Security for the METEOR Workflow Management System

John A. Miller, Mei Fan, Shengli Wu, Ismailcem B. Arpinar,
Amit P. Sheth and Krys J. Kochut
Large Scale Distributed Information Systems Lab (LSDIS)
Department of Computer Science
The University of Georgia
Athens, GA 30602-7404
email: *jam@cs.uga.edu*
URL: *http://LSDIS.cs.uga.edu*

Abstract

The METEOR Workflow Management System emphasizes Web technology. It includes two enactment services, WebWork which is purely Web-based and ORBWork, which uses both Web and CORBA (OrbixWeb) technology. Workflow management systems utilizing Web technology are increasingly becoming a viable choice for workflows that span multiple organizations and geographical areas. Until recently, Web technology has not been terribly secure. However, by utilizing appropriate encryption algorithms, digital signatures and access control, such workflow management systems can be made secure. Since these systems include many subsystems (operating system, database system, etc.) and may involve multiple organizations, complete security solutions are enormously complex. In this paper, we sort out some of the more promising security alternatives and organize them into a security architecture suitable for workflow management systems utilizing Web technology. This security architecture has been tested using a toy worlflow enactment service and is now being encorporated in the major METEOR enactment services.

1 Introduction

As the popularity of workflow continues to grow and invade more and more application domains (healthcare, military, electronic commerce, etc.), security becomes an evermore vital concern. Since workflows can span multiple organizations and interface with all sorts of existing applications, security issues can become quite complex. As workflows utilizing Web technology may be rapidly deployed over large geographical areas and involve several organizations, they potentially represent

a major security risk.

This paper focuses on security for Workflow Management Systems (WfMSs) utilizing Web technology because of their importance and associated risk. The METEOR WfMS [Krishnakumar and Sheth, 1995, Sheth, 1996] emphasizes Web technology. It includes two enactment services: WebWork [Miller et al., 1998] which is purely Web-based and ORBWork [Kochut et al., 1999] which uses both Web and CORBA (OrbixWeb) technology.

Workflow Management Systems provide an automated framework for managing intra- and inter-enterprise business processes. According to the Workflow Management Coalition (WfMC), a Workflow Management System is a set of tools providing support for process definition, workflow enactment, and administration and monitoring of workflow processes [Hollingsworth, 1994]. Application domains where workflow technology is currently in use include healthcare, education, telecommunication, manufacturing, finance, banking and office automation. WfMSs are being used today to reengineer, streamline, automate and track organizational processes involving human and automated information systems [Joosten et al., 1994, Georgakopoulos et al., 1995, Fischer, 1995, Sheth et al., 1996b, Sheth et al., 1996a] The success of WfMSs has been driven by the need for businesses to stay technologically ahead of the ever-increasing competition in typically global markets.

Web technology is an appropriate choice for a workflow infrastructure for two distinct and important reasons. First, the ubiquitous nature of Web browsers makes them a natural user interface. Web browsers satisfy one of the primary concerns in application deployment - it allows users with any of the popular computing platforms to be able to participate in a workflow without any additional hardware. Second, Web technology provides a solid communications infrastructure for building WfMSs.

A workflow usually contains a set of automated and/or human tasks that are organized based on real world rules or business processes. A task represents an abstraction of activities that can be performed by a variety of processing entities. Tasks in METEOR are of two basic types: human tasks used for interaction with users (workflow participants), and automated tasks that do not directly involve human interaction. Automated tasks may be transactional (e.g., a database update) or non-transactional (e.g., a program making non-atomic file updates). A compound task consists of a collection of sub-tasks (thus forming a hierarchy), while a top-level task is considered to be a workflow. A human task typically has a worklist which organizes work-items as a sort of to-do list. Task scheduling, data flow, control flow, exception handling, recovery and monitoring

functions are provided by task managers, schedulers and auxiliary enactment service processes. [Sheth, 1996, Miller et al., 1996, Miller et al., 1998, Kochut et al., 1999].

There are many potential threats for Web-based workflow: Not everyone is allowed to execute certain workflow tasks. Not every one is allowed to access all information. We do not want information sent over the network to be stolen. We do not want tampered information to be stored in a database. We need to identify the security issues related to workflow and find solutions. Many groups and companies working on the workflow management system are considering security issues, including Knowledge Based Systems, Inc. They are working on the Workflow-Enhanced Role-Based Authorization Control (WERBAC) [KBSI, 1999]. JetForm's Workflow (2.0) provides data encryption capabilities for more secure workflows across networks [JetForm, 1999]. Karlapalem and Hung presented a architecture of secure CapBasED-AMS [Karlapalem and Hung, 1997] in NATO Advanced Study Institute (ASI) on workflow management systems and interoperability.

From a user's point of view, there are three principal elements to security. First, a user wishing to participate in a workflow must be identified and authenticated before being allowed to participate. In addition, the activities of an authenticated user must be tracked/audited. Second, once in the system, all communication involving the user must be secure (i.e., encrypted). Finally, access to each object (task or data object) must be checked to make sure that such usage is authorized.

We sort out some of the more promising security alternatives and organize them into a security architecture suitable for Web-based workflow. Three key aspects of security will be addressed in depth in this paper, authentication, access control and secure communication. Authentication is a prerequisite for proper access control. Secure communication makes stealing or tampering the messages sent across a network practically impossible. A reference workflow prototype was developed using Java in order to test the architecture.

This paper is organized as follows: For an appropriate background, section 2 discusses security services relevant to workflow. Section 3 considers Role-Based Access Control from a workflow point of view. A security architecture for Web-based workflows is presented in section 4. Section 5 discusses a prototype Java implementation of this architecture. Section 6 gives conclusions and explores future work.

2 Security Services

The International Standards Organization (ISO) has specified five common security services that network-based information systems should provide [Oppliger, 1998]:

- Authentication Services,
- Access Control Services,
- Data Confidentiality,
- Data Integrity and
- Non-Repudiation.

Authentication services are to provide the ability to verify that an entity is the one it claims to be. Access control services are to provide the protection of system resources against unauthorized use. Data confidentiality services are to provide the protection of data from unauthorized disclosure. Data integrity services are to provide the protection of data from unauthorized modifications. Non-repudiation services are to prevent one of the entities involved in an action from later denying participation in all or part of the action.

Authentication services are important because they are prerequisites for proper access control. A user or a process must be properly authenticated before an access control service can effectively mediate access to system resources. Data confidentiality services can protect information from being stolen, but can not provide protection against the duplication or modification of data. To prevent this, data integrity services are needed. Non-repudiation services are becoming important in the context of electronic commerce on the Internet.

