users and the operation of the system. In many organization some one who will not be operating it, will commission the software development project. The people who are not sure that the software meant to make their job easier. In the initial stage, doubt occur about the software but have to ensure that the resistance does not build up as one has to make sure that

- The active user must be aware of the benefits of using the system
- Their confidence in the software is built up
- Proper guidance is imparted to the user so that she/he is comfortable in using the application.

Before going ahead and viewing the system, the user must know that for viewing the result, the server program should be running in the server. If the server object is not up running on the server, the actual processes will not take place.

7. 2.1. SYSTEM MAINTENANCE

The maintenance phase of the software cycle is the time in which a software product performs useful work. After a system is successfully implemented it should be maintained in a proper manner. System maintenance is an important aspect in software development lifecycle. The need for system maintenance is for it to make adaptable to the changes in the system environment.

There may be social, technical and other environmental changes, which affect a system, which is being implemented. Software product enhancements may involve providing new functional capabilities, improving user displays and mode of interactions, upgrading the performance characteristics of the system. So only through proper system maintenance procedures, the system can be adopted to cope up with these changes.

Software maintenance of course, far more than “finding mistakes”. We may define maintenance by describing four activities that are undertaken to after a program is released for use.

The first maintenance activity occurs because it is unreasonable to assume that the software testing will uncover latent errors in a large software design. During the use of any large program, errors will occur and be reported to the developer. The process that includes the diagnosis and the correction of one or more errors is called **corrective maintenance**.

The second activity that contributes to a definition of maintenance occurs because of the rapid change that is encountered in every aspect of computing. Therefore, **adoptive maintenance**- an activity that modifies software to properly interfere with a changing environment is both necessary and common place.

Third activity that may be applied to a definition of maintenance occurs when a software package is successful. As the software is used recommendations for new capabilities, modification to existing functions and general enhancements are received from users. To satisfy request in this category, **perfective maintenance** is performed. This activity accounts for the majority of all effort expended on software maintenance.

The fourth maintenance activity occurs when software is changed to improve future maintainability or reliability, or to provide a better basics for future enhancement. Often called **preventive maintenance**, this activity is characterized by reverse engineering and reengineering techniques.

7.3. Future Enhancement

Since, in the proposed framework, a modification on a document also implies an update of the security-enhanced version and of the secure structure of the document, a key issue is the efficient management of updates. To this purpose, we discuss, for each kind of update that could occur over the Owner's XML source or the Policy Base, the corresponding updates that the Owner has to perform on the security-enhanced version and on the secure structure of the involved documents. There are two main kinds of updates that need to be considered. The first is an update of the PB, such as, for instance, an access control policy insertion, deletion, or update. In the case of policy insertion, it is necessary to update the SE-XML version and the secure structure of all the documents to which the new policy applies.

However, this update does not require too much overhead. Indeed, for each document to which the policy applies, it is only necessary to insert the identifier of the new policy in the Policy element, and to update all and only the policy configurations of those nodes to which the new access control policy applies. Management of policy updates is simpler since it is not necessary to update the Policy element, but only the policy configurations of those nodes that are influenced by the policy update. Moreover, in case of policy deletion, it is only necessary to set equal to zero the identifier of the deleted policy in the Policy element of the SE-XML version and of the secure structure of all the documents to which the revoked policy applied.

All PB updates do not require to recompute the Merkle Signature of the involved documents since this value is computed over the root of the original XML document, thus it does not include the policy configuration values. Thus, upon a modification of PB, the Owner sends the Publishers only the Policy element of all the documents affected by the policy updates and, only in case of policy insertion and modification, a set of policy configurations, one for each node affected by the policy update.

CONCLUSION

Working over the project, “**SECURED THIRD PARTY DISTRIBUTION**”, has been a great experience with a lot of exposure to various evolving software trends. The project has been found to work effectively and efficiently. It clearly gives the client a competitive advantage tool that would help improve the business’ financial bottom line.

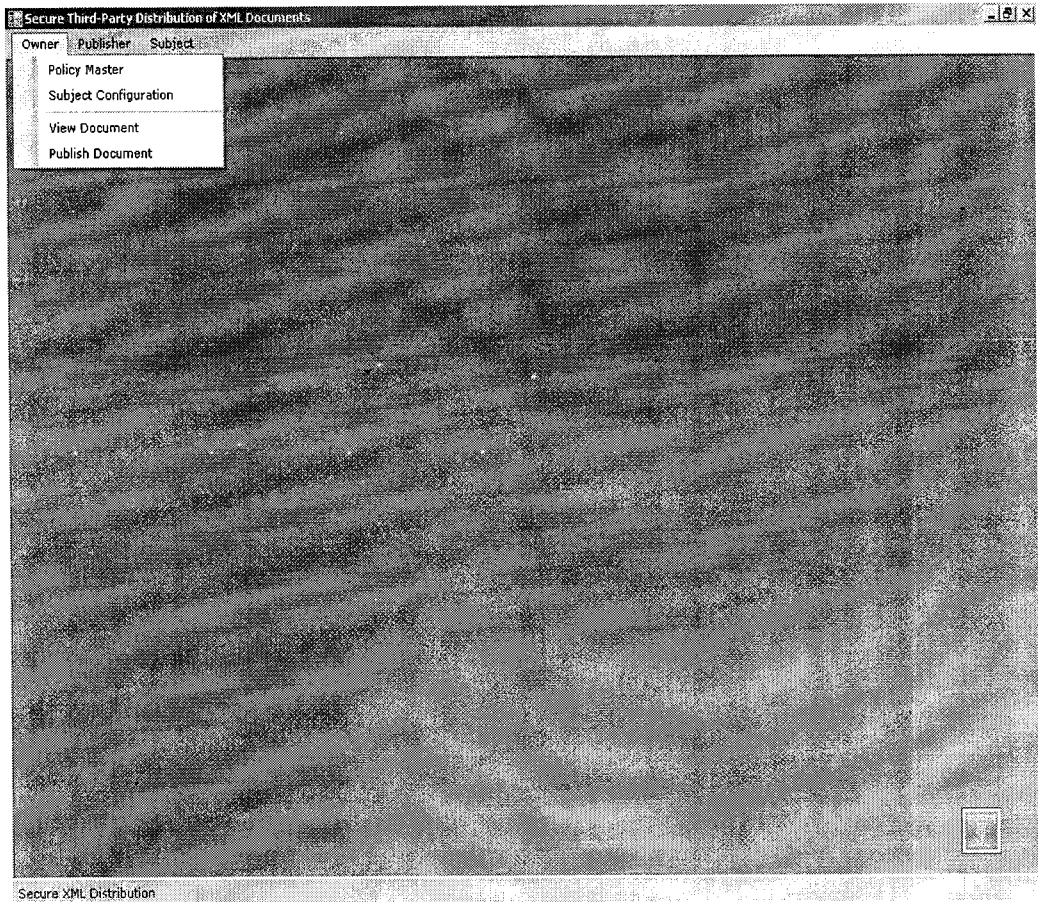
The application is formulated by analyzing the requirements of the users in the company. Each and every module has undergone various test conditions. With a full stretch testing, it has been ensured that the system can enhance ideally without any bugs or crashes, which will make the end user more compatible with the project.

The application is designed as user friendly and all the options available are clear and self explanatory so that the user can understand the system easily.

