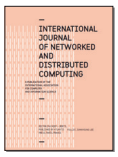
International Journal of Networked and Distributed Computing
Vol. 7(2); April (2019), pp. 68–84
DOI: https://doi.org/10.2991/ijndc.k.190326.003; ISSN 2211-7946
https://www.atlantis-press.com/journals/ijndc



A Formal Model of Distributed Security for Electronic Commerce Transactions Systems

Sylvanus A. Ehikioya[1,*], Adepele A. Olukunle[2]

[1]Department of Computer Science, Baze University, Abuja, Nigeria
[2]Department of Computer Science, University of Manitoba, Winnipeg, MB, Canada R3T 2N2

ARTICLE INFO

Article History

Received 25 December 2018
Accepted 21 February 2019

Keywords

Electronic commerce
distributed systems
distributed security
requirements
UML

ABSTRACT

Every distributed system requires a secure environment for its users. Security becomes even more important if users exchange sensitive information and value, across the network. An e-commerce environment is an example of a distributed system in which security is of a high priority. It is important that messages are confidential and tamper proof; users cannot repudiate transactions, and only authorized and properly authenticated users can access resources. These functionalities require proper security layer to provide access and sharing functions between the e-commerce systems and their respective customers. A Distributed Security Management System (DSMS) provides these services/functionalities. The DSMS is a security middleware for e-commerce servers, which coordinates secure communication, access and sharing of resources between the distributed applications, objects, databases and entities that make up the system. It is designed to provide an interface between clients and the databases of the merchants in a secure way, such that authorized clients can retrieve and send information to the system securely. It is required to have high quality and high tolerance for errors since the server is required to be always available continually. It must also have a user-friendly interface and feature set. The DSMS design discussed in this paper provides these requirements using public and private key systems, data encryption standard encryption, log files and a secure hashing algorithm. A prototype of the system was implemented using the Java security platform.

© 2019 *The Authors*. Published by Atlantis Press SARL.
This is an open access article distributed under the CC BY-NC 4.0 license (http://creativecommons.org/licenses/by-nc/4.0/).

1. INTRODUCTION

The rapid growth and advances in information and communication technologies have fueled the phenomenal growth, development and adoption of Internet-of-things and web-centric applications such as e-commerce. E-commerce entails the exchange of goods and services online over the Internet where all transactions are driven electronically. E-commerce has tremendously influenced conventional business transactions. E-commerce has grown rapidly, a lot of individuals and organizations now offer e-commerce services. E-commerce activities involve the transfer and exchange of e-commerce data set on the Web. The set of e-commerce data represents information about e-commerce activities or entities in the domain of e-commerce. The need to develop secure, reliable, and error-free e-commerce applications that guarantee the correctness of online transactions is well documented [1] in the literature.

E-commerce has made great impact on traditional business activities. Accessing the large volume of potentially heterogeneous e-commerce data distributed on the Web is becoming increasingly difficult because of the integrated e-commerce data from diverse data sources into a global view, while retaining the autonomy of the local data sources [2]. Regrettably, data integration remains a complex process that requires human intervention despite recent advances in technology[a]. Determining semantic relationships between a set of objects in one schema and a different set of objects in another schema is central to the integration process; see Adiele and Ehikioya [3–8] for more details. Mapping results that are mathematically unsatisfying cannot be tolerated in an e-commerce environment where accuracy is of the essence. A formal model of security system for e-commerce transactions provides a clear understanding of the system, reveal ambiguities, incompleteness, and contradictions in the informal definitions of mapping between domains, and thus permit the correctness verification of the integration components [9,10].

Four basic steps necessary for a transaction in an e-commerce system were examined in Ehikioya and Jayarama [11]. The four primary shopping steps in online transactions are: (1) product location (by viewing the hyperlinks to the page representing the product), (2) negotiation and cart placement (i.e. the consent to the advertised/marked price and the placement of the product into the shopping cart), (3) purchase of the product (by supplying the necessary payment information and/or effect the payment for the transaction and initiating a check-out event), and (4) transaction activity and status notification which completes the transaction (e.g., an invoice or a record of the transaction and status report). So any complete solution must strive to capture these steps. Obviously, a complete solution spans different areas. To provide a full functional e-commerce infrastructure requires providing solutions to the issues highlighted in each problem dimensions discussed in Ehikioya and Hiebert [12]. Resolving issues in one

*Corresponding author. Email: ehikioya@gmail.com
[a]Data integration, although crucial for e-commerce activities, will not be discussed in this paper. Data integration is a different subject in its own right.

dimension alone is inadequate, but concerted efforts at resolving all the issues in all the problem dimensions in an integrated manner is a fundamental requirement for the provision of functional e-commerce infrastructure. In this paper, we focus on the design and transactional management issues of the security problem dimension which provide mechanisms for supporting e-commerce transactions.

A formally designed and verified model of security for e-commerce transactions provides confidence and guarantees a correct, fail-safe, secure and robust transaction processing environment for e-commerce. The hall-mark of our model reported in this paper is the eventual provisioning of a trusting environment for the activities involved in e-commerce. Particular beneficiaries are business-to-business users, although all consumer-to-business functionalities are specified and implemented.

Ehikioya [13] emphasizes the need to develop secure (providing a framework for guaranteeing security of customers' and merchants' data), reliable (producing exact results in a consistent manner), and error-free e-commerce applications that guarantee the correctness (producing consistent and correct operational and transactional results) of online transactions. Guaranteeing the correctness (entails various stages in e-commerce transactions producing consistent results) of an e-commerce application is difficult, due to many situations in e-commerce transactions where errors could emerge, resulting in unexpected system failures. The application of mathematical and logic based formal specification techniques in developing e-commerce applications is widely believed to result in reliable and dependable applications[b].

Although several e-commerce models, such as business-to-business, business-to-consumer, consumer-to-consumer, and auction systems, have been identified in the literature [12,14], they (except the auction systems) have the same fundamental characteristics. However, the degree to which each model possesses the characterizing features of e-commerce systems solutions outlined in Ref. [15] varies. Nonetheless, the satisfaction of these features in a specification should be an integral part of the design objectives of any e-commerce system design.

1.1. Statement of the Problem

Information and data have become the most valuable and critical asset of most modern organizations, public or private. Increase in the value and volume of information passed across distributed database systems today makes the need for secure systems urgent. In most systems today, the communication channel is unreliable [16]. Thus, there is a need to protect highly confidential information (confidentiality). Access to resources and data also needs to be controlled (authentication and authorization). In e-commerce systems, there is an additional need to ensure that actions taken, such as data transfers and changes (in a transaction), cannot be denied (non-repudiation).

[b]Several specification methods are available for capturing ecommerce systems requirements, but these extant specification methods only have functionalities that can be used in modeling some components of e-commerce systems. However, neither of these existing formal specification methods has all the requisite functionalities to model all the components of an e-commerce system.

Participants in an e-commerce transaction expect a high level of confidentiality and privacy to protect the highly sensitive information (e.g., payment data, personal data, preferences, etc.) they provide in order to complete a transaction. Data in the different databases and those in transit need to be protected from access by unauthorized users (hackers) that can tamper with the integrity of, and perhaps steal, personal information and organizational plans. Furthermore, there is need for a standard technique for proof of participation, in order to ensure a commitment to initiatives taken. These needs motivate the desire to design and implement a secure middleware for the application/web server, which enables data to be accessed, and transferred securely over the communication network.

The scope of this paper is to design and implement security requirements in the second tier (application server end) of the three-tier distributed system. The design spans the six security controls; namely authentication, authorization, integrity, non-repudiation, confidentiality and availability. The implementation, however, is limited to integrating authentication, non-repudiation and confidentiality controls in the application server and the database server. Specifically, the implementation provides proper authentication (password, username and private key), privacy of messages (data encryption) and proof of participation (generation of log files) for a client that logs into the e-commerce server.

This paper discusses the characteristics of e-commerce systems and e-commerce transactions and models the security services in a typical business-to-consumer (B2C) e-commerce domain, which is more popular, more widely patronized by users, and attracts a higher volume of e-commerce transactions. We propose a design for implementing the primary security controls in a distributed system. This paper provides a description of the software architecture, the tools and the techniques that are used to achieve this goal, as a basis for module design and implementation.

The rest of this paper is organized as follows: In Section 2, we discuss the critical characteristics of e-commerce systems environment while we explain the properties of an e-commerce application in Section 3. In Section 4, we discuss the fundamental concepts in distributed security to provide the necessary foundation for the understanding of the details of our proposed secure middleware discussed in Section 5. Finally, we conclude the paper with a discussion of the results and limitations of our design in Section 6.