2.1 Authentication Service

In most domains, it is critically important to limit access to data and tasks to authorized individuals. Before any subject (user/client) begins participating in a workflow they must be identified (who are they) and authenticated (it is really them). Unique identification in a workflow system is based on the user's digital signatures, which use public/private keys. If anyone else gets a user's private key, the system will be insecure. To minimize this risk, both physical and logical barriers should be set up to make stealing the private key difficult. Physical barriers include putting the private key on removable media (e.g., a floppy disk, CD or smart card). The user may then take the private key

with them or lock it in a secure place. Since these may be stolen, it may also be important to have logical barriers (e.g., combinations of passwords, identifying information (name, SSN, birthday), fingerprints, and/or retinal scans).

2.2 Access Control Service

Only authorized users should be allowed to execute given tasks or access given data objects. Access control must be effective in the sense that no unauthorized user should be granted access, while at the same time, no authorized user should be denied access.

In certain application areas (e.g., healthcare), security should not get in the way in a critical situation. In case of an emergency, certain authenticated users may temporarily assume greater privileges or a higher role. This is referred to a Break-Glass Procedure. Such unusual events should be recorded and the workflow administrator should be immediately informed. In addition, maximal auditing/tracking should be done until this user logs off from the higher role.

2.3 Data Confidentiality Service

For most network-based information systems, it is important that messages not be stolen, since they often contain confidential information (e.g., medical patient records). To effectively guarantee this, sufficiently strong encryption should be used, so that no one will be able to decipher data flowing over the network, except intended recipients. Symmetric key encryption algorithms are typically used to encrypt data since they are faster than public key algorithms. However, symmetric key algorithms require complex key exchange. So public key algorithms are used to send symmetric keys over a network.

2.4 Data Integrity Service

Even if stolen messages are useless because they are encrypted, a malicious user may still cause problems. If s/he intercepts a message that is just a byte stream, modifies it and then sends it along, the message will no longer be valid. To ensure the message was not tampered with, a Message Authentication Code (MAC) is calculated by applying a hash function to the message to form a digest. This digest is sent along with the message (e.g., in a digital signature). Upon receipt of the message, the message is assumed to have been tampered-with, if the same hash function applied to the received message disagrees with the digest.

2.5 Non-Repudiation Service

In a workflow setting, it is important to monitor all accesses. If a user gains access illegally or a user makes a significant mistake, administrators must be able to quickly find out what has been done and by whom. Through sufficiently effective and reliable authentication and tracking, it should be possible to undeniably establish the identity of the individual responsible for executing a given task. The system must keep track of who executed each task and when it was executed. In general, WfMSs provide auditing, tracking and monitoring capabilities for purposes of evaluating the efficiency and effectiveness of elements within a workflow (e.g., automated tasks, workflow participants (end-users) as well as the overall workflow design itself). Of course, this information is also useful for security as well. If a security or privacy violation is suspected, utilities available to workflow administrators should facilitate such investigations.

Having introduced common security services needed for workflow systems, in the next section as well as the appendix, we now examine how some of the relevant technologies and solutions that currently exist in today's computer industry can be applied [Cooper et al., 1995, Rubin et al., 1997, Pfleeger, 1997, Oaks, 1998, Knudsen, 1998].

3 Role Based Access Control for Workflow

Security administration for large networked systems is costly and error-prone. With Role Based Access Control (RBAC) [Barkley et al., 1997b] security may be managed at a level that corresponds to the organization's structure. This simplifies management of authorization while providing an opportunity for greater flexibility in specifying and enforcing enterprise-specific protection policies compared to Access Control Lists (ACLs). ACLs are a common access control mechanism used to control user access through the creation and maintenance of access control lists associated with protected objects. ACLs associate a protected object with a list of users or groups of users according to their respective modes of access to the protected object. ACLs have advantages if the users are the owners of the protected objects. Users have the right to grant other users access to the object. ACLs make it easy to know who has access to the object and under what access mode. But in the real world enterprises, end users may not be the owner of the information. The corporation may own the information. Access priorities are controlled by the organization and are often based on employee functions rather than data ownership [Barkley et al., 1997a]. Also, it is very difficult to retrieve the information about the access rights for a particular user under an ACL structure

[Barkley, 1997].

Under RBAC users do not have discretionary access to enterprise objects. Instead, access permissions are administratively associated with roles, and users are administratively made members of appropriate roles. This idea simplifies the management of authorization while providing an opportunity for flexibility in specifying and enforcing enterprise-specific protection policies. It reduces the cost of administering access control policies as well as making the process less error-prone. The process of defining roles may be based on an analysis of how an organization operates. Users can be made members of roles as determined by their responsibilities and qualifications and can be easily reassigned from one role to another without modifying the underlying access structure when the job assignment is changed. The operations that a user is permitted to perform are based on the user's role. Roles can be granted new operations as new applications and actions are encorporated and operations can be revoked from roles as needed without updating the privilege for every user on an individual basis. This simplifies the administration and management of privileges.

In workflow, RBAC may be used to determine what role(s) a subject (user/client) may perform. The role then determines which tasks a user may execute and/or which data objects s/he may access. Having established the identity of the user, we need to ensure that the user can only perform the roles s/he is entitled to. Typically, workflow designers would define roles for workflow participants, while the workflow administrator would assign users to specific roles. At design time, a task is assigned a role expression (e.g. a set of roles with optional constraints). This approach is convenient in two distinct ways: (a) All users are not known at design time so it is infeasible to authorize users at this time. (b) A role groups together a collection of privileges (e.g., which tasks can be executed) so rather than assigning each user a long tedious list of privileges (ACL), they are simply assigned one or more roles. Therefore, RBAC provides an solid foundation for security in workflow management systems.

3.1 Role Hierarchies

The capability for one role to inherit another role is a common feature of RBAC models. Roles can have overlapping responsibilities and privileges; that is, users belonging to different roles may need to perform common operations.

To make role definition and assignment even more convenient, roles may be organized hierarchically with one role implicitly including the operations, constraints, and objects that are associated with another role. In addition, role hierarchies also enhance the logical structuring of roles (e.g., if

a hospital wishes to ensure that doctors may perform tasks that nurses perform). In general, role hierarchies can be designed to reflect the responsibilities, authorities and competencies of groups within an organization.

3.2 Role Assignment Constraints

RBAC administrators can impose constraints on role design and user role assignment. There are two types of constraints: Static Separation of Duties (SSD) and Dynamic Separation of Duties (DSD) [Barkley et al., 1997b, Gavrilla and Barkley, 1998].