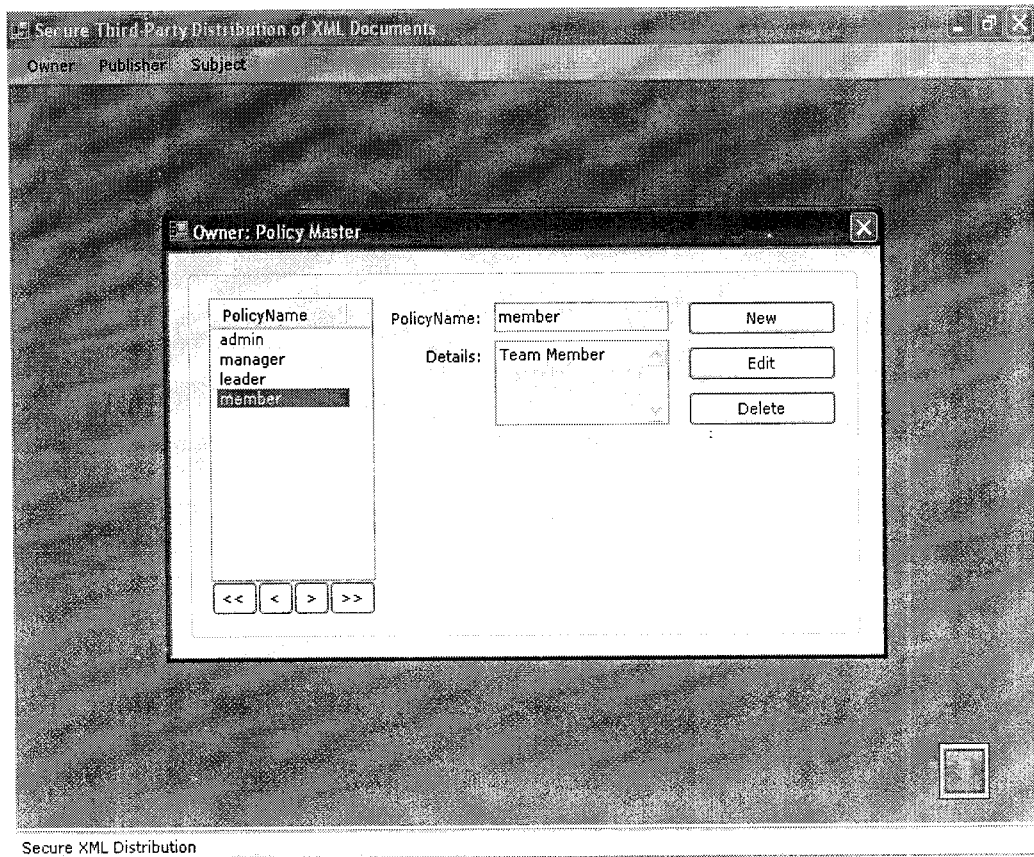
VIII APPENDICES

8.1 Sample Screens.

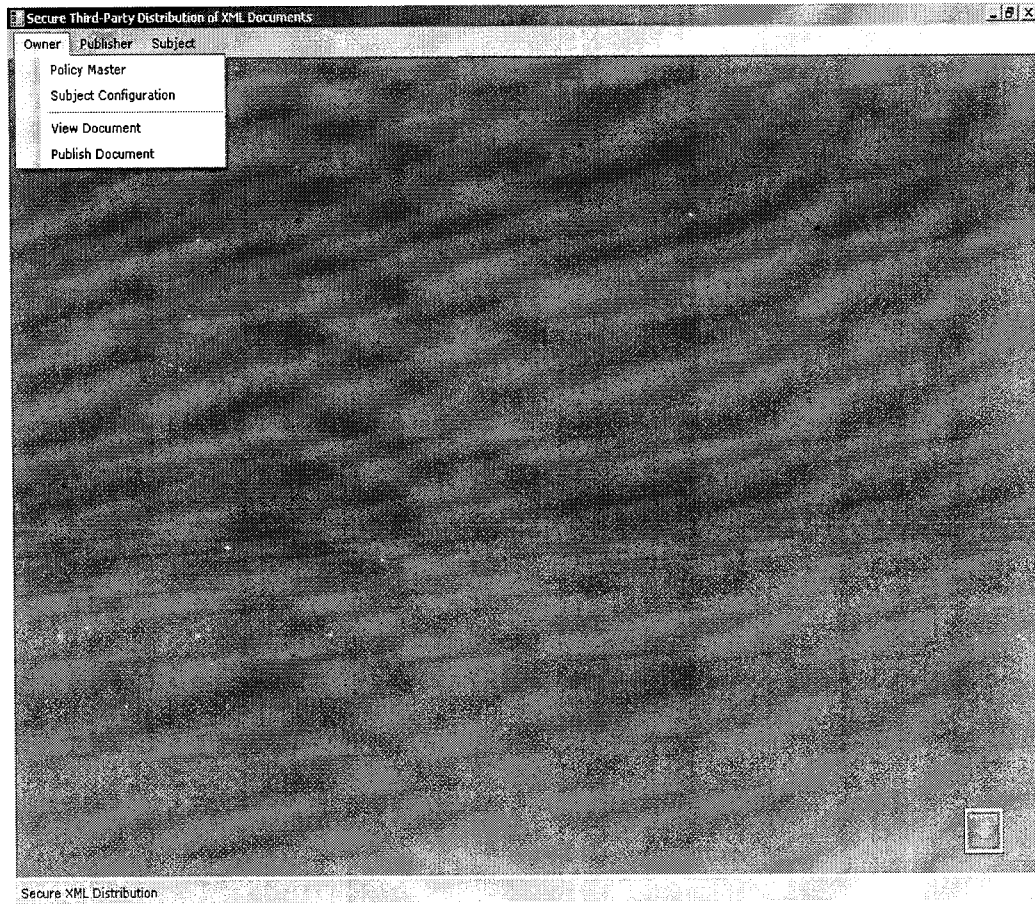
Main Form.



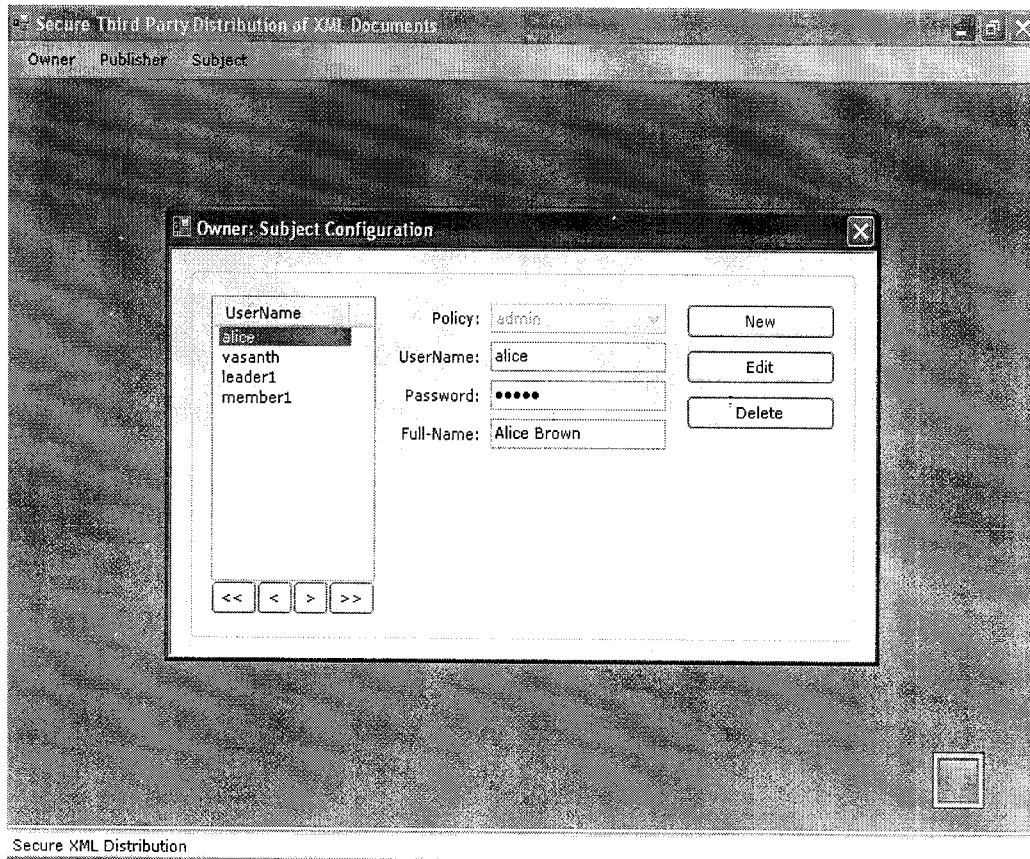
Policy Master.