2. CHARACTERISTICS OF E-COMMERCE SYSTEMS

E-commerce systems enable customers to make online purchases. A typical ordering process in business-to-customer e-commerce systems allows customers to search and find items to purchase, negotiate the price of items, add items to a shopping cart, checkout items (i.e., purchase items), and pay for items purchased; the system also allows e-commerce merchants to update their inventory, verify customers' payment methods and plan logistics for shipping items to the customer.

E-commerce systems are inherently complex [17]. This complexity of e-commerce systems results from the concurrent, distributed, dynamic, and real-time behavior of e-commerce transactions:

(i) *Concurrency*: Many processes (a unit of concurrent activity) in e-commerce transactions may execute concurrently. This interaction may involve communication, synchronization, cooperation, parallelization, and competing for resources with other processes and the environment. For example, debiting a customer's credit card account and crediting the e-commerce service provider's credit card company could occur concurrently. However, concurrency controls (including serializations) must be in place to preserve transaction isolation.

(ii) *Distribution*: E-commerce systems are inherently distributed. An e-commerce application is usually distributed over possibly heterogeneous databases, web servers, networks, and operating systems across various locations. E-commerce transactions are characterized by complex data access patterns. Also, distributed e-commerce transaction processes can be invoked directly or indirectly from remote locations.

(iii) *Dynamism*: Dynamic systems have a number of states and changes are made to these states from time to time. When e-commerce transactions occur, various database tables are updated, therefore, changing the database state. In e-commerce systems, this dynamic behavior results in data dependencies as well as the need for proper synchronization and communication among the various subcomponents of the system.

(iv) *Real-time behavior*: E-commerce transactions occur in real time. Real time implies the interaction between the e-commerce system and its environment occurs instantaneously. The real-time behavior of e-commerce is influenced by the input environment and the application processing requirements. The system captures input from customers in real time. The real-time requirements of a system usually make some constraints on the input environment and output environment of the system.

(v) *Complex data access patterns*: E-commerce transactions inherently involve multiple data accesses across possibly multiple independent autonomous heterogeneous domains possibly across multiple distinct geographic locations. The propagation of access rights to data must be controlled while each autonomous domain enforces its security provisions.

Recall, e-commerce applications provide various back-end transactional processing and information access services across many heterogeneous databases and different networks which requires complex interaction relationships in order to fulfill a discrete e-commerce objective. While the number of users in the business-to-business domain is much smaller than the business-to-consumer domain (e.g., the business-to-consumer domain may involve several million users whereas business-to-business may involve just a few hundred or thousand users), both require a highly reliable system that always guarantees consistent and correct results.

Additional essential attributes of e-commerce applications have been identified in Ref. [15]. In particular, these attributes make e-commerce applications attractive candidates for formal modeling. However, these attributes will not be discussed in this paper.

3. DESIRABLE FEATURES OF E-COMMERCE APPLICATION

An e-commerce application must demonstrate some operational characteristics so that users (consumers and the owners of the system) can have some confidence in the system. Some of the essential critical operational characteristics of e-commerce applications, some of which are listed below, are available in Ref. [1].

- E-commerce applications should be fail-safe systems. A fail-safe system is characterized by a high degree of resilience, correctness, and graceful recovery in cases of errors. Whenever failure occurs, the system should gracefully recover from that failure to a state that is consistent with that failure and progressively continue execution. For example, if a customer pays for the contents of his/her shopping cart with a credit card and the transaction/system suddenly fails, the credit card should not be debited. In an ideal situation, on recovery, the system should display the contents of the shopping cart or execute other remedial actions to return the states of the credit card and inventory to their respective correct states.

- E-commerce applications should *guarantee correctness of operations*. Correctness is an important feature that any design of e-commerce transaction system should possess. E-commerce systems must operate correctly; provide correct and consistent transaction results, particularly for payment operations, manipulation of inventory, etc. For example, when a customer purchases items using the system, the system should correctly reduce the system's inventory by the exact quantity of items purchased by the customer, correctly debits the customer's credit card while crediting the same amount to the system's accounts receivable, and prints the correct shipping label for the customer. Similarly, if a customer returns an item, the customer's credit card account should be correctly credited by the appropriate amount as defined according to the company's return policy and also adds the corresponding number of items returned to the inventory. Thus, the system's specification must guarantee the following correctness types:
  - Computational results correctness.
  - Transaction execution correctness --- begin and abort transaction atomicity construct.
  - Operational (sequence) correctness --- order of transaction steps/serialization
  - Payment atomicity correctness.
  - Delivery atomicity correctness.
  - Product atomicity correctness.

Tygar [18] discusses the application of the traditional transaction atomicity concepts in determining correctness of e-commerce transactions and will not be discussed further in this paper.

- E-commerce applications/systems should be *dependable*. Besides the guarantee of correctness results (as discussed in the above paragraph), and demonstrating a high degree of performance, e-commerce systems must be dependable [19,20]

in the presence of faults. The desire to improve performance has resulted in algorithms of considerable sophistication, yet understood by few and prone to errors [21]. Laprie [22] defines dependability as "that property of a computing system that allows reliance to be justifiably placed on the service it delivers". Dependability[c] has other measures such as correctness, reliability[d] and availability. A system is reliable if it consistently produces the same results. E-commerce system availability is the degree to which an e-commerce system suffers degradation or interruption in its service to the users (customers and management) as a consequence of failures of one or more of its components. Thus, availability is a measure of the frequency the system is available to the users when required by the users (i.e., of being operational --- not failed --- at a given instant in time). Correctness is the delivery of service that follows specified requirements (i.e., the correctness of the state that a committed transaction produces). We must guarantee safety and liveness properties, which represent important characteristics of a transaction necessary to accurately describe any intended behavior. For example, mechanisms to deal with mutual exclusion and deadlock are safety properties that must be explicitly specified. Similarly, liveness properties include termination and responsiveness, while fairness properties ensure that every process has a chance of executing. Although dependability is an essential requirement for e-commerce systems, there has been little effort to specify and verify the techniques for attaining the desired resilience (e.g., failure recovery) formally.

The critical tasks of business processes that e-commerce systems fulfill reinforce the need for the reliability requirement. In addition, e-commerce systems cohesively integrate retailers, suppliers, financial institutions, and buyers to attain the values of engaging in the initiatives of e-commerce. This makes reliability a key feature. Therefore, the designer must design software reliability into the system.

- E-commerce applications should be *trustworthy* and *minimize risks*. Maintaining a high level of trust and minimization of risks should be an integral part of e-commerce transaction systems design. Trust involves both the system and people, hence the transactions. Trust is anchored on rules of behaviors for both the system and participants, called trust rules. These rules should be adequately captured in the requirements and design specifications. Risks, on the other hand, always exist because of uncertainty (a consequence of residual risk). Thus, the relationship between trust and risk is dual relationship.
- E-commerce applications should provide and guarantee *secure environment* for e-commerce activities. The transactions should be secure; maintaining the integrity of the application system's data. Similarly, the system must guarantee minimum security by implementing security modules for authorization, authentication (verification and validation) of users and their access rights, and the encryption of users' sensitive data such as personal data and credit card information. The system should maintain the confidentiality of confidential information. For example, credit card data and certain personal information should remain confidential. Packet snifters often try to collect data by examining raw data on data links in the network, however they cannot see encrypted data. For secure transmission, packet snifters are blind to encrypted data, thus making them useless. Lindquist [23] and Mahmoud [24] examined security considerations for distributed web-based e-commerce applications.

  In this paper, we focus on the basic user access security issues as well as the advanced security measures. It suffices to reiterate that provision of a secure environment for e-commerce transactions is paramount because of the critical tasks of business processes in e-commerce transactions.

  Besides the above known security issues/measures, the system must maintain and guarantee the integrity of communications between users. For example, the e-commerce server cannot repudiate any communication exchange among the users has taken place.

  An ideal e-commerce system should possess the following additional characteristics:

- Versatile.
- Extensible --- A system is extensible if it is designed to easily allow the addition of new features at a later date, e.g. through the use of extensible interface designs.
- Tested and proven correct.
- Easy deployment --- In the competitive environment of the knowledge and information-based economy, timing has always been a critical success factor to adopting new technologies. Businesses that are poised to succeed and remain competitive in the world marketplace are always very sensitive to introduce new systems in a coordinated but timely manner. Organizations need to quickly develop and deploy custom applications, and to easily refine and enhance them to improve their value. They need ways to seamlessly and efficiently integrate these applications with existing enterprise information systems and to scale them effortlessly to meet changing demands.