- Static Separation of Duties: A user is authorized for a role only if that role is not mutually exclusive with any other roles for which the user is already authorized. That is if two roles have a SSD relationship, no user may be authorized for both roles. For example, a user can not be assigned to role Teller and role Auditor in a bank since these roles are mutually exclusive according bank policy.

- Dynamic separation of duties: A pair of roles may be designed as mutually exclusive of one another with regard to role activation. That is a user may be authorized for both roles if two roles have a DSD relationship, but the user may not have both roles active at the same time. For example, a bank teller can be an account holder at the bank he is employed. But these roles can not be activated at the same time.

A properly administered RBAC provides flexibility and breadth of application which is achieved by statically and dynamically regulating users' actions through the establishment and definition of roles, role hierarchies, relationships and constraints.

3.3 Role Domains/Ontologies

Rather than defining a set of roles for every new workflow designed, or having a set of roles that apply to all designed workflows, it is desirable to take the middle ground. A group of roles is collected into an role domain. This role domain may then be associated with one or more workflows. Support for mappings between role domains could also be supported for the purpose of integrating workflows using different role domains (such domains may be referred to as ontologies). Mappings may also be used to map a workflow role ontology to a local ontology used by a subsystem (e.g., roles defined for a local Oracle database).

4 Security Architecture

This section discusses how the above security mechanisms and solutions can be effectively combined and organized into an architecture for security in METEOR WfMSs. In particular, the architecture requires message encryption, digital signatures and Role Based Access Control (RBAC).

As mentioned in the Introduction, METEOR workflow elements include tasks, task managers and schedulers. For the purposes of security three types of agents are added to these common workflow enactment service elements:

- **Security Agent**. A security agent is the final arbiter for security decisions. Security information is stored in a security database (possibly replicated) which can only be accessed by security agents. Other agents and workflow enactment elements consult with a security agent when necessary.

- **Login Agent**. A login agent is in charge of the login procedure and the role selection procedure. Users (workflow participants) must login and select an appropriate role since proper authentication is the prerequisite of proper authorization. A security agent is consulted for user identification verification and a list of permitted roles associated with the user, if properly authenticated.

- **Worklist Agent**. A worklist agent launched by a login agent is in charge of the associations of the role a user currently plays and its permitted tasks and corresponding work items. A security agent is consulted to see if the user is authorized to execute the tasks.

Schedulers and task managers work together to schedule and manage the execution of tasks. As such it is their responsibility to authorize the execution of tasks. In the case of a human task, a role object embedding the user's digital signature will be created during login and maintained by the worklist agent. When a work-item is selected from a worklist, the scheduler/task manager will check authorization based on the role object. In the case of an automated task, execution proceeds because of incoming requests on behalf of some subject (e.g., a user). A natural way to implement this is send role objects along with data objects sent to automatic tasks. In either case, the scheduler/task manager for a task examines the user's digital signature and/or role information to decide whether to permit the task to execute with the information provided by the security agent. In addition, they handle task authentication and data communication among different tasks. To prevent messages from being stolen or tampered with, communication between the

system components is encrypted. The way in which these elements interact is shown in Figure 1, and discussed further in the subsections below.

Figure 1: Security Architecture

From a more workflow centric point of view, let us consider a fragment of a workflow consisting of two human tasks feeding an automated task (with the inputs ANDed together). This situation is depicted in Figure 2.

For each human task, the check is made assuming the requesting role object is r_i where r_i is the role created when the user logins in under a particular role type (e.g., Doctor). For the automated task, each role object (r_1 and r_2) is checked because of the ANDing of both inputs (ORing would allow either role object to pass the check). The details of how this check is made are presented in the next subsection.

Figure 2: Security in a Workflow Fragment

4.1 Modeling RBAC for Workflow

In this subsection, role-based access control for workflow is examined in terms of the entities associated with roles and the relationships between roles and the other entities.

4.1.1 Entities Associated with Roles

There are several types of entities in workflow relevant to the enforcement of role based access control.

- **User**. A participant in a workflow.
- **Role**. A participant may play one or more roles at any particular time.
- **RoleDomain**. Related roles are organized into a role domain and may be used in one or more workflows.
- **Task**. A task may only be executed by participants logged in under authorized roles.

- **DataObject**. There are two main types of data objects in workflow: workflow level data objects and application level data objects (for a more refined categorization see [Wu, 1999]). Access to a data object is controlled on the basis of which task under what role is requesting access.

4.1.2 Role Relationships

A central issue in applying RBAC to workflow is how to relate roles with the other entities discussed. There is a need for a thread of security to be weaved from a user to a role to a task to a data object. One could choose to relate these entities through an arity four relationship. However, to be true to the spirit of RBAC, User and Role should be connected through a separate binary many-to-many relationship. This leaves the option of associating Role, Task and DataObject with a ternary relationship.

Using a ternary relationship is the most general and flexible approach, but it may be complex. Even if a ternary relationship is used, it is still be useful to have a binary relationship between roles and tasks in order to determine if the participant logged in under a given role may execute the task. In addition, without sacrificing too much flexibility, one may choose to decompose the ternary in multiple binary many-to-many relationships. The four possibilities are listed below:

- One Ternary Relationship. Decisions on Task execution and DataObject access may be arbitrarily linked with Roles.
- One Ternary Relationship and one Binary Relationship. Execution of Task is dependent on which Role. Access to DataObject is dependent on which Task and which Role.
- Two Binary Relationships. In this case, there are two binary relationships: one associating Role with Task and the other associating Task with DataObject.
- Three Binary Relationships. In this case, there are three binary relationships: one associating Role with Task, another associating Task with DataObject, and the last associating Role with DataObject.

At this point, we have chosen the most general approach, the second possibility listed. In total, the security thread we weave is composed of three principal relationships. We specify these precisely using functional specifications. (a Unified Modeling Language (UML) data model of a restricted version is given later in the paper).

- **User to Role Mapping**. When a user needs to participate in workflows, s/he is assigned one or more roles.

  $$f_u(User, Role)$$

  This functions returns true iff User has been assigned to perform Role.

  When a user successfully logins, a role object is created representing the role being played by this user. The role object is sent along with request made by the user so that this information can be used in making access control decisions.

- **Task to Role Mapping**. When a task is designed, its is assigned a role expression and optionally a constraint.

  $$f_t(Task, Role)$$

  We specify this by associating roles with the task. For a given task, role r is authorized to run the task iff

  $$r \subseteq f_e() \text{ and } f_c(r)$$

  where f_e is a role expression and f_c is a role constraint. A role expression is formed using set operators (union (+), intersection (*), difference (-) and symmetric difference (/). A role constraint is a predicate (comparison operators combined with Boolean operators) on role attributes. Role expressions and constraints are fundamentally different in that expressions are type expressions, while constraints operate on role objects. One example, is a constraint indicating that a Nurse performing a given task must have three years of experience.

  $$\text{Nurse.experience} >= 3$$

  These constraints introduce more complexity and possible conflicts. Suppose a doctor with two years of experience wishes to execute the task, is s/he permitted? This is an example of how role constraints may be incompatible with role hierarchies. Some conflicts can be resolved by letting an organization set a policy, e.g., hierarchy overrides constraints (see [Arpinar, 1999] for details).