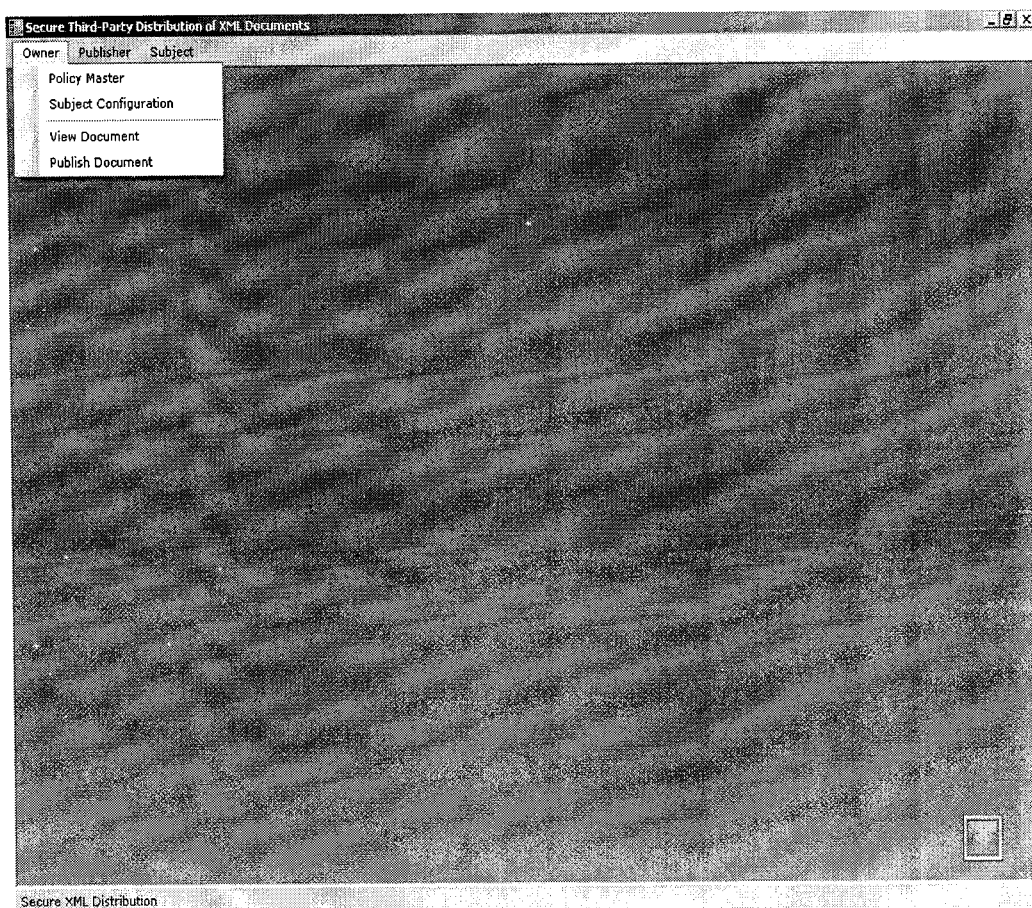
Opening Subject Configuration.



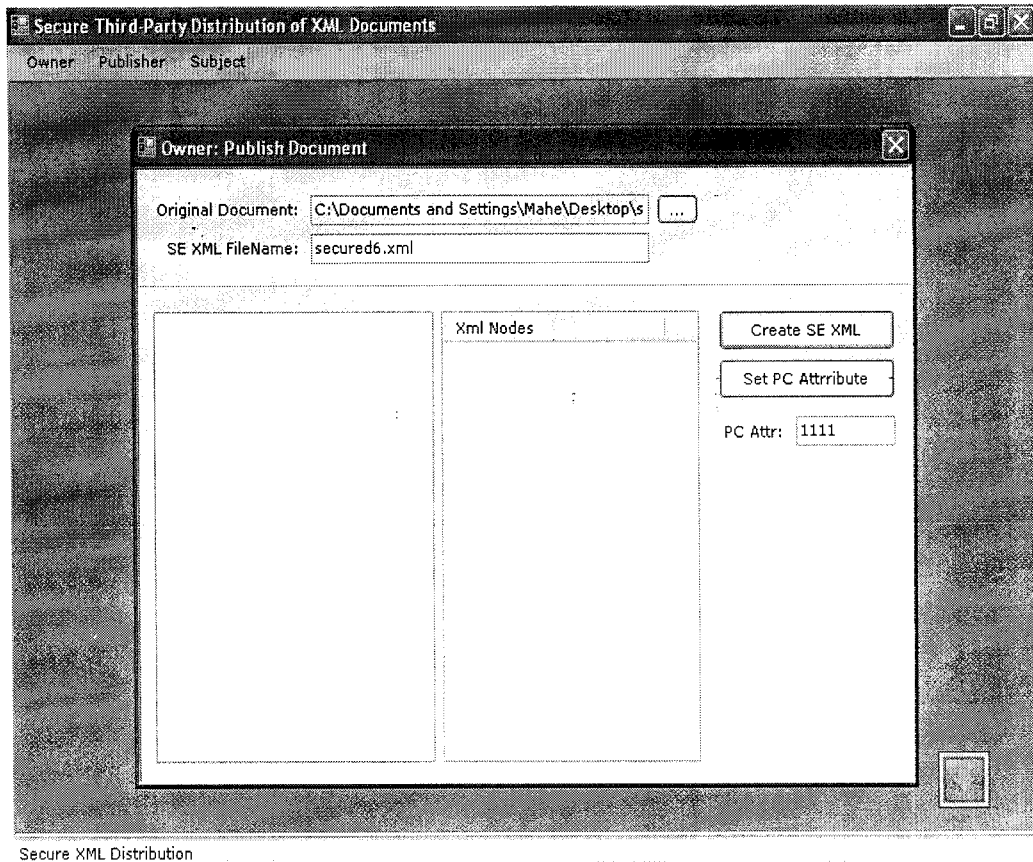
Subject Configuration.