From the critical characteristics (properties and requirements) of electronic commerce systems/environment discussed in Section 2 above, and the properties of an e-commerce application in Section 3, clearly to have confidence in the transactions in the inherently complex and distributed environment, a clear and unambiguous model of the (distributed) security management is a fundamental requirement.

Notice that the properties of an e-commerce transaction and the characteristics of e-commerce systems/environments discussed above require the adoption of formal mathematical modeling and specification techniques in specifying and documenting the artifacts of an e-commerce system, the benefits of which are well documented in the literature [17,25] and are abstracted away from this paper.

We begin by first discussing the fundamental concepts in distributed security to provide the necessary foundation for the understanding of the details of our proposed security model in e-commerce environments.

[c]There are four approaches to achieving system dependability. These are: (1) *fault avoidance* to prevent, by construction, *fault* occurrence or introduction; (2) *fault tolerance* to provide, by redundancy, a service complying with the specification in spite of faults; (3) *fault removal* to minimize, by verification, the presence of faults; and (4) *fault forecasting* to estimate, by evaluation, the presence, the creation, and the consequences of faults. Specifying e-commerce transactions formally falls within the ambits of both approaches (1) and (3).

[d]Recovery is a subset of reliability. Since the focus of this paper is on systems' e-commerce (distributed) security management, recovery issues are integrated into the model.

4. DISTRIBUTED SECURITY MANAGEMENT

A distributed system is a collection of multiple autonomous computers, which are interconnected and integrated over a computer network, such that the applications and databases across the network are decentralized [26,27]. A good example of a distributed system is the client/server architecture in which the applications, databases and the workload are distributed between client computers and server systems. These components require proper security layer to provide access and sharing functionalities. A Distributed Security Management System (DSMS) is the software, which coordinates secure communication, access and sharing of resources between the distributed applications, objects, databases and entities that make up the system. It provides a middle layer (middleware) of abstraction between the various local security management systems running in each domain, and the network.

Distributed systems are gradually becoming commonplace. This pervasiveness of distributed systems, due mainly to growth in organizations and in the use of data, has created a need to access data and network resources in widely distributed locations, each of which still needs to retain its autonomy. For instance, an organization may extend its information system to include suppliers and customers. These new (distributed) entities, while having access to information on products and prices in the organization's database, are not authorized, for instance, to access employee data. There is, therefore, a need to specify levels of access for users within and between domains, ensure the identification of each entity, protect data while in transit, and hold users responsible for transactions they initiate. The DSMS provides these functions to local (possibly heterogeneous) systems seamlessly. A DSMS functions can be classified into two broad categories [28]: Secure communication between entities or processes [realizable through modules for authentication, integrity, confidentiality, and non-repudiation (acid test) functionalities], and assured allocation of access rights (achievable through module functionalities of authorization and availability).

Distributed systems offer higher reliability, availability and performance, but they result in greater complexity and a need for more security and control structures. The world's largest and most widely used distributed system, arguably the World Wide Web (WWW), provides a favorable and suitable paradigm/environment for e-commerce transactions; and it has three-tier client-server distributed architecture. This architecture is illustrated in Figure 1. The three-levels consist of:

1. *The clients:* These are single user computers that provide interfaces to user applications, connectivity to other clients and to servers. They also host the individual objects and processing services.
2. *The application servers:* These are specialized systems which host the application/business logic, the server objects and coordinate the connection between clients systems and the database server.
3. *The database servers:* These are specialized systems, which store the database of the network, the database management systems and execute the database commands.

The WWW refers to the collection of services that are available through the Internet based on the http protocol. The Internet is a communication network made up of several distributed networks. Application servers of domains that make up the Internet have

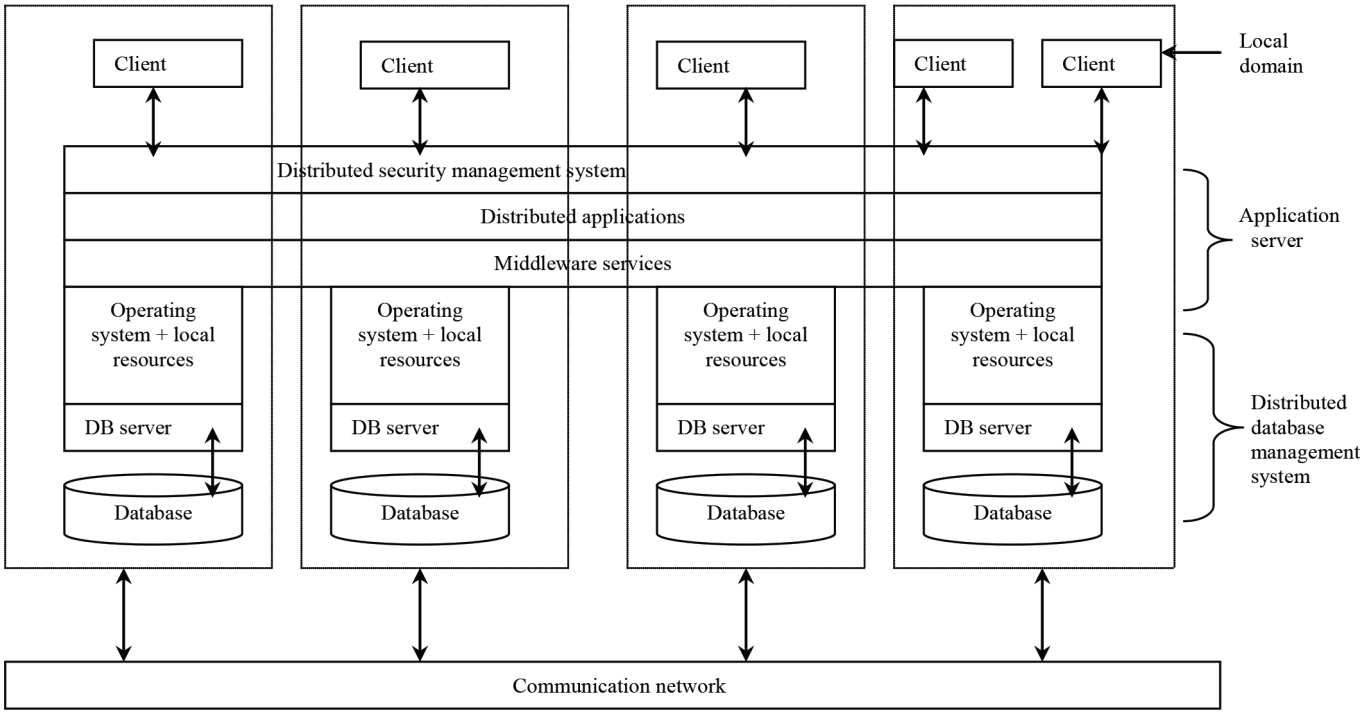


Figure 1 | A three-tier client-server distributed architecture.

the additional capacity to service HTML requests, and thus, can be regarded as web servers. Each web server needs a layer of logic (DSMS) that specifies access to that domain, and identifies it (and entities belonging to it) in order to communicate with other domains that make up the WWW.

E-commerce is one of the most important benefits of the Internet. Users of the Internet can exchange information and value across the distributed system. In this paper, we used e-commerce transaction systems as a case study for the design of a secure distributed system. An e-commerce environment is selected because it requires extremely high security standards. Therefore, we assume that securing our distributed database management system to meet e-commerce standards will most likely suffice for most distributed systems.

Security measures should be taken in all three tiers (client application, application server and database server) and along the communication network of the architecture to ensure end-to-end protection. According to Rajput [16], the primary security controls / functionalities for e-commerce systems are:

- *Confidentiality:* This is defined as "the protection of information from unauthorized disclosure to a person or computing entity" [16].
- *Access controls (authentication and authorization):* This implies ensuring the identity of the user of computing resources.
- *Integrity:* Includes all techniques to make data or computing resource tamper-proof.
- *Availability:* Refers to methods to ensure that there is continuity of resources and information. It implies protection from service attacks.
- *Non-repudiation:* Refers to measures taken to ensure irrefutable proof of participation by users, e.g., a user initiating a transaction would not be able to deny engaging in such activity.