- **Task to DataObject to Role Mapping**. Such a mapping will encode the ternary relationship. Specification of this part of access control is especially difficult for the following reasons: (1) The number of possible accessible data objects may be too large to deal with any systematic, yet convenient way. (2) Some data objects, for example in a legacy systems may be accessed invisibly to the workflow system. (3) The METEOR workflow model emphasizes

abstraction in sense that the details of what a task does need not (and one may argue can not) be known to task managers and scheduler. If these workflow level processes do not know which application level data objects the task they control will access, how can they enforce access control? Indeed, the specific data objects accessed may only be determined after the task has begun to run. For these reasons, access control to data objects is made the responsibility of the task themselves and not the workflow system. Based on the invoking role objects, input data objects and internal logic, the task must decide whether access is permitted or defer the decision further to a system resource (e.g., a database system). This suggests that in the worst-case the only solution is custom coding in the application task. However, in many cases automation may be helpful as discussed in the next subsection.

4.2 Control of Access to Data Objects

Besides information stored by a WfMS, workflow applications themselves may access sensitive information stored in files and databases. Systems should ensure that sensitive information is not accidentally made available. The following security measures can be applied to help assure this.

Care should be taken by application developers, task implementors, system administrators and database administrators to secure data in these component systems (e.g., database systems typically now provide role-based security).

Security in workflows can be simplified if responsibilities can be divided between the workflow developer/administrator and application task implementors. Authorization to execute a task is the concern of the workflow developer/administrator and is enforced by schedulers/task managers; whereas, access to data objects is the concern of the task implementor and is enforced by the task. (For an alternative approach in which control of access to data objects is managed at the workflow level see [Wu, 1999].)

Let us now consider how access control can be done for several different types of data objects:

- **Accessing Databases**. Authorization to access database objects from a task is handled by the task implementor in conjunction with the database administrator. In particular, to access an Oracle database, a username and password must be passed as parameters to the connect function. These strings could be hardcoded in the task implementation. If they are hardcoded, source code should not be made available. Also, in order to make it more difficult for hackers to steal database passwords, executable files should not readable. As an

alternative to hardcoding passwords in tasks, they may be stored with role objects.

- **Accessing Persistent Objects**. Typically, distributed object management systems provide (or will provide) Persistent Object Services as well as Security Services. Access to a persistent CORBA object may be controlled using CORBA Security Services [OMG, 1998, Vogel and Duddy, 1998]. For example, a target CORBA object can check the invoking client/user's credential (analogous to our role object) when the client binds to the target object.

- **Accessing Workflow Data Objects**. METEOR workflow enactment services transfer data objects from task to task (e.g., transient objects or references to persistent objects may be sent between tasks). Since these objects are at the workflow level, role expressions may be assigned to them. As a consistency check, task receiving data objects be able to access all of them. In other words, the task's role expression must subsume the union of all the incoming data objects' role expressions.

- **Accessing Files**. Typically workflow processes are run under a special account (e.g., an account called orbwork). It may be the case that a file is accessible to this account (e.g., a UNIX file with mode = 644 is generally readable). In other situations, an orbwork process may communicate with another process running under a different account. Although a potential security risk, it is also possible for task to switch to another user account.

To minimize the amount of handcoding involved in task level access control to data objects, helper classes are being developed for access to common resources such as Oracle DBMSs.

4.2.1 Example of Access Control

As an example, suppose there is a task called `GetPatientRecords` which can be executed by ordinary doctors (Doctor role) or doctors that are heads of departments (LeadDoctor role). A hospital may have a policy that *Doctors can only browse database records of patients they have treated, but LeadDoctors can browse all patient records.* Both Doctors and LeadDoctors can execute `GetPatientRecords`. In the case of LeadDoctors, all patient records are available, however, under the Doctor role only a subset of the records should be available. LeadDoctors may execute `GetPatientRecords` and this task has access to patient records (data objects). The situation is not so simple for the Doctor role. Again Doctors may execute `GetPatientRecords` and this task

has access to patient records (data objects). The questions is, how are the additional restrictions expressed and enforced. Embedded in a LeadDoctor role object is a database username and password permitting access to all records; whereas, the username and password embedded in a Doctor role object will only permit access to his/her patient records.

5 Prototype Implementation

A prototype of this architecture has been implemented to refine the architecture before porting it to the main METEOR enactment services (ORBWork or WebWork). The prototype is a fully distributed system, implemented using Java Remote Method Invocation (RMI). Hence, it provides scalability to the system. In addition, the Java Naming and Directory Interface (JNDI) was used to provide location-independent access to remote objects. Furthermore, the prototype relies on the foundation security APIs supplied with Java Platform 2 (JDK 1.2) and provides its own general purpose packages for security. This latest release of Java contains a subset of cryptography functionality, including APIs for digital signatures, message digests and tools for keystore creation and management. It also provides services for algorithm parameter management, algorithm parameter generation, key factory and certificate factory. Those trusted applets in downloaded JAR files signed by a trusted entity are granted more permissions as signed applets [Oaks, 1998]. In our case, our Login Agent and Worklist Agent together is a signed applet which is granted read permission in order to load user's private key. They are also granted network connection permissions since it need to talk to other system components such as Schedulers/Task Managers. APIs for data encryption and decryption, key exchange and message authentication code (MAC) are released separately in a Java Cryptography Extension (JCE) package [Gong et al., 1997].

5.1 UML Diagram for Security Database

In the prototype, an object-relational database system (Oracle 8i) is used to store information required by the security mechanisms used. This security database must support user authentication, data encryption and access control (RBAC). In the prototype role constraints are not provided because of there inherent complexity (e.g., they can make it impossible to statically analyze the termination correctness condition for a workflow). The data model for the security database includes most of role relationships previously discussed. (For a more complete data model see [Wu, 1999] and for related workflow meta-data see [Arpinar, 1999]).

The entities, their attributes and relationships are shown in Figure 3 shows the UML data model for the security database. Corresponding to the entity types mentioned in the previous section are the following object types: Role, RoleDomain, User, Task and Resource. (Resource is used as a coarse-grain alternative to DataObject.)