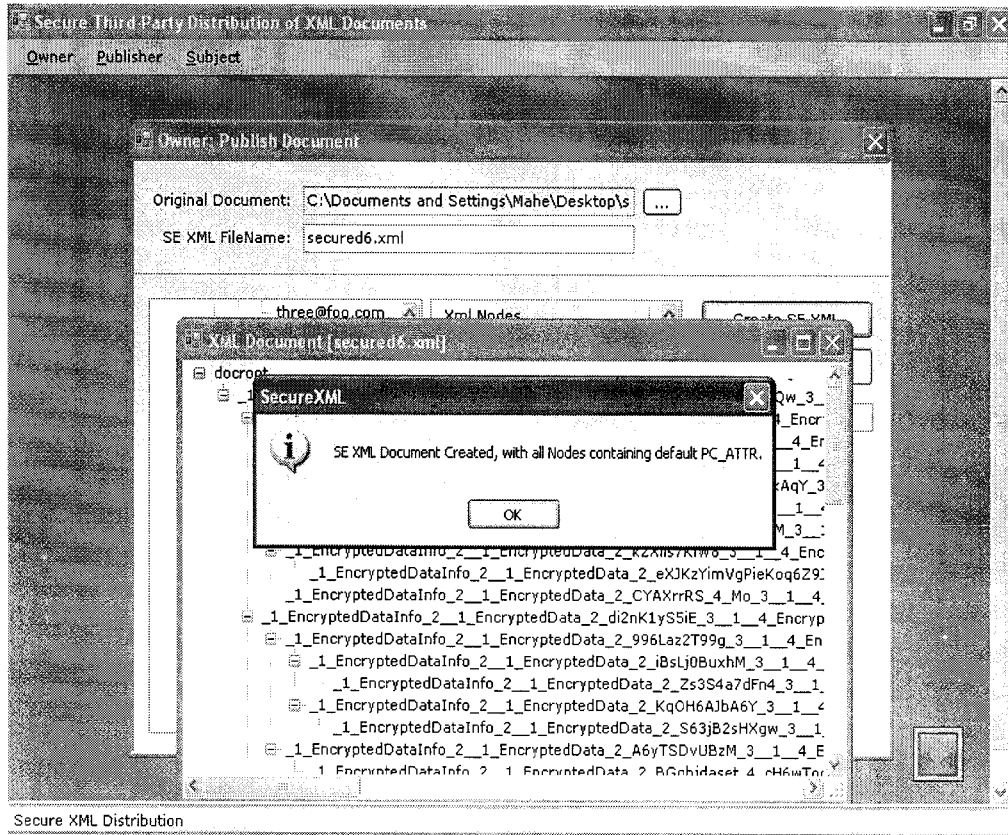
Opening Publish Document.



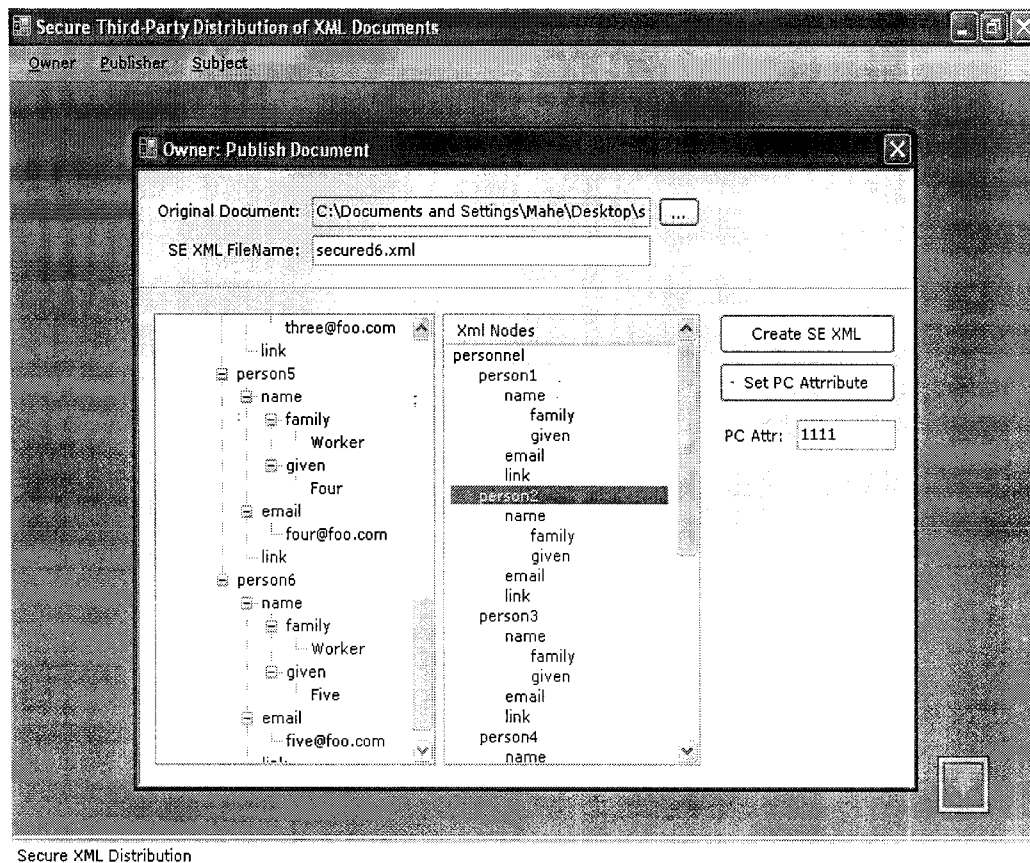
Publish Document.



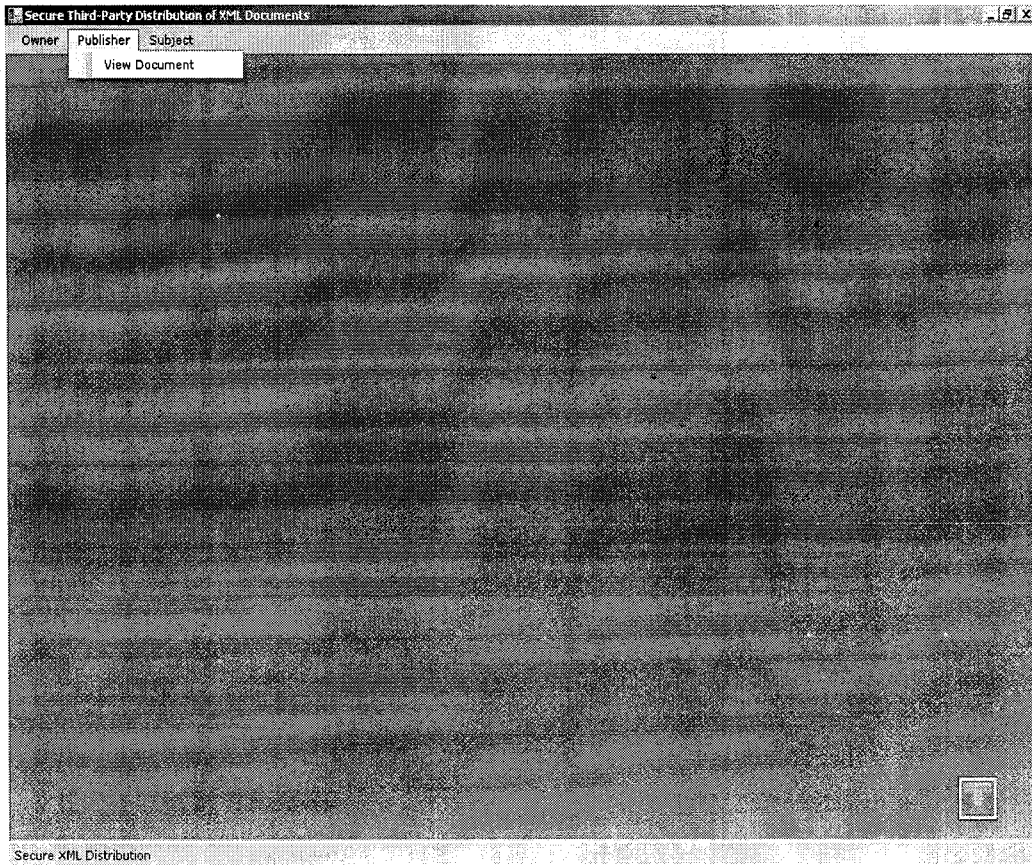
Published Document.