5. METHODOLOGY

This section describes the design and the tools used to implement our DSMS. It includes the constraints taken into consideration in the design, the requirements to be met and a description of the tools needed to implement our design.

5.1. Domain Facets

Here, we describe the global and local domains obtainable in our DSMS. These include the users, the behavior of users, constraints and rules that define the local and global domain in which the security manager is used. This section also describes the technology, protocols and algorithms we employ in our design.

5.1.1. The intrinsics

5.1.1.1. Confidentiality and authentication

The use of cryptography[e] is one way of ensuring confidentiality within the e-commerce domain. One method of cryptography, which we used, is the public and private key cryptography [29,30]. This involves the use of two keys: a public key (for encryption), which is distributed to the public and a private key (for decryption) known only by the owner. Most public key cryptography algorithms work in conjunction with the Secure Sockets Layer (SSL) protocol. This form of encryption is particularly suitable in a distributed environment where it is difficult to maintain a key for every pair of communicators, and impossible to share a single key (that would not be compromised) amongst a large number of entities with varying levels of access.

In addition to protecting the message in transit, public and private key encryption also offers a level of authentication. This is because the receiver is assured that the message originated from the sender since it was encrypted with the sender's private key, and decrypted with the sender's public key. To authenticate the receiver, the sender can first use his/her private key to encrypt the message and then use the public key of the receiver to further encrypt the code. Thus, the sender is assured that only the receiver can access the message, using his private key and the sender's public key. In our design, each domain has a set of authenticated users. An authenticated user has a unique username and password that correspond to a unique ID. A user's ID specifies the level of access that the user has. Every user has a private key for further authentication and all other users have access to the users' public key. The confidentiality preservation and authentication protocol is shown in Figure 2.

5.1.1.2. Integrity

We implement integrity requirement using a message digest algorithm. A message digest is a fixed sized value that is calculated from the message that is sent. If the message is tampered, the message digest changes. To check the integrity, the sender sends the message and an encrypted message digest to the receiver, who recalculates the digest on receipt of the message. If this value varies from the encrypted digest, the receiver knows the message has been tampered with. Figure 3 illustrates our message digest implementation. Popular algorithms for calculating message digest include Secure Hashing Algorithm (SHA), MD2, MD4, and MD5 [29]. We used the SHA because of its simplicity and speed.

5.1.1.3. Non-repudiation

We implement non-repudiation using log files. Messages sent from the sender to the receiver are all recorded in a file. Information such as the IP address of the PCs used, the time and date, and a transaction identifier serve as irrefutable evidence that the transaction took place. These log files are stored in the machine of both the sender and the receiver, and the process of logging is automatic with every message sent and received. See Figure 3 for detail illustration.

5.1.1.4. Authorization

Access control lists can be used to implement authorization. An access list [31] is maintained on the web server, which specifies the computing resources the user is authorized to access. Security levels are used to group users into various degrees of system access.

[e]Cryptography implies changing the format of a message so that it does not make sense to viewers, unless a key is applied to decode the message.

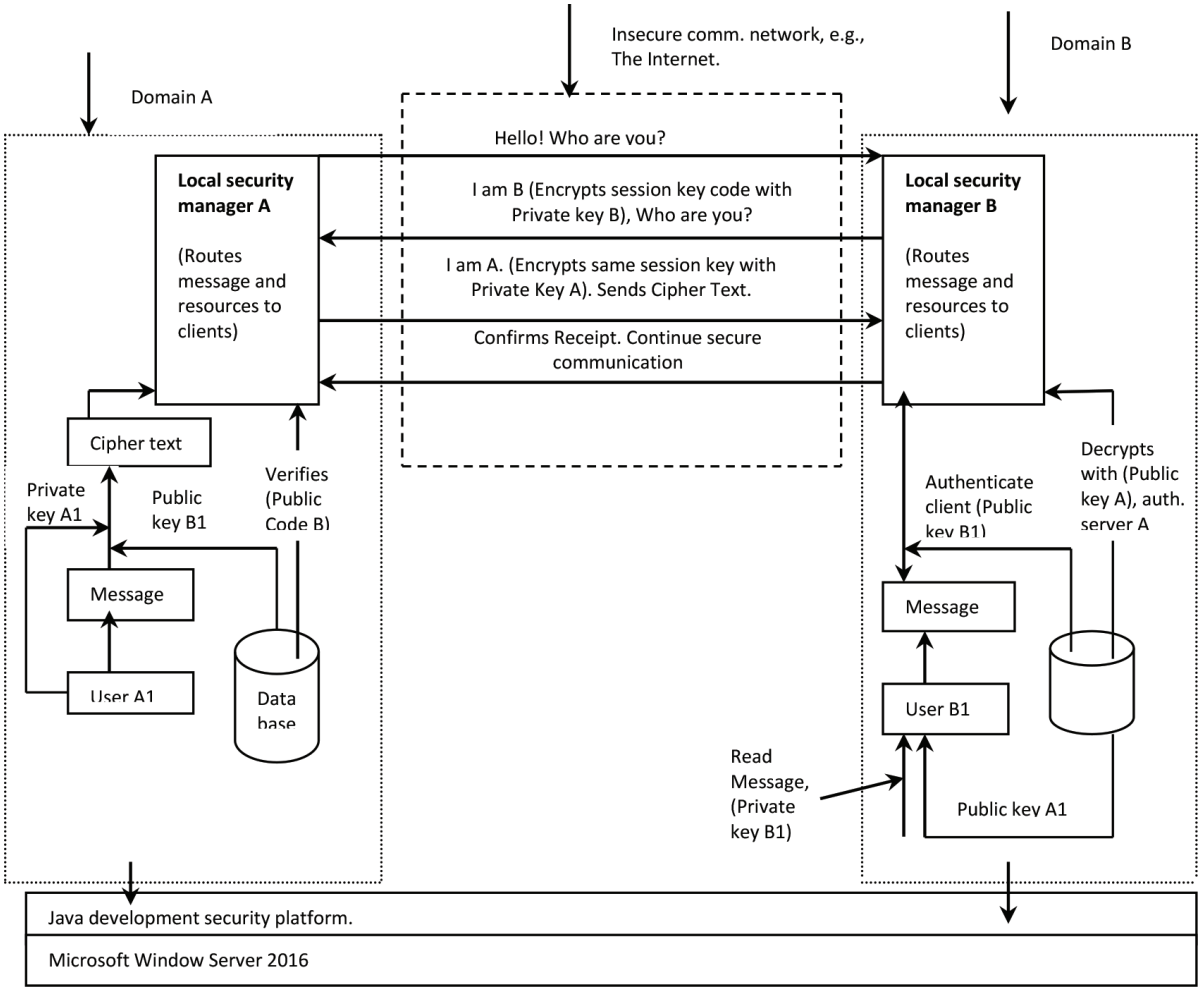


Figure 2 | Confidentiality and authentication in DSMS.

See Figure 3 for details. The access level could be based on the roles of the respective users, using the role-based access control model.

5.1.2. Operational support technology

The client and server platform used is Windows Server 2016 Edition [32,33]. This choice is based on the extensive security features available on this platform, such as the use of access control lists, passwords, and privileges. Other reasons for our choice are its robustness, scalability, and fault-tolerance. Furthermore, Windows Server is user-friendly [has Graphical User Interface (GUI) features] and is widely used in many distributed environments.

The following domain rules and regulations apply:

1. Clients are autonomous systems, which can belong to only one domain.
2. The distributed system is made up of a number of local domains at various locations.
3. The security policies and mechanisms in each local domain is local to that domain.
4. There must be mutual authentication before users in different domains can exchange information.
5. Users can obtain logs of operations with or without participating party's consent.
6. All entities in each domain have their private and public keys. All public keys are available in each database.
7. Each domain's application server has its own public and private key, which functions similarly to that of users.
8. Resources have access list in the domain they belong, which specifies entities that can access them.
9. Each public and private key is made up of two portions, which specify the domain identification and the entity identification.
10. Entity identifications are unique in each domain.