Figure 3: Security Database Design

The following associations (relationships) are supported:

- Subsumes. Each role hierarchy forms a directed acyclic graph (DAG) whose nodes correspond to roles in a single role domain.
- Collects. Based on organization or functional requirements, roles are organized into role domains. Since each role must be assigned to a single domain, the role domain name and role name together can be used to distinguish each role.
- Plays. Workflow administrators assign users roles to play. Login involves chosing a role to play and results in the creation of a role object which servers as a token for that user under the chosen role.
- Authorizes. The principle of mapping between roles and tasks is least privilege. According to role's job function, each role is assigned the minimum set of tasks this role can perform.

- Accesses. As discussed in the previous section, control of access to data object is complicated. In this prototype, the security database contains information that can be used by tasks to access resources such as databases and files. Resource contains account and password information that can be used for example to log into an Oracle database. This information will be embedded in the role object created when the user logs in.

5.2 Security Agent

The Login Agent, Worklist Agent and Schedulers/Task Managers consult a Security Agent in order to make security decisions. Only the Security Agent can access the security database. For performance reasons, multiple Security Agents may be utilized. However, if multiple Security Agents are used, we must ensure the consistency of the security database. A fundamental use of a Security Agent is to determine whether a user may login under a role and execute a task. The Login Agent will ensure that the user has been authenticated and has been assigned the role. The task's Scheduler/Task Manager will ensure that the role is authorized to execute the task.

- Authenticate User. The Security Agent loads the user's public key from the database and then verifies the user's digital signature attached at the end of the message.

```
PublicKey pubKey = getPubKey (username);
Signature dsa    = Signature.getInstance ("SHA/DSA");
dsa.initVerify (pubKey);
dsa.update (message);
verifies = dsa.verify (userSignature);
```

- User to Role Mapping. One user may be authorized to play a set of roles. But these roles may subsume other roles. When the user browses all the roles he can play, a Security Agent will look up the subsume table and return all roles.

- Role to Task Mapping. If one role subsumes the other roles, this role can do what all the other roles can do. Each role has a list of tasks it can execute. The tasks this role can execute are a union of all the tasks the other roles can execute as illustrated below. The result set SumTaskList is a union of each TaskList for each roleName.

```
// get all the roles this user's current role contains
```

```
Vector roles = getRoles (userName, rootRoleName);
// get tasks for each role, then union
Enumeration rlEnum = roles.elements ();
while (rlEnum.hasMoreElements ()) {
    roleName = (String) rlEnum.nextElement();
    String qry = "Select * from RoleTask WHERE rolename = " + roleName + "'";
    DBQuery dbq = new DBQuery (qry);
    TaskList = dbq.getColumn ("taskname");
    union (SumTaskList, TaskList);
}; // while
```

Another responsibility of the Security Agent is key management. Without appropriate key management to generate, disseminate and destroy keys, security will become either unwieldy or ineffective. Each organization (or part of an organization) participating in workflows will need to disseminate keys, so ideally it should also generate and destroy keys. The workflow administrator may use a utility provided by the Security Agent to produce unique key pairs for users. Some organizations may wish to get their keys from recognized Certificate Authorities (e.g., VeriSign) or from a trusted affiliated organization. Tools such as the Netscape Certificate Server allow organizations to conveniently generate keys. This piece of code below is used to generate key pairs. Private keys are store in a floppy disk while public keys are stored in the security database.

```
KeyPairGenerator keyGen = KeyPairGenerator.getInstance ("DSA");
keyGen.initialize (1024, new SecureRandom ());
keypair = keyGen.generateKeyPair ();
PersistentKeys.savePrivateKeyToFile (keypair.getPrivate (), Path);
PersistentKeys.savePublicKeyToDB (keypair.getPublic (), userName);
```

The user's public key object is stored in the database. It has four member variables: p, q, g and y (p, q and g are algorithm parameters and y is the user's public key). We implement a persistent key class that can convert a public key object to a byte stream then convert the byte stream to a string which can be stored in the database. We use Java UCEncoding to convert a byte stream to string that only contains 64 characters:'0'-9', A'-Z', a'-z', (and)'. Then the string can be used as part of JDBC queries sent to a JDBC database without any trouble, since special characters may

be restricted from the query language. UCEncoding use 3 bytes instead of 2 bytes to encode the message, so the length of message will increase.

```
public static String stringify (PublicKey publicKey) {
    try {
        /* write the key object to a ByteArrayOutputStream object */
        ByteArrayOutputStream bos = new ByteArrayOutputStream ();
        ObjectOutputStream oos = new ObjectOutputStream (bos);
        oos.writeObject (publicKey);
        oos.flush ();
        UCEncoder ucencoder = new UCEncoder ();
        return ucencoder.encodeBuffer (bos.toByteArray());
    } catch (Exception e) {
        System.out.println (e);
        return null;
    }; // try
}; // stringify
```

Since the parameters of the public key object are the same for all the users, in the future, we could only store user's public key y, which is much smaller than the public key object into the database. When the user's public key (object) is needed, we should restore it using parameters and user's public key y. Thus we can save a substantial amount of space.

5.3 Login Agent

To participate in a workflow, a user carries out the following procedure. From a Web browser, the user enters the URL for the Login Agent. It is important that the Login Agent be authenticated using its own digital signature so it can access user's private key. The Login Agent is implemented as a Signed applet. We also need to grant read and network permission to the Login Agent. The login script displays three buttons: Login, Logout and Quit. A user types in his/her user name and role domain name, role name s/he is trying to perform from login script (role domain name and role name are optional, since the user can browse all the roles s/he can perform and chose one). The Login Agent will generate a digital signature from the user's private key loaded from his/her smartcard. The message will be encrypted by Security Agent's public key and sent to

Security Agent. The Security Agent decrypts the message and loads user's public key form security database then verifies the signature. Once the user has been identified, the Login Agent will display a list of roles the user is authorized to perform. The user can choose a role that has been assigned or the roles he can perform from the role hierarchies. The login procedure is shown in Figure 4. The user may logout from workflow procedure s/he is working on by clicking the Logout button. But the applet itself is still there, so the user may login again when s/he wants to play another role. When Quit button is clicked, the memory image of the applet is cleared before exiting. The applet itself does not store anything persistently in files, databases, etc.

Figure 4: Authentication Procedure

5.4 Worklist Agent

Once a user has selected a role, Login Agent will launch Worklist Agent to display the list of user tasks this role is authorized to perform by consulting Security Agent shown in Figure 4. In Figure 5, T1 T2 and T3 are the tasks the user can execute shown in a browser. The user wants to execute T1 by clicking the button T1. The Worklist manager will display a list of work items the user can work on if the user is authorized, otherwise, a warning message will appear.