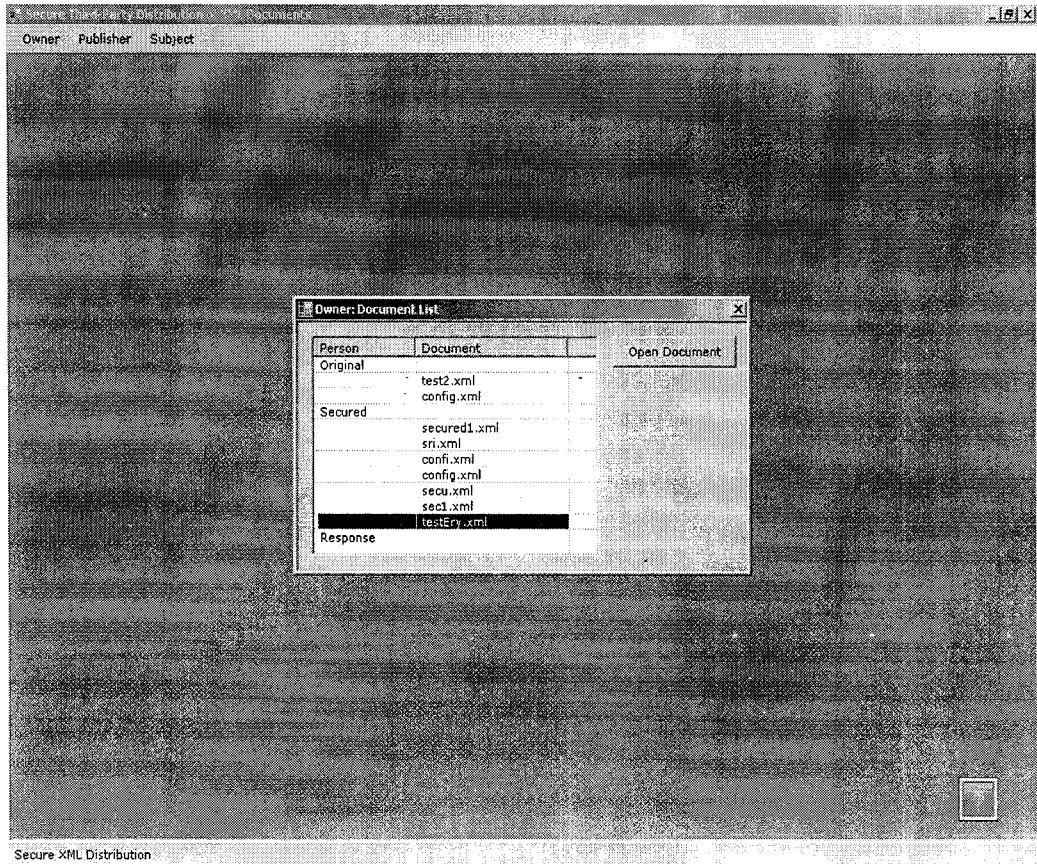
Nodes of Xml Document.



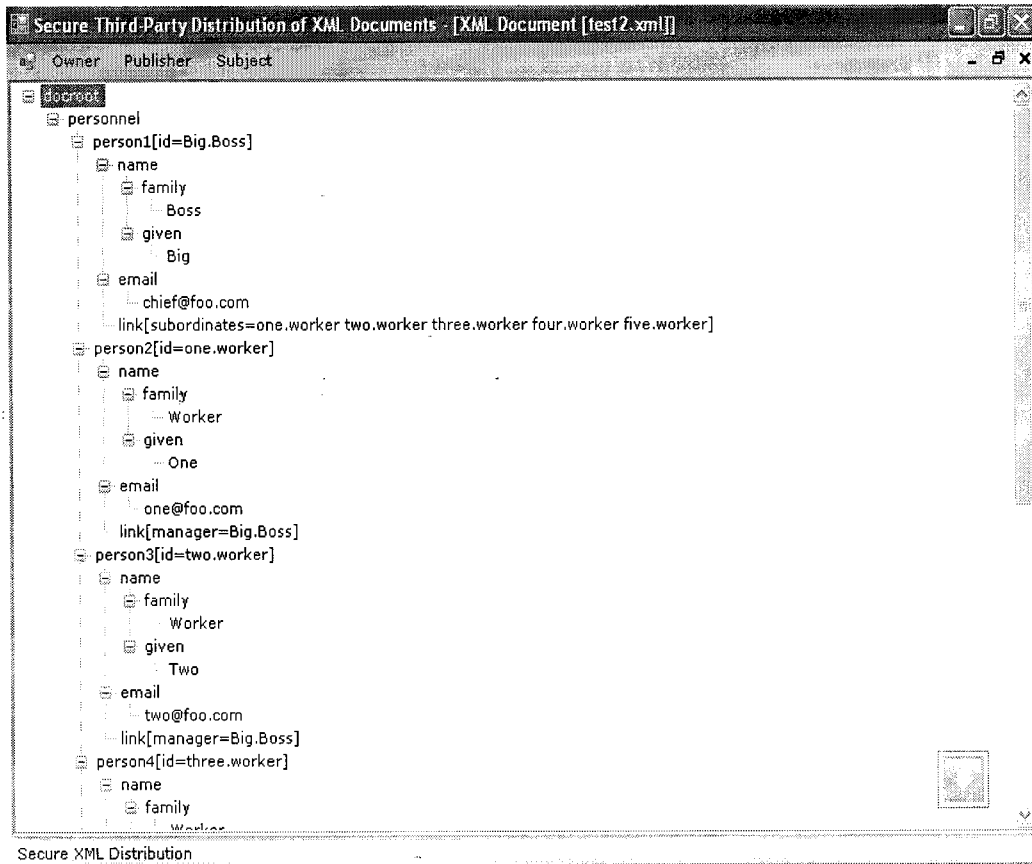
Opening View Document.



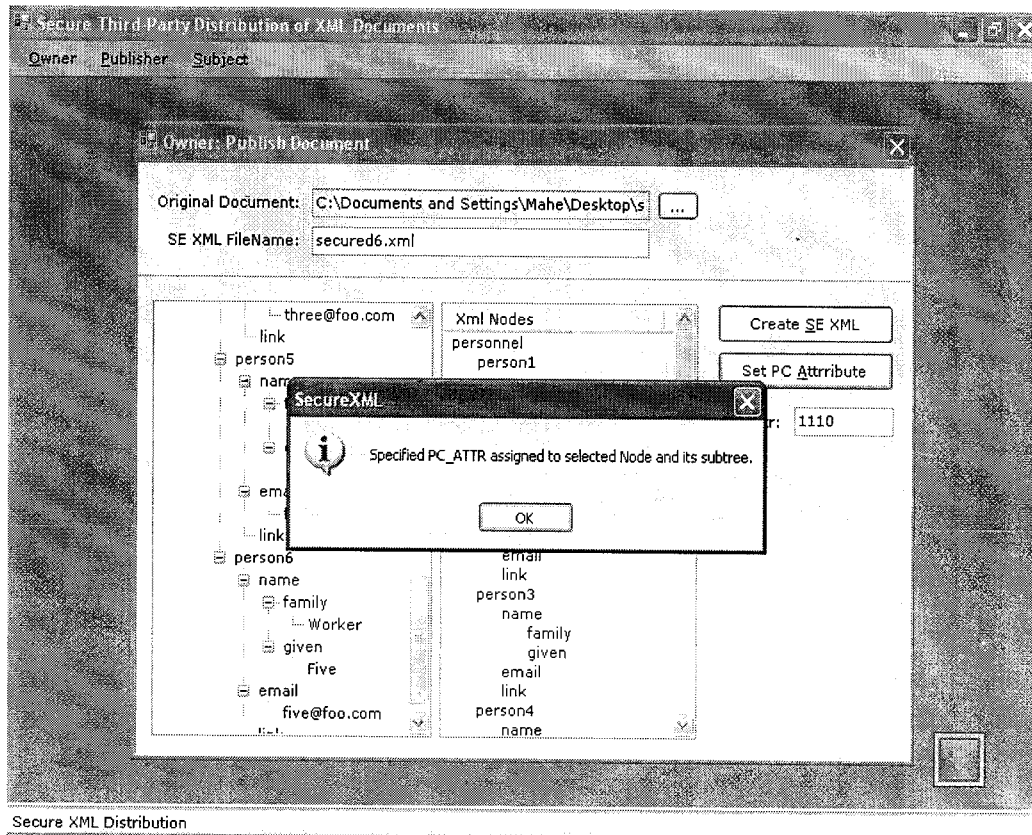
Selecting Xml Document.