5.1.3. User behavior

The users of our distributed system can be classified as:

1. *Authorized users:* Users who have rights to access the database and the transmitted message. The users have various levels of access into the database. Each authorized user has a password and unique username, which is mapped into a unique domain

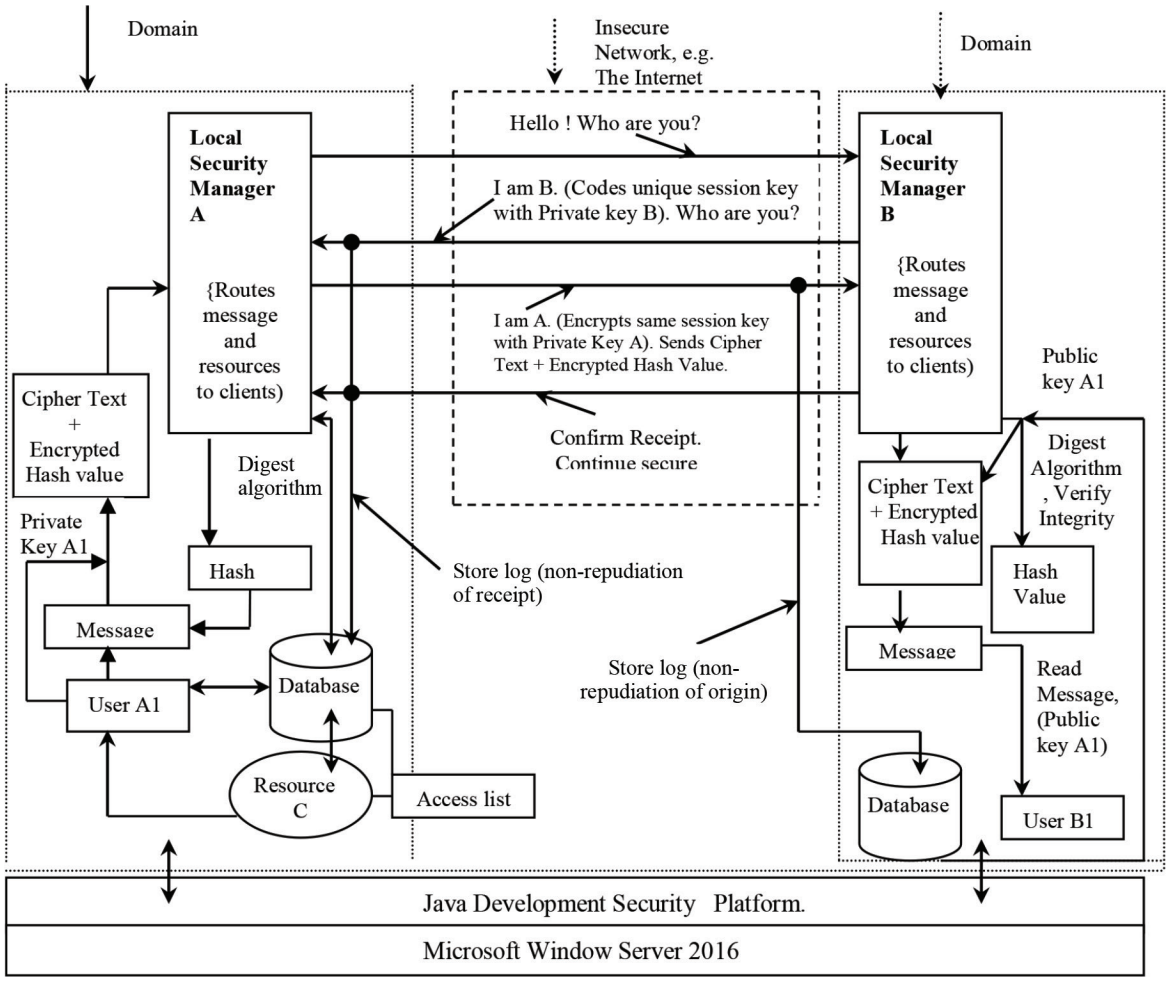


Figure 3 | Non-repudiation, authorization and integrity in DSMS

ID at the database level. Authorized users and their processes can access the domain, but are limited to operations defined by their access levels.

2. *Unauthorized Users:* Users who do not have right to access the database. This refers to unregistered clients and malicious individuals who may seek to violate the system.
3. *Third party participants* who can resolve repudiation conflicts. These are users such as the certification and registration authorities which give users their certification, for instance, thorough the use of public and private keys.

The following constraints (input, output, and problem space constraints) impact the manner in which the software is specified, designed, implemented or tested. These constraints are briefly summarized below.

5.1.3.1. Input constraints

1. We are limited to the use of a keyboard and mouse.
2. Messages to be sent are limited to a maximum of 128 characters (for the simplicity of this implementation).

5.1.3.2. Output constraints

1. The type of algorithm used determines the encrypted message size and format.
2. Error messages are displayed on the GUI only.

5.1.3.3. Problem space constraints

1. The communication channel is unreliable.
2. If the local security manager is not secure enough, then security at the global level is of little or no effect.

5.2. Requirements Specifications

The DSMS is a security middleware for e-commerce servers. It is designed to provide an interface between clients and the database of the merchant in a secure way, such that authorized clients can retrieve and send information to the system securely. It is required to have high quality and high tolerance for errors since the server is required to be available continually 24 h of the day. It must also

have a user-friendly interface and features set. The specific requirements for the DSMS follow. Note that a requirement is a statement of service of a system or a system constraint. It may range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification.

Here, we describe what we intend to achieve with this software, the different components available, and the high- and low-level software architecture, and the data flow diagram.

5.2.1. Functional requirements

Functional requirements refer to all security service requirements that should be met by the system. It provides answers to questions such as what security services should be achieved through this application. These requirements include:

1. *Confidentiality of transmitted information (message, data, or document)*: An intruder who accesses the network shall be unable to comprehend any message.
2. *Integrity:* An intruder shall not be able to alter or insert messages unnoticed. This requirement gives an assurance that a received message has not been tampered.
3. *Irrefutable proof of transaction:* Non-repudiation of origin: To provide the recipient with an evidence of the origin of a message, this makes it impossible for the sender to deny having sent it.
4. *Non-repudiation of receipt:* To provide the sender with evidence that the recipient has received the message, so that it cannot be denied that the message was received.
5. Defend system's information from unauthorized internal and external users.
6. Provide seamless end-to-end security by ensuring that customers and merchants shall be able to:
   - Encrypt their sensitive information,
   - Keep log of each other party's communication, and
   - Add an integrity check to a message to be sent.

A non-repudiation protocol can be implemented in two ways [34]: Either the exchange of evidence is done simultaneously, using digital signatures or log files, or the involvement of a trusted third party to assist in fair exchange, e.g., a delivery authority, a notary authority, or a certification authority. In the physical world, this method is akin to the "Proof of Delivery" message or signature requested of courier delivery service providers by senders of any package.

5.2.2. Non-functional requirements

The application development platform is Windows Server 2016 Edition. Our secure distributed system should be able to meet the following system quality of service non-security related constraints:

1. High reliability, availability and maintainability of the system.
2. High performance of the system. The overhead incurred by this system should not noticeably affect the performance of the distributed system or any of its sites.
3. The integration of the six security controls in various locations should be transparent to the user.

5.2.3. Development requirements

We implemented the DSMS application in the Java programming language. We used the Java programming language because the Java security platform includes support for security implementation.

The DSMS development follows all the standard constraints and rules for the maintenance of databases. All operations are valid and they conform to e-commerce trust standards.

To meet the standard usability requirements, we ensure adherence to Microsoft GUI standards [35] and making sure error messages are clear and unambiguous. Conformance to Microsoft's GUI standards makes it easier for users to easily learn how to use the system's functions. Icons are chosen to be meaningful and as helpful as possible to the user, sometimes illustrating their purpose. Clear and unambiguous error messages offer comprehension of errors (and their incident environments) and may proffer suggestions to solving the cause of the errors. Many GUI resources [36–38] exist to enhance the overall user experience in an e-commerce environment.

5.2.4. Domain requirements

The intended users of the DSMS system are located around the world, and are linked by the Internet. Their machines are most likely a heterogeneous collection of personal computers having differing operating system software and file formats. Messages could range from intention to communicate to credit card numbers. The DSMS must provide enough security to protect the most confidential message.

The system interfaces must provide universally consistent information, which are available on as many platforms as possible and incrementally extensible to as many clients as want to login. HTML is used as the user interface because of its simplicity and generality. Simplicity is required to enable the system easy to understand and use regardless of the operating system platform.

5.3. Software Design

First, we briefly describe the high level software architecture of the DSMS (see Figure 4) in order to provide context for the understanding and the relationships between components in the detailed design (see Figure 5) that follows. The DSMS architecture is divided into five modules. These modules are:

1. *The graphical user interface:* This serves as a link between the users and the local security manager. It also accepts inputs, e.g., messages and keys, and it displays the outcome of a transaction.
2. *The local managers:* This module consists of the local security manager and the database manager.
3. *The database tables:* This consists of the access list, the log, and the public key tables.