Figure 5: Handle Work Items

5.5 Scheduler/Task Manager

After the work items displayed on the screen, the user can begin to work. The Scheduler/Task Manager is invoked when the user clicks the work item s/he wants to work on. When invoking a Scheduler/Task Manager, some security control information is sent along with the application data both encoded as name-value pairs.

```
Task_ID=&lt;instance-id&gt; &amp; Task_Name=&lt;task-name&gt; &amp;...&lt;data&gt;
User_Name=&lt;user-name&gt; &amp; Digital_Signature=&lt;user-signature&gt; &amp; Role_Name=&lt;role&
```

This allows the Scheduler/Task Manager to uniquely identify the sender and make sure s/he is authorized to execute the task. Figure 6 illustrates a Scheduler/Task Manager for a human-computer task.

The user wants to work on the item1. The Scheduler/Task Manager for the task T1 is invoked and some security control information is sent to him. Upon receiving a message, the Scheduler/Task Manager must make sure that message was not forged or tampered with. This is accomplished using the message authentication code (digital signature, etc.) in the message. The Scheduler/Task

Figure 6: Scheduler/Task Manager

Manager must now consult the Security Agent to see if the requesting user has authorization by checking the user's digital signature. Since the Scheduler/Task Manager and Security Agent may not be on a same host and Scheduler/Task Manager is not allowed to query the database, all the communication are done by remote method invocations. In order to improve the performance, the Scheduler/Task Manager maintains a cache of user information (including user name, user public key, user roles and tasks associated with those roles). Cached user information does not need network communication but does require that the cache be kept consistent with the up-to-date user information in the security database. The Scheduler/Task Manager will look up its cache first. If it could not verify the user using information in the cache, it will consult the Security Agent. The Security Agent will do authentication and authorization and also return user information, so the Scheduler/Task Manager may cache the information for later use. If the user is authorized, the form and data for this item will be displayed, otherwise a warning message will be displayed. The checking consists of three steps:

- Identify the User. User's digital signature is verified by using his/her public key.
- User to Role Mapping. Scheduler/Task Manager must make sure that the user is indeed permitted to perform the indicated role.
- Role to Task Mapping. The Scheduler/Task Manager must check the role name to establish that this task may be executed by this role.

The form for human-computer task displayed can come from two sources either an html file or hardcoded in workflow application program, which is generated from code generator. For non-repudiation, we must know which nurse administered the medications, not just that a nurse administered them. From the user's digital signature, we know who executed this task and we also know what role the user was playing when he executed this task. This information should store in security database or workflow repository system for later use.

5.6 Secured Communication Channel

All the components of this system can be fully distributed. Remote method invocation is used for communication between different components. All the communication is encrypted. We were trying to use third party's SSL implementation, but RSA public key algorithm is not free, so we implement a simplified SSL using the ELGamal algorithm to demonstrate how the secure communication channels work. The ELGamal system [ELGamal, 1985] is a public-key cryptosystem based on the discrete logarithm problem. Detailed information about ELGamal algorithm and our implementation can be found in [Fan, 1999]. Analysis based on the best available algorithms for both factoring and discrete logarithm shows that RSA and ELGamal have similar security for equivalent key lengths. The main disadvantage of ELGamal is the need for randomness, and its slower speed (especially for signing). Another potential disadvantage of the ELGamal system is that message expansion by a factor of two takes place during encryption. However, disadvantages are negligible if the cryptosystem is used only for exchange of secret keys such as in SSL. We discuss this in the next section.

5.7 Encryption of Communication

In this section, the implementation of our simplified SSL is introduced. As depicted in Figure 7, the client generates a symmetric key first, then encrypts it with server's ELGamal public key and sents it to the server.

Note that in the figure E stands for encryption algorithm and D stands for decryption algorithm. Only intended server can decrypt message since nobody else has the corresponding ELGamal private key. The server recovers the symmetric key and using it to encrypt data required by the client. In this way, the application data is secured. Since only the designated server can recover the symmetric key using his private key, the server has effectively authenticated itself to the client.

Figure 7: Simplified SSL

5.8 Authentication between Different Components

The authentication of different components is done by SSL protocol. As discussed in last section, the server can authenticate itself to the client since nobody else has the private key corresponding to the public key used for encryption. In this section, we discuss client authentication, such as authentication between Login Agent and Security Agent, authentication between Worklist manager and Security Agent. In these cases, the Security Agent acts as SSL servers, while Login Agent and Worklist manager act as SSL clients. As mentioned before, one of the Security Agent's duties is key management. Similar to generating key pairs for a user, it can also generate key pairs and certificate for Schedulers/Task Managers. When a Worklist Agent communicates with a Scheduler/Task Manager, it sends its certificate and digital signature to the Scheduler/Task Manager. Upon receiving the message, the Scheduler/Task Manager extracts the Worklist manager's public key from his certificate and verifies the signature. Through this procedure, the Security Agent (sever) can authenticate the Worklist manager (client).

6 Conclusions and Future Work

We have designed a security system for Web-based workflow, which provides the following capabilities:

- User authentication and software components authentication: to verify the entity is the one it claimed to be.
- Role based access control: user can only execute the tasks his/her current role has authorized

for.

- Data confidentiality: all communication is encrypted.
- Data integrity: to detect if messages are tampered with.
- Non-repudiation: to undeniably establish/record identity. A workflow administrator needs to monitor and keep track of all accesses. He may want to know who executed a given task instead of which role executed the task (which nurse administered the medications, not just that a nurse administered them). From the user's digital signature, we know who did this task and we also know what role the user is playing when he executes this task. Since all task executions are logged, this information will be available.

For examples of actual workflows using this security architecture see [Fan, 1999, Wu, 1999].

At present, we are working on the following improvements and extensions to the architecture as well as the prototype. Porting to the ORbWork workflow enactment service is also underway.

- Interaction with Worklist Management. The responsibilities of a Worklist agent is to assign user a list of work items based on a specific assignment policy [Arpinar, 1999]. For example, a user is assigned a work-item according his qualifications, availability, current workload and other business rules. Exactly the best way to provide this functionality at run time is still an issue.
- Efficient Task Authorization. Rather than having schedulers/tasks managers consult a security agent for authorization decisions, we are exploring the option of including task authorization tickets in the role object.
- Multilevel Security. This security architecture will also be ported to an MLS version of ORBWork being developed with the Naval Research Laboratory [Froscher and Kang, 1997, Kang et al., 1999]. This version will provide MultiLevel Security (MLS) via networks partitioned into security domains (e.g., top secret, secret and unclassified).