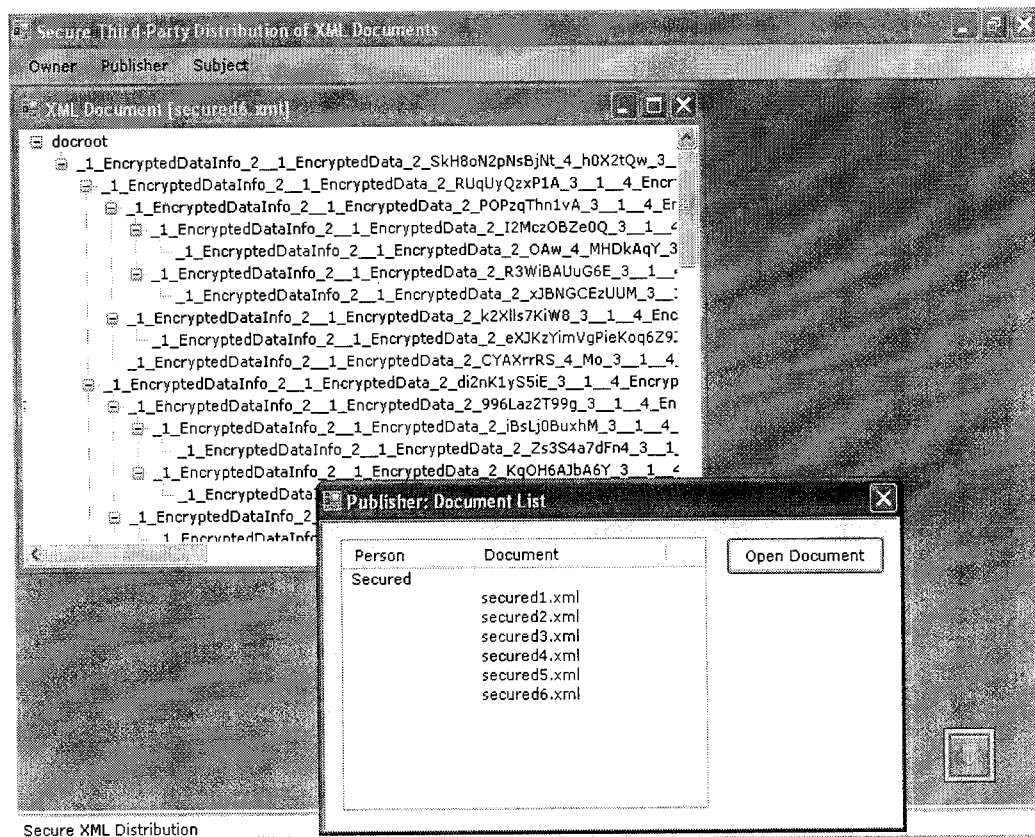
Overview Of The Document.



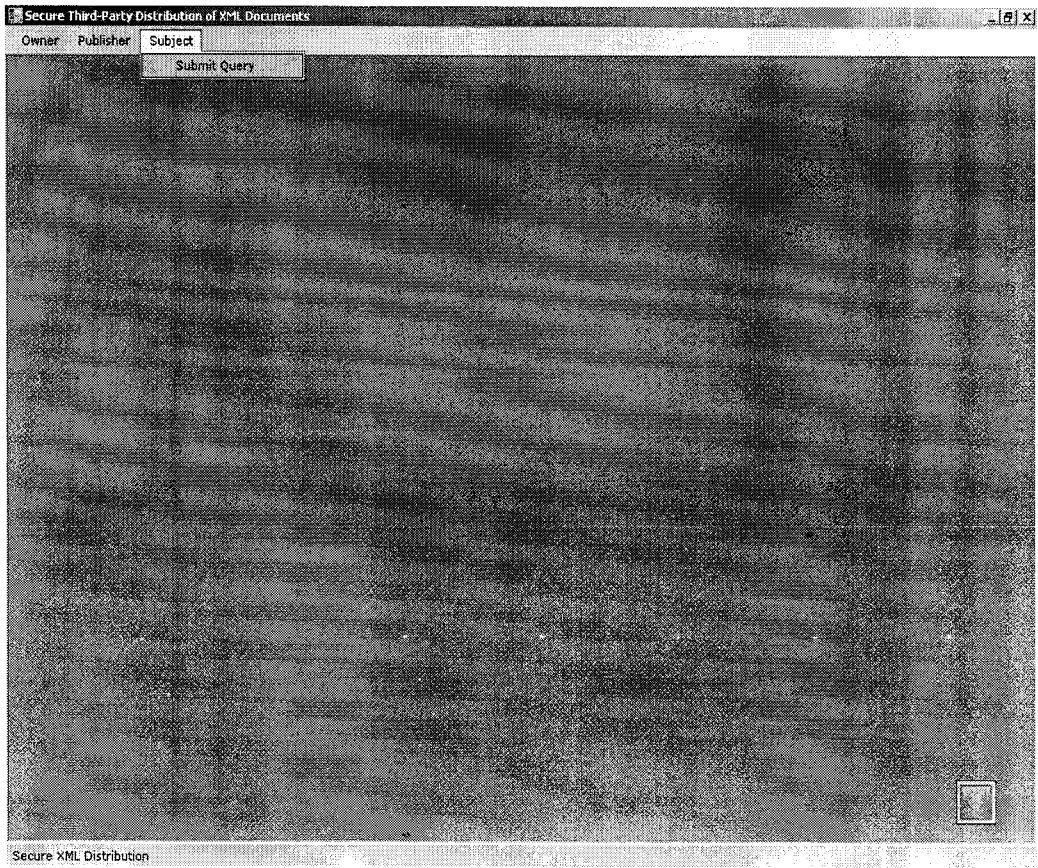
Set Attribute.



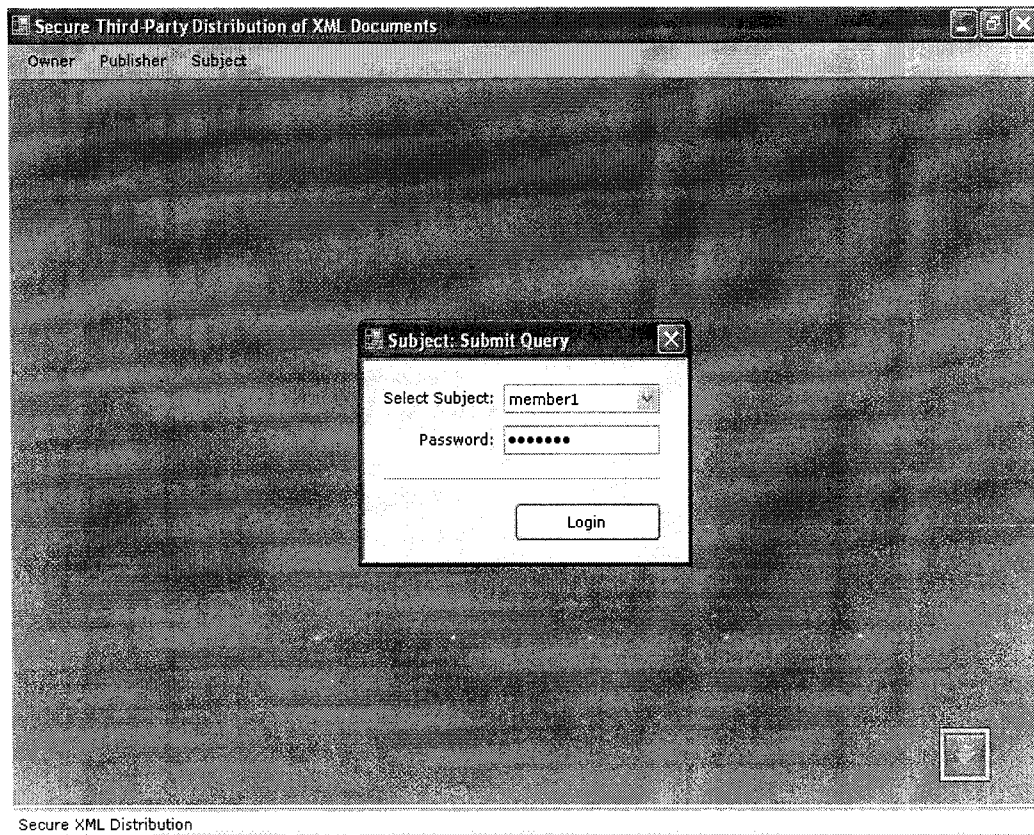
Encrypted Document.