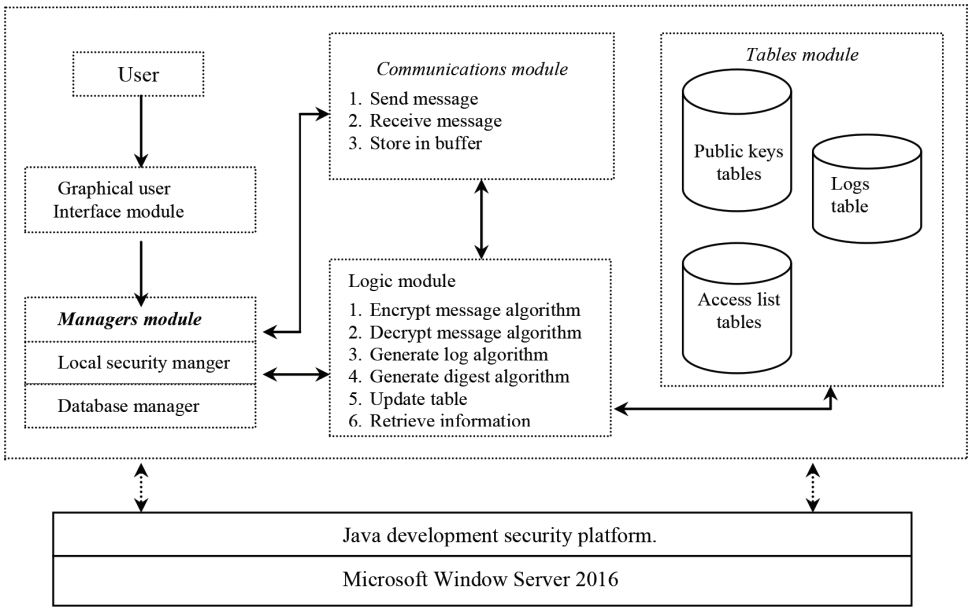


Figure 4 | High level software architecture of DSMS.

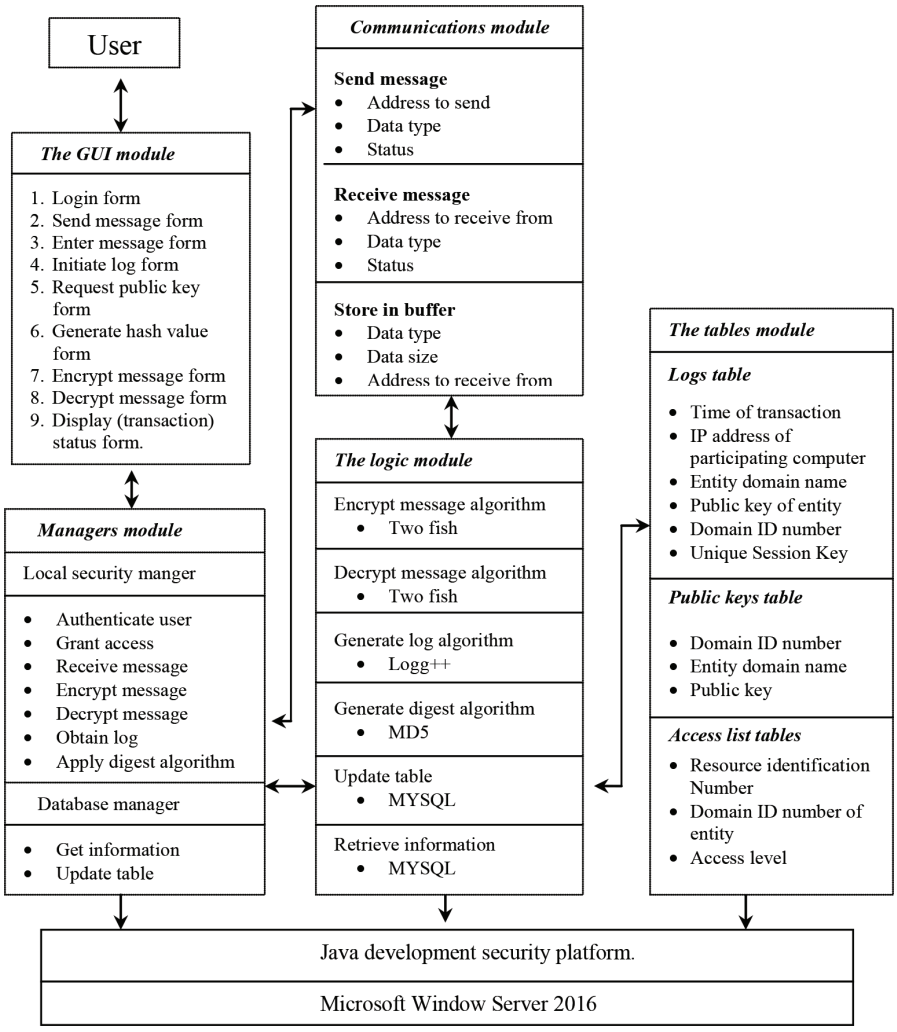


Figure 5 | Low-level software architecture of DSMS.

4. *The logic module:* This module is made up of the different algorithms required in the system. These are the generate digest algorithm, the store log algorithm, the encryption algorithm, and the decryption algorithm.
5. *The communication module:* This module controls the movement of data between the modules, and has it buffers to hold data in intermediate stages.

5.4. Detailed Software Architecture

Sommerville [39] provides a reference for the detailed specification of software architecture. The detailed software architecture shown below, see Figure 5, depicts all the components and sub-modules in each module within the application and how they interconnect with one another. Figure 6 shows the class diagram of our system.

5.4.1. Implementation tools

The implementation of our model benefits from the following tools with their relevance indicated for the appropriate functions/modules.

- *Server functions:* DHTML, ASP or Java servlets can be applied.
- *HTTP protocol* and DCOM is used for communication between the clients and the application server.
- *HTML* and *XML* used for the user interface on client machines.
- SQL is used to execute the database queries.
- *Java database connectivity, object database connectivity or RMI* used for managing the global database and implementing connectivity to local relational databases, e.g., MYSQL server for the databases.

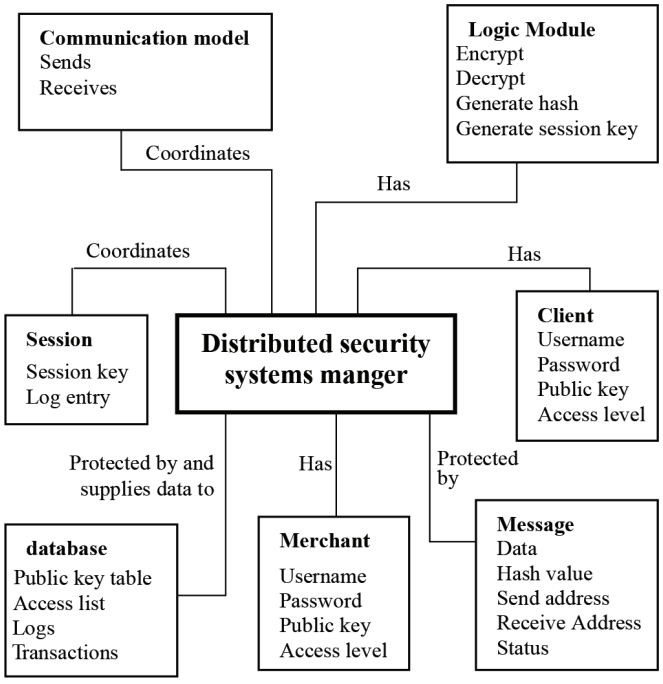


Figure 6 | Class diagram of software architecture of DSMS.

- *Java and JavaScript:* Java is a good tool for web-based implementations because it is platform independent. It provides useful tools for creating distributed applications. Furthermore, it is inherently secure. It uses a three-tier approach, which is appropriate for our design.
- *Security protocols*: We used SSL version 3.

Building e-commerce solutions [32,40] could benefit from several development tools (both commercial and non-commercial).

5.5. Design Specification

We begin with the problem description from the respective perspectives of both the client and the merchant.

Client's Perspective

- Must be assured he/she is communicating with the intended merchant.
- Information supplied must be kept confidential.
- Must know the status of message (acknowledgment). Must be assured of the integrity of the message received.
- Must have a log of merchant's communication.