Acknowledgements

This research was partially supported by grants from the National Institute of Standards and Technology (NIST) (under the HIIT ATP contract, number 70NANB5H1011), and the Naval Research Laboratory (NRL) (US Department of Navy, number N00173-98-2-L005).

References

[Arpinar, 1999] Arpinar, I. (1999). A Proposal for Role Domain Design and Specification of Work-Item Assignment Strategies in METEOR. Technical report, UGA-LSDIS. URL: http://LSDIS.cs.uga.edu/activities/NSF-workflow.

[Barkley, 1997] Barkley, J. (1997). Comparing Simple Role Based Access Control Models and Access Control Lists. Technical report, NIST. URL: http://hissa.ncsl.nist.gov/rbac.

[Barkley et al., 1997a] Barkley, J., Cincotta, A., Ferraiolo, D., Gavrilla, S., , and Kuhn, D. (1997a). Role Based Access Control for the World Wide Web. Technical report, NIST. URL: http://hissa.ncsl.nist.gov/rbac.

[Barkley et al., 1997b] Barkley, J., Cincotta, A., Ferraiolo, D., Gavrilla, S., and Kuhn, D. (1997b). A Role Based Access Control Model and Reference Implementation within a Corporate Intranet. Technical report, NIST. URL: http://hissa.ncsl.nist.gov/rbac.

[Cooper et al., 1995] Cooper, F. et al. (1995). *Implementing Internet Security*. New Riders Publishing.

[Dogac, 1996] Dogac, A. (1996). Special-Theme Issue: Multidatabases. In *Journal of Database Management*. Idea Group Publishing, Harrisburg, PA.

[ELGamal, 1985] ELGamal, T. (1985). A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE ???*

[Fan, 1999] Fan, M. (1999). "Security for the METEOR Workflow Management System. Master's thesis, University of Georgia.

[Fischer, 1995] Fischer, L. (1995). *The Workflow Paradigm - The Impact of Information Technology on Business Process Reengineering*. Future Strategies, Inc., Alameda, CA, 2nd. edition.

[Freier et al., 1996] Freier, A., Karlton, P., and Kocher, P. (1996). SSL 3.0 Specification. Technical report, Netscape Communications Corporation. URL: http://home.netscape.com/eng/ssl3/index.html.

[Froscher and Kang, 1997] Froscher, J. and Kang, M. (1997). MLS Workflow on a MLS Distributed Architecture. Technical report, Naval Research Laboratory. URL: http://www.hokie.bs1.prc.com/ia/archnrl.htm.

[Garfinkel and Spafford, 1997] Garfinkel, S. and Spafford, G. (1997). *Web Security & Commerce*. O'Reilly & Associates, Cambridge, MA.

[Gavrilla and Barkley, 1998] Gavrilla, S. and Barkley, J. (1998). Formal Specification for Role Based Access Control User/Role and Role/Role Relationship Management. Technical report, NIST.

[Georgakopoulos et al., 1995] Georgakopoulos, D., Hornick, M., and Sheth, A. (1995). An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, 3(2):119–154.

[Gong et al., 1997] Gong, L., Mueller, M., Prafullchandra, H., and Schemers, R. (1997). Going Beyond the Sandbox: An Overview of the New Security Architecture in the JavaTM Development Kit 1.2. In *Proc. of USENIX Symposium on Internet Technologies and Systems*, Monterey, CA.

[Hollingsworth, 1994] Hollingsworth, D. (1994). The Workflow Reference Model. Technical Report TC00-1003, Issue 1.1, The Workflow Management Coalition, Brussels, Belgium.

[JetForm, 1999] JetForm (1999). JetForm. Technical report, JetForm. http://www.jetform.com/pressroom/1997/prir970414-2.html.

[Joosten et al., 1994] Joosten, S., Aussems, G., Duitshof, M., Huffmeijer, R., and Mulder, E. (1994). *WA-12: An Empirical Study about the Practice of Workflow Management*. University of Twente, Enschede, The Netherlands. Research Monograph.

[Kang et al., 1999] Kang, M., Froscher, J., Sheth, A., Kochut, K., and Miller, J. (1999). A Multilevel Secure Workflow Management System. In *Proc. of 11th Conference on Advanced Information Systems Engineering (CAiSE'99)*, Heidelberg, Germany.

[Karlapalem and Hung, 1997] Karlapalem, K. and Hung, P. (1997). Security Enforcement in Activity Management Systems. In *NATO Advanced Study Institute (ASI) on Workflow Management Systems and Interoperability*.

[KBSI, 1999] KBSI (1999). KBSI. Technical report, KBSI. http://www.kbsi.com/Research/projects/workflow.htm.

[Knudsen, 1998] Knudsen, J. (1998). *Java Cryptography*. O'Reilly & Associates, Cambridge, MA.

[Kochut et al., 1999] Kochut, K. J., Sheth, A. P., and Miller, J. A. (1999). Optimizing Workflow. *Component Strategies*, 1(9):45–57.

[Krishnakumar and Sheth, 1995] Krishnakumar, N. and Sheth, A. (1995). Managing Heterogeneous Multi-system Tasks to Support Enterprise-wide Operations. *Distributed and Parallel Databases*, 3(2):155–186.

[Miller et al., 1998] Miller, J., Palaniswami, D., Sheth, A., Kochut, K., and Singh, H. (1998). WebWork: METEOR2's Web-Based Workflow Management System. *Journal of Intelligent Information Systems, Special Issue on Workflow Management Systems*, 10(2):185–215.

[Miller et al., 1996] Miller, J. A., Sheth, A. P., Kochut, K. J., and Wang, X. (1996). CORBA-based Run-Time Architectures for Workflow Management Systems. *[Dogac, 1996]*, 7(1):16–27.

[NIST, 1994] NIST (1994). Digital Signature Standard. Technical report, NIST. URL: http://www.itl.nist.gov/div897/pubs/fip186.htm.

[Oaks, 1998] Oaks, S. (1998). *Java Security*. O'Reilly & Associates, Cambridge, MA.

[OMG, 1998] OMG (1998). Security Service Specification. Technical report, OMG. URL: http://www.omg.org/pub/docs/formal/98-12-17.ps.

[Oppliger, 1998] Oppliger, R. (1998). *Internet & Intranet Security*. Artech House.

[Pfleeger, 1997] Pfleeger, C. (1997). *Security in Computing, Second Edition*. Prentice Hall, Upper Saddle River, NJ.

[Rubin et al., 1997] Rubin, A., Geer, D., and Ranum, M. (1997). *Web Security Sourcebook*. John Wiley & Sons.