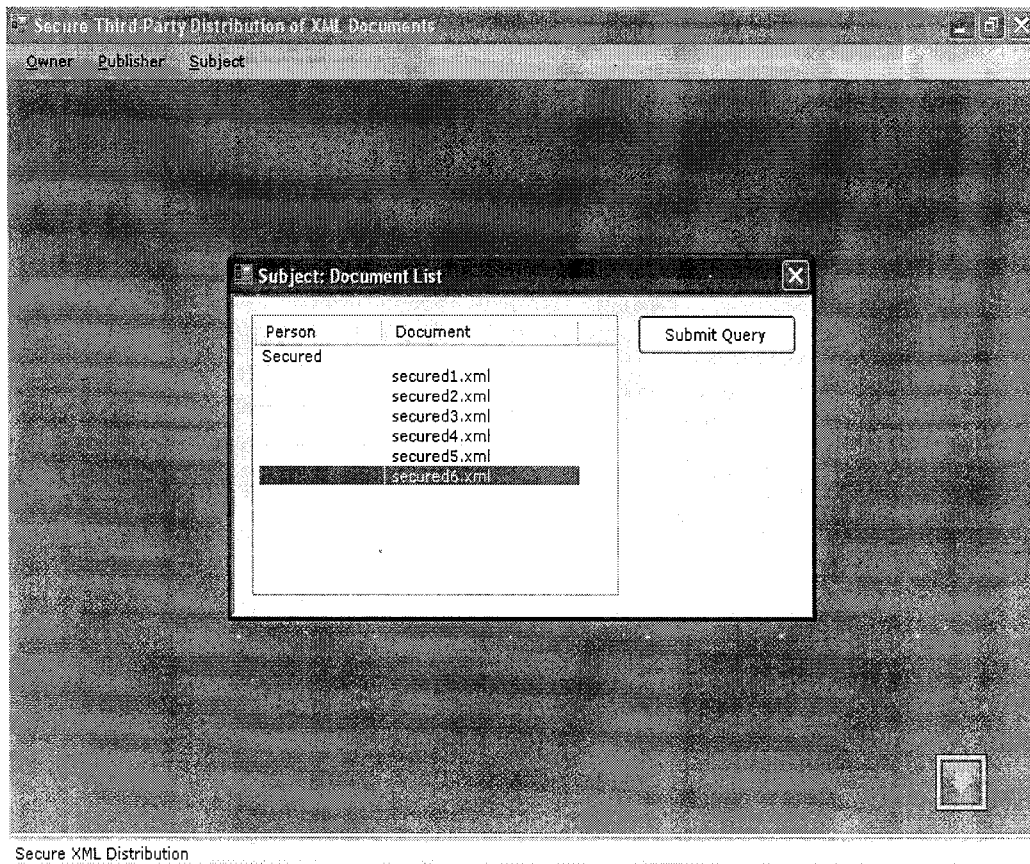
Subject Query.



Opening Members Document.



Selecting Document.



Member View.

Secure Third-Party Distribution of XML Documents - [XML Document [member1.xml]]

Owner Publisher Subject

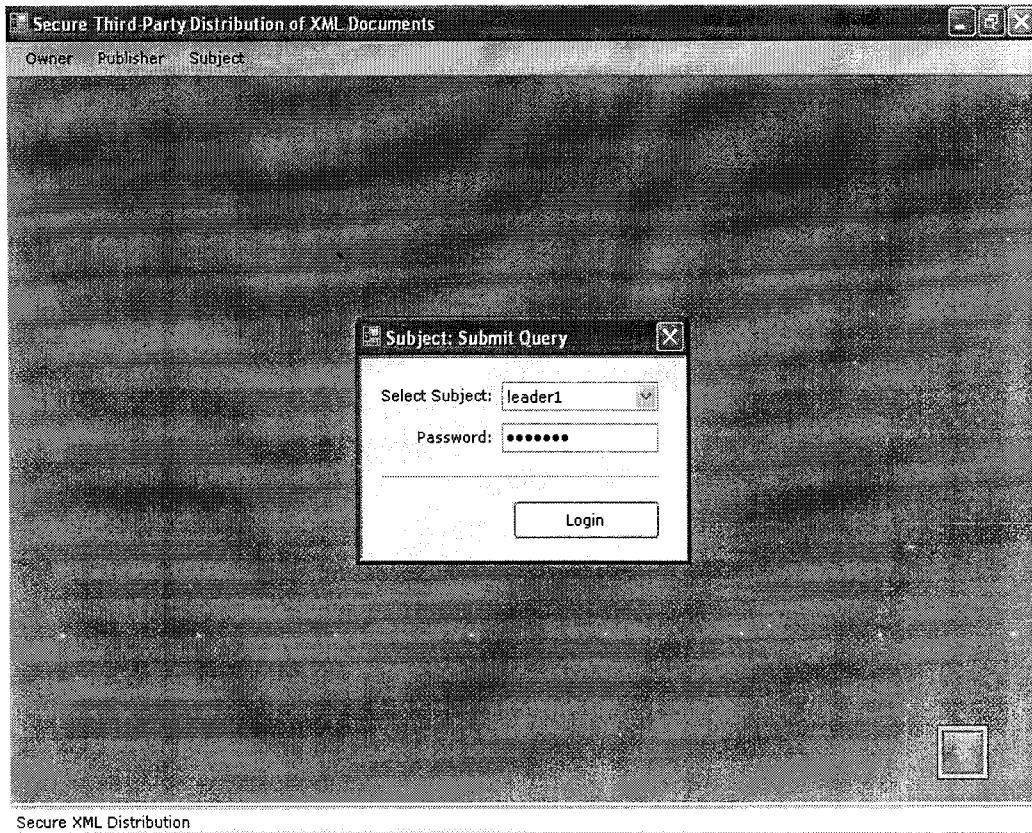
```

<?xml root="personnel"
  <person1[id=Big.Boss]
    <name
      <family>Boss</family>
      <given>Big</given>
    </name>
    <email>chief@foo.com</email>
    <link[subordinates=one.worker two.worker three.worker four.worker five.worker]></link>
    <_1_EncryptedDataInfo_2_1_EncryptedData_2_di2nK1yS5iE_3_1_4_EncryptedData_2_1_EncryptedIV_2_pIBO_5_30DgN
      <_1_EncryptedDataInfo_2_1_EncryptedData_2_996Laz2T99g_3_1_4_EncryptedData_2_1_EncryptedIV_2_gSc6N01yk
        <_1_EncryptedDataInfo_2_1_EncryptedData_2_iBsLj0BuxhM_3_1_4_EncryptedData_2_1_EncryptedIV_2_Vx_5_Tr
          <_1_EncryptedDataInfo_2_1_EncryptedData_2_Zs3S4a7dFn4_3_1_4_EncryptedData_2_1_EncryptedIV_2_HtNE
            <_1_EncryptedDataInfo_2_1_EncryptedData_2_KqOH6AJbA6Y_3_1_4_EncryptedData_2_1_EncryptedIV_2_IbzFcy'
              <_1_EncryptedDataInfo_2_1_EncryptedData_2_S63jB2sHXgw_3_1_4_EncryptedData_2_1_EncryptedIV_2_IuPI
                <_1_EncryptedDataInfo_2_1_EncryptedData_2_A6yTSDvUBzM_3_1_4_EncryptedData_2_1_EncryptedIV_2_gJpEZuml
                  <_1_EncryptedDataInfo_2_1_EncryptedData_2_BGqbjdaset_4_cH6wTogK_5_CQ_3_3_1_4_EncryptedData_2_1_E
                    <_1_EncryptedDataInfo_2_1_EncryptedData_2_RnynRcOnluk_3_1_4_EncryptedData_2_1_EncryptedIV_2_OQ86_5_FJ
  </person1>
  <person3[id=two.worker]
    <name
      <family>Worker</family>
      <given>Two</given>
    </name>
    <email>two@foo.com</email>
    <link[manager=Big.Boss]></link>
  </person3>
  <person4[id=three.worker]
    <name
  </person4>
  </personnel>
  </?xml>

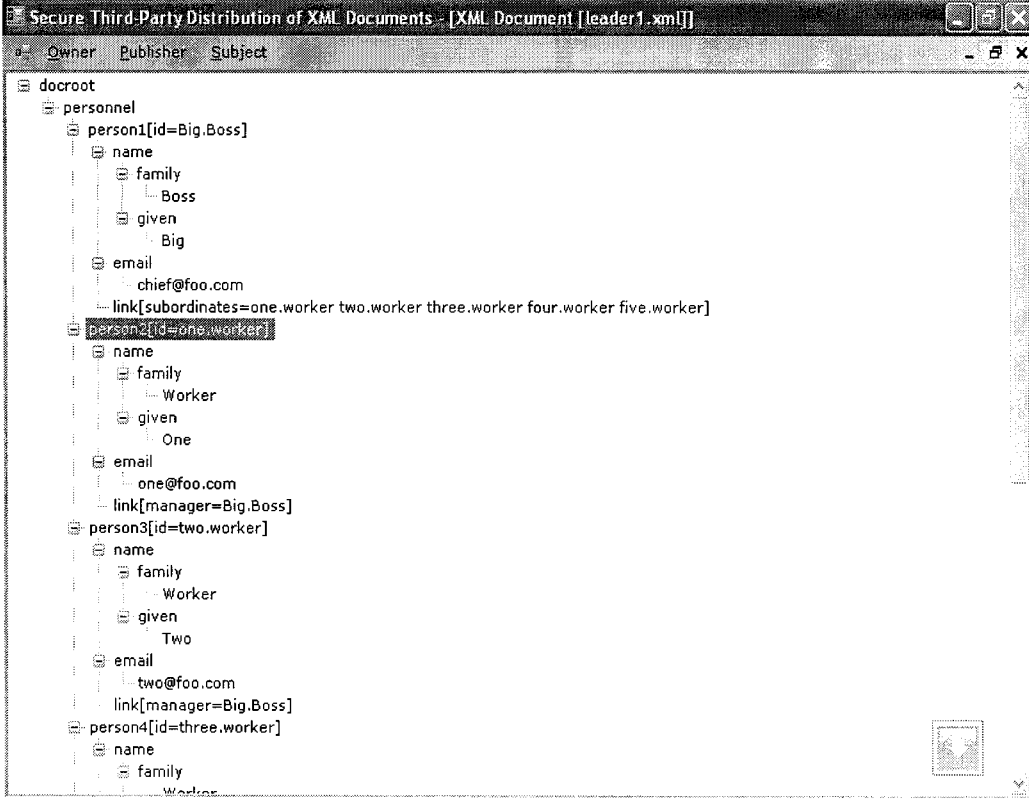
```