Merchant's Perspective

- Must be assured he/she is dealing with whoever the client claims to be.
- Must be assured of the integrity of the message received.
- Transaction must be kept confidential.
- Must have proof of user's communication (for non-repudiation).

5.5.1. The DSMS architecture

The implementation of the DSMS is a three-tier architecture software system. It consists of the following three-tiered components: *User Interface* that runs on user's machine (GUI); *Application Server* that runs the business logic and algorithms that carry out the processes; and *Database Server* that coordinates requests for data, manages data, and stores logs.

5.5.1.1. Assumptions

- Each client has a public key and a private key.
- Each merchant has a public key and a private key.
- Public keys of users are available to all other users.
- Each client has unique username and password, which translates, to a domain ID in merchant's domain. The domain ID dictates the client's access level.
- Clients with no unique domain number have to register.
- All users must belong to at least one access level, but they could belong to more than one access level.

5.5.2. Functionality

1. **Authentication of clients**

   The DSMS receives a username and password on its log in GUI; that corresponds to the identification of the client. The DSMS must determine if the username and password is valid. If they are, the client's domain ID is checked (for access rights), and communication can begin, otherwise, the DSMS generates an error message for the client through the GUI, explaining the cause of the access denial. Reasons for denying access include:

- The username or password (or both) does not exist in the server's access list table.
- User is not registered. The error message GUI generates a link to the new registration GUI.

2. **Maintain history of transactions**

   The DSMS maintains a history of transactions with clients. This history should be sufficient to prove that the transaction transpired. It includes date, time, IP address and communication, unique session key, public key of client and domain ID.

3. **Ensure privacy of messages**

   The DSMS ensures that messages from the client are encrypted, to ensure that they are meaningless and unintelligible to an intruder.

4. **Integrity of messages**

   The DSMS calculates the hash value of every message and sends it along with the message. If the hash value differs when it is recalculated on delivery, the message has been tampered with. An alert is sent through the GUI to the receiver. This mechanism enables the receiver to ascertain the integrity of the received message.

5.6. Algorithmic Designs/Pseudocodes

We present the design of the algorithms of the functionalities of the system using the informal structured pseudocode approach. The simplicity and ease of understanding of pseudocode makes this approach our candidate choice. We provide our design with some structure and appropriate annotation to support the simplicity attribute. For example, in the "Purpose" and "Remarks" sections of the pseudo-codes, we provide commentary on the algorithms.

SET 001: Intention to communicate

Index:	**001**
Name:	**Intention to communicate**
Purpose:	To indicate an intention to communicate and to ensure communication with actual merchant, say B.
Input:	Address of merchant, B. Hello message and request for identification
Output:	None
Actions:	Initiate intention to communicate Request for proof of identification

SET 001: Intention to communicate

Index:	**001**
Name:	**Intention to communicate**

Pseudo code:	**begin**
	Open Intention_to_send form;
	Input B.address;
	Input Text "Hello! Who are you";
	function communicate_message(message);
	end;

Time constraints:	None
Exceptions:	None
Remarks:	This function is called when the client specifies the web address of the merchant in the web browser. The main page of the merchant appears and the client clicks on server authentication.

SET 002: Merchant identification

Index:	**002**
Name:	**Merchant identification**
Purpose:	To prove identity; To send unique session key; to request for identity.
Input:	Address of client, A Unique session key encrypted with private key of B Request for identification
Output:	Cipher text to A
Actions:	Generate unique session key in database Encrypt it with private key **function** message (cipher text) Request for identification of A

Pseudo code:	**begin**
	Open identification form;
	function Generate_session_key;
	function (D)encrypt_message (session key, private key of B);
	Input A. address;
	function communicate_message(cipher text);
	end;

Time constraints:	None
Exceptions:	None
Remarks:	This function is called after a client clicks on server authentication object. The merchant identification screen appears. It shows a cipher text its plain text equivalent. The clievnt selects the public key of the merchant from the link to the public key tables. Decrypting to obtain the original text shows that the message really originated from the merchant.

SET 003: Merchant authentication

Index:	**003**
Name:	**Merchant authentication**
Purpose:	To confirm identify of merchant, B. To obtain session key. To log session
Input:	Cipher text, public key of merchant
Output:	Unique session key, merchant approved or denied, log of session.
Actions:	Obtain public key of merchant from database. Run decryption algorithm on cipher text. Authenticate merchant and store session key,

SET 003: Merchant authentication

Index:	**003**
Name:	**Merchant authentication**

Psuedo code:	**begin**
	Receive Message;
	Read (publickeytable)
	select publicytable.publickey.B
	function decrypt
	(publickeytable.publickey)
	{return result}
	if (result= = valid session key) **then**
	{
	Generate log
	Store session key
	begin login
	}
	else output "Error in session";
	endif;
	end

Time constraints:	None
Exceptions:	Public key of merchant not available
Remarks:	None

SET004: New client registration

Index:	**004**
Name:	**New client registration**
Purpose:	To register new client, A.
Input:	First name, last name, public key, username and password, Encrypted session key (with private key A).
Output:	User ID and password
Actions:	Ensure required information is complete and correct. Ensure there is no duplication of the unique username. Ensure same session key is returned. Set up username, password and public key in database system. Create unique domain ID within database and set access level.

Pseudo code:	**begin**
	Open main window;
	Input session key cipher;
	read (publickeytable)
	select publickeytable.publickey.A
	function decrypt algorithm
	if (result = valid original session key) **then**
	{
	Open registration form
	Input firstname
	Input lastname
	Input public key
	Input username
	if (username =
	validusername.accesstable) **then**
	{
	output "Existing username,
	please try again!"
	input username
	}
	endif
	Input password;
	if (input format is incorrect
	or input or missing) **then**
	output "Please enter correct input"
	else

SET004: New client registration

Index:	**004**
Name:	**New client registration**

	{
	open connection to database
	store firstname, lastname,
	publickey, username and
	password in access table;
	generate domain ID and access
	level;
	store domain ID and access level;
	}
	endif;
	}
	else
	output "Send correct session key"
	endif;
	end

Time constraints:	None
Exceptions:	1. There is a duplication of usernames and public keys. 2. Mandatory personal information is missing. 3. Client is already registered. 4. Input format is incorrect. 5. Client does not have a public key. 6. Session key is incorrect.
Remarks:	Verify public key with certification authorities to confirm identity. In the event of duplication, deny registration and ask client to use another name. If information is not complete, session key or format is incorrect, deny registration and request for information.

SET 005: Logon client

Index:	**005**
Name:	**Logon client**
Purpose:	To authenticate and authorize client A, into merchant's domain.
Input:	Client's username, password. Encrypted session key (with private key).
Output:	Access gained or Denied
Actions:	1. Check access list if user's user name and password is authentic, check level of authorization using domain ID. 2. Ensure session key is correct. 3. Error in 1 or 2? Deny session.

Pseudo code:	**begin**
	input cipher(session key)
	read (publickeytable)
	select publickeytable.publickey.A
	run decrypt algorithm
	count =1;
	if (result = valid original session key)
	then
	{
	Open login form
	Input username and password;
	read (accesstable)
	if (input.password =
	accesstable.password and
	input.username =
	accesstable.username) **then**
	{

SET 005: Logon client

Index:	**005**
Name:	**Logon client**

	generate log;
	start access;
	open transaction information;
	}
	else output " Error in session";
	return to main window
	endif;
	}
	else
	{
	Output message "Incorrect parameters.";
	Return to login form;
	count = count++
	}
	endif;
	if (count > 3) **then**
	output message "login blocked";
	endif;
	end

Time constraints:	If access is denied 3 consecutive times, then display "login blocked" message.
Exceptions:	Client is not registered, request to register.
Remarks:	Clients have different access levels based on their domain IDs. Unregistered clients will be asked to register.

SET 006: Enter and protect message

Index:	**006**
Name:	**Enter and protect message**
Purpose:	To enter message. To calculate hash value of message and encrypt both using public key B and private key A.
Input:	Text message, user's private key, merchant's public key.
Output:	Cipher Text
Actions:	1. Calculate hash value of message. 2. Run encrypt algorithm on message with merchant's public key and hash value. 3. Run encrypt algorithm on result with client's private key.