[Sheth, 1996] Sheth, A. (1996). Proc. of the NSF workshop on workflow and process automation in information systems. University of Georgia. URL: http://LSDIS.cs.uga.edu/activities/NSF-workflow.

[Sheth et al., 1996a] Sheth, A., Georgakopoulos, D., Joosten, S., Rusinkiewicz, M., Scacchi, W., Wileden, J., and Wolf, A. (1996a). Report from the NSF Workshop on Workflow and Process Automation in Information Systems. Technical report, University of Georgia, UGA-CS-TR-96-003. URL: http:// LSDIS.cs.uga.edu/activities/NSF-workflow.

[Sheth et al., 1996b] Sheth, A., Kochut, K. J., Miller, J., Worah, D., Das, S., Lin, C., Palaniswami, D., Lynch, J., and Shevchenko, I. (1996b). Supporting State-Wide Immunization Tracking using Multi-Paradigm Workflow Technology. In *Proc. of the 22nd. Intnl. Conference on Very Large Data Bases*, Bombay, India.

[Vogel and Duddy, 1998] Vogel, A. and Duddy, K. (1998). *Java Programming with CORBA*. John Wiley and Sons, New York, NY.

[Wu, 1999] Wu, S. (1999). Task and Role Combined Access Control Model for Workflow Applications. Technical report, UGA-LSDIS. URL: http://orion.cs.uga.edu:5080/ wsl.

Appendix: Applicable Cryptography

Cryptography is a collection of techniques used to protect information sent over communication channels. It mainly includes encryption and decryption techniques. Encryption is the transformation of data into some unreadable form. Its purpose is to ensure privacy by keeping the information hidden from anyone for whom it is not intended. Decryption is the reverse of encryption. It is the transformation of encrypted data back into some intelligible form.

There are two kinds of approaches: symmetric key algorithms and public key algorithms (asymmetric). When an algorithm uses the same key for both encryption and decryption, it is a symmetric key algorithm. When a pair of key is used, it is called a public key algorithm (asymmetric). One key is used for encryption and only the corresponding key can be used for decryption.

Symmetric key algorithms are used to encrypt bulk data since they are faster than the public key algorithms [Garfinkel and Spafford, 1997]. However since two parties must have the same key, one must make sure to send the key or a secret message which can used to generate the symmetric keys later to the other party in a secure way after the key is generated. Typical symmetric key algorithms are DES, 3-DES, IDES, Blowfish, RC2, RC4 and RC5 [Garfinkel and Spafford, 1997].

Public Key Encryption involves the use of a "public/private key pair". In the Figure 8, E stands for the encryption algorithm and D stands for the decryption algorithm. Data is encrypted using B's public key and sent to B over the network. Upon receiving the encrypted data, B decrypts it using his/her private key to get the original data.

When the public key is used for encryption and the private key is used for decryption, the message is only meaningful to the designated recipient since only the entity having the private key will be able to decrypt the message. Sometimes, the private key is used for encryption, but under

Figure 8: Public Key Algorithm

this scenario, we care more about who is the message sender instead of securing the message since anyone may have the sender's public key to decrypt the message. Digital signatures use public key algorithms under this scenario as shown below in Figure 9 (S stands for signing algorithm, and V stands for verifying the signature). Another difference between digital signatures and public key encryption is that the message sent over the network under this scenario is not encrypted. We will discuss digital signatures in more detail later.

Figure 9: Digital Signature Algorithm

A significant advantage of public key algorithms is that private keys never need to be sent out over a network. Unfortunately, these algorithms are also more computationally expensive. Typical public key algorithms are Diffie-Hellman, RSA, ELGamal, and DSS [Garfinkel and Spafford, 1997].

6.1 Digital Signature

A digital signature is a replace of authentication that enables the receiver to verify the sender is whom he claims to be [NIST, 1994]. Like handwriting signatures, digital signatures satisfy several properties [Oppliger, 1998]. It is unforgeable - only the sender who has the private key can attach his signature to a document, verifiable - a receiver is able to verify it, non-deniable - the sender can not be able to repudiate it later. Unlike handwriting signatures, digital signatures incorporate the data that are signed. A digital signature is not constant but a function of the data. Thus it is unalterable - if the document is changed after it has been signed the signature is not valid anymore and it is non-reusable - one can not cut a signature off one document and attach it to another.

6.2 Certificate

A certificate is a digitally signed statement from a certificate issuer, saying that the public key of some other entity (the subject) has some particular value. A certificate is like a driver's license. It binds an entity's public key to his name.

A typical certificate file includes the subject's name, his public key, certificate issuer, the issuer's digital signature, an expiration date and a serial number. If you trust the issuer, you trust that the association in the certificate between the specified public key and the subject is authentic.

Another term related to certificates is Certification Authority (CA). A CA is an entity (e.g., a business) that is trusted to sign (issue) certificates for other people (entities). There are many such Certification Authorities, such as VeriSign, GTE and Netscape.

6.3 Security Socket Layer (SSL)

SSL (3.0) is a program layer created by Netscape for managing the security of message transmissions in a network. It provides four security services: authentication, non-repudiation, data integrity and data confidentiality [Freier et al., 1996, Garfinkel and Spafford, 1997].

Authentication and non-repudiation of server and client is provided using public-key certificate exchange and digital signature. Data integrity is provided using message authentication code (MAC). Data confidentiality is provided using keys and encryption algorithm negotiated during SSL handshake.

The SSL handshake protocol consists of two parts, server authentication and client authentication. The second part is optional, but for our workflow project, we need both of them.

According to [Freier et al., 1996], when setting up a server, the administrator creates a key pair and gets a digitally signed certificate from a trusted Certification Authority.

After a client connects to a server, it sends a list of supported cryptographic algorithms. The server chooses one of them and sends back its selection together with the server's certificate. The client can then retrieve the server's public key from its certificate.

The client requests the server to prove its identity by sending a message encrypted using the server's public key. The server deciphers it using its private key and sends a finish message to the client for his verification. Since only the server holds his private key, nobody but the server can decrypt the encrypted message and generate the expected finish message, the client is assured that the server is who it claimed to be.

Both the client and the server use the same hash function on the message the client first sent and convert it into a secret that is only known to the two sides. The secret is used to generate session keys for encryption and MAC computations.

Similarly, a client can authenticate itself to the server. It sends its certificate and a verify message which is its digital signature in response to the server's request, so the server can verify its digital signature.

The handshake procedure is not encrypted. The finish message is the first protected with the just-negotiated algorithm, key and secrets.

After the handshake finishes, communication is encrypted with the selected algorithms and keys illustrated below. The bracket means "encrypted with".

```
[Application data, MAC] session key
```

Data integrity is checked every time.