Secure XML Distribution

Opening Leader Document.



Leaders Documents View.



The screenshot shows a window titled "Secure Third-Party Distribution of XML Documents - [XML Document [leader1.xml]]". The window has a menu bar with "Owner", "Publisher", and "Subject". The main content area displays an XML tree structure starting with "docroot".

```
docroot
├── personnel
│   ├── person1[id=Big.Boss]
│   │   ├── name
│   │   │   ├── family
│   │   │   │   └── Boss
│   │   │   └── given
│   │   │       └── Big
│   │   └── email
│   │       └── chief@foo.com
│   │       └── link[subordinates=one.worker two.worker three.worker four.worker five.worker]
│   ├── person2[id=one.worker]
│   │   ├── name
│   │   │   ├── family
│   │   │   │   └── Worker
│   │   │   └── given
│   │   │       └── One
│   │   └── email
│   │       ├── one@foo.com
│   │       └── link[manager=Big.Boss]
│   ├── person3[id=two.worker]
│   │   ├── name
│   │   │   ├── family
│   │   │   │   └── Worker
│   │   │   └── given
│   │   │       └── Two
│   │   └── email
│   │       ├── two@foo.com
│   │       └── link[manager=Big.Boss]
│   └── person4[id=three.worker]
│       ├── name
│       └── family
│           └── Worker
```

At the bottom of the window, the text "Secure XML Distribution" is visible.

Policies Database.

localhost / localhost / Securexml / policies | phpMyAdmin 2.10.3 - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address: http://localhost/phpmyadmin/

Server: localhost Database: Securexml Table: policies

[Browse](#)
[Structure](#)
[SQL](#)
[Search](#)
[Insert](#)
[Export](#)

[Import](#)
[Operations](#)
[Empty](#)
[Drop](#)

| Field | Type | Collation | Attributes | Null | Default |
|-------------------------------------|----------|-------------------|------------|------|---------|
| <input type="checkbox"/> ID | int(11) | | | No | |
| <input type="checkbox"/> PolicyName | longtext | latin1_general_ci | | No | |
| <input type="checkbox"/> Details | longtext | latin1_general_ci | | No | |

Check All / Uncheck All *With selected*

[Print view](#)
[Propose table structure](#)

Add 1 field(s)
 At End of Table
 At Beginning of Table
 After

Indexes: 0

| Keyname | Type | Cardinality | Action | Field |
|---------|---------|-------------|--------|-------|
| PRIMARY | PRIMARY | 4 | | ID |

Done Local intranet

Subject Database.

The screenshot shows the phpMyAdmin interface for the 'Securexml' database, specifically the 'subjects' table. The table structure is as follows:

| Field | Type | Collation | Attributes | Null | Default |
|------------------------------------|-------------|-------------------|------------|------|---------|
| <input type="checkbox"/> PolicyID | int(11) | | | No | |
| <input type="checkbox"/> ID | int(11) | | | No | |
| <input type="checkbox"/> UsrString | varchar(20) | latin1_general_ci | | No | |
| <input type="checkbox"/> PwdString | longtext | latin1_general_ci | | No | |
| <input type="checkbox"/> FullName | longtext | latin1_general_ci | | No | |

Below the table structure, there are options to 'Add 1 field(s)' and a 'Go' button. The 'Indexes' section is also visible at the bottom.

REFERENCES

Books

Rama Ramachandran, Bill Evjen, Billy Hollis, Rockford “Professional VB.Net 2003”, Wrox Publications

Hemlata “**SQL SERVER 2000 REFERENCES**”, First Edition, Cyber-Tech Publications

Arthur Tat, “Web Portals : The New Gateways to Internet Information and Services”, Idea Group Publishing 2005

Xml in a nut shell by Elliotte Rusty Harold, w.Scott Means-Computers-O’reilly(2004)

Websites

http://www.frontlinedefenders.org/manual/en/eseaman/chapter2_8.html

<http://www.computerworld.com/securitytopics/security/story/0,10801,71726,00.html>

<http://www.aes.org/events/113/papers/I.cfm>