Psuedo code	**begin**
	At transaction information window;
	open message form;
	input message;
	procedure hash algorithm(message);
	{return hash value}
	read (publickeytable);
	select publickeyB;
	procedure (D)encrypt_message (message, public key.B);
	{return result}
	input private key;
	procedure (D)encrypt_message (result, private key.A);
	{return cipertext};
	end;

Time constraints:	None
Exceptions:	If there is no message, Click end of transaction.
Remarks:	Merchant's public key is assumed to be available to client, A

SET 007: Communicate_message (message)

Index:	**007**
Name:	**Communicate_Message (message)**
Purpose:	To send/receive a message/cipher text.
Input:	Message, address of destination
Output:	None
Actions:	Activate communication module and send/receive a message.

Pseudo code:	**begin**
	At transaction information window
	function communication module (IPadressB, ciphertext)
	if (arrived = = true) **then**;
	return (delivered_status)
	else
	return (output "error in sending");
	end

Time constraints:	None
Exceptions:	Communication module is down
Remarks:	None

SET 008: Receive message (message)

Index:	**008**
Name:	**Receive message (message)**
Purpose:	To receive message
Input:	Cipher text, address of source
Output:	Original message Status of message integrity
Actions:	Decrypt cipher using public key A. Decrypt result using private key B. Calculate hash value of message. Compare with decrypted hash value to confirm integrity of message. Activate log taking.

Pseudo code:	**begin**
	procedure communicate_message (IPaddress of A);
	{receive cipertext}
	input public key A;
	procedure Decrypt algorithm (cipertext, public key.A);
	{return result}
	input private keyB;
	procedure Decrypt algorithm (result, private key.B);
	{return hash value}
	procedure hash algorithm(message);
	{return hash received}
	if (hash value = = hash received) **then**
	{
	Output "valid message"
	store message
	status = "message intact";
	}
	else
	{
	procedure communicate message ("invalid message");
	status = "message tampered"
	return (status);
	}
	endif;
	end

SET 008: Receive message (message)

Index:	**008**
Name:	**Receive message (message)**
Time constraints:	None
Exceptions:	Hash values differ.
Remarks:	Send message tampered error!

SET 009: Calculate hash value

Index:	**009**
Name:	**Calculate hash value**
Purpose:	To generate a hash value for message (SDA algorithm)
Input:	Message (string, S)
Output:	Message digest of S
Actions:	Get string S Calculate hash for each 512 bytes of S For the last block, pad with zeros Update hash after each block Return Hash

Pseudo code:	**begin**
	get string S;
	Holder = number of bytes of S;
	Hash=0;
	Bytes=0;
	while (holder-bytes < 512)
	{
	calculate hash value for each 512 bytes of S;
	bytes = bytes+512;
	update hash;
	}
	return hash;
	end

Time constraints:	None
Exceptions:	None
Remarks:	None

SET 010: Generate session key

Index:	**010**
Name:	**Generate session key**
Purpose:	To generate a unique session key
Input:	none
Output:	Session key
Actions:	Seed the generator (to prevent repeats) Generate a number between 0 and 5000
Pseudo code:	**begin** seed random number generator; sessionkey = generate a number between 0 and 5000; **return** (sessionkey); **end**
Time constraints:	None
Exceptions:	None
Remarks:	Seeding the generator ensures that each session key is unique.

SET 011: (D)Encrypt message

Index:	**011**
Name:	**D(E)ncrypt_message (DES)**
Purpose:	To encrypt or decrypt a string S
Input:	String S/cipher
Output:	Cipher text
Actions:	Read String Generate encryption key Use Key to encrypt string

Pseudo code:	**begin**
	read input string;
	B = convert string to binary;
	for every block of 64 bits (convert string to key)
	{
	Use permutation table, get 56 bit permutation;
	Split into 2 halves ($Left_0$ and $Right_0$);
	for i = 1–16
	do {
	$Left_i = Left_{i-1}$;
	$Right_i = Right_{i-1}$;
	left shifts for each halve of 28 bits (left and right);
	Key_i = concatenate pairs ($Left_i$ and $Right_i$)
	use 2nd permutation table, on Key_i
	to get 46 bit permutation;
	}
	Encrypt using keys
	For every 64 bit block,
	use 1st Permutation table on message B,
	to get 64 bit permutation;
	Split into two halves, ($Left_0$ and $Right_0$);
	for i = 1–16
	do
	{
	$L_i = R_{i-1}$;
	$R_i = L_{i-1} + f(R_{i-1}, Key_i)$;
	Concatenate 32 bit pairs;
	}
	return 64 bit cipertext;
	end

Time constraints:	None
Exceptions:	None
Remarks:	None

SET 012: Generate log

Index:	**012**
Name:	**Generate log**
Purpose:	To create log for current session
Input:	Session key
Output:	Log table
Actions:	For unique session key Get IP address of machines Get date, time and message Store in table

SET 012: Generate log

Index:	**012**
Name:	**Generate log**

Pseudo code:	**begin**
	procedure create_ table_ log (*k*) ;
	if (session key = *k*) **then**;
	Get time, username, IP of machines,
	message;
	update in Log table *k*;
	endif;
	end

Time constraints:	None
Exceptions:	None
Remarks:	None

6. CONCLUSION AND FUTURE DIRECTION

Every distributed system requires a secure environment for its users. Security becomes even more important if users exchange sensitive information and value, across the network. An e-commerce environment is an example of a distributed system in which security is a high priority. It is important that messages are confidential and tamper proof, users cannot repudiate transactions, and only authorized and properly authenticated users can access resources.

The DSMS design discussed in this paper provides these requirements using public and private key systems, data encryption standard encryption, log files and a SHA. A prototype of the system was implemented using the Java security platform.

This paper can benefit from further extension in several directions. The DSMS assumes that every participant has a public and private key. This is not the case in real world systems. Therefore, it is important to create a DSMS that removes this limitation by devising other efficient and simple mechanism that relies on phenomena applicable to all users. Another major limitation is that this system generates a session key each time communication is established between the client and the merchant's server, and this key has no relationship with the identity of the client. It would be more secure if the session key can reflect the frequency of visits and the identity of that particular client. To take care of this limitation, designing and developing a web content logging mechanism with the capability to associate client identity with the logged web content is clearly desirable. Finally, the specifications of our DSMS, particularly the pseudocode algorithms, can benefit from using formal methods to specify the functionalities of the system. Using a formal method which is formally verifiable provides a correct, fail-safe, and robust transaction-processing environment for e-commerce.

CONFLICTS OF INTEREST

There is no conflict of interest.

REFERENCES

[1] S.A. Ehikioya, A formal characterization of electronic commerce transactions, Int. J. Comput. Inform. Sci. 2 (2001), 97–117.

[2] M.P. Reddy, B.E. Prasad, P.G. Reddy, A. Gupta, A methodology for the integration of heterogeneous databases, IEEE Transactions on Knowledge and Data Engineering 6 (1994), 920–933.

[3] C. Adiele, S.A. Ehikioya, Evolving a "wise" integration system for e-commerce transactions, J. Electron. Commer. Res. Appl. 6 (2007), 219–232.

[4] S.A. Ehikioya, C. Adiele, A formal model of dynamic identification of correspondence assertions for e-commerce data integration, Int. J. Comput. Intern. Manage. 13 (2005), 52–66.

[5] C. Adiele, S.A. Ehikioya, Algebraic signatures for scalable web data integration for electronic commerce transactions, J. Electron. Commer. Res. 6 (2005), 56–74.

[6] S.A. Ehikioya, C. Adiele, Algebraic signatures to analyze correspondence assertions for web data integration, Int. J. Comput. Inform. Sci. 10 (2009).

[7] C. Adiele, S.A. Ehikioya, Towards a formal data management strategy for a web-based community, Int. J. Web Based Communities 1 (2005), 226–242.

[8] C. Adiele, S.A. Ehikioya, Managing Data in a Data-Centric E-Commerce Application Model, Proceedings of the Seventh International Conference on Electronic Commerce Research (ICECR7) INFOMART, Dallas-Texas, ATSMA Inc., 2004, pp. 421–430.

[9] J.M. Wing, A symbiotic relationship between formal methods and security, Proceedings of IEEE Computer Security, Dependability, and Assurance: From Needs to Solutions, IEEE, New York, UK and Williamsburg, VA, USA, 1998, pp. 26–38.

[10] S.A. Ehikioya, A Formal Framework for Electronic Commerce Transactions, International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet (SSGRR-2000), L'Aquila, Italy, Scuola Superiori G. Reiss Romoli, 2000.

[11] S.A. Ehikioya, J. Suresh, Electronic Commerce for Services and Intangible Goods, *First International Conference on Internet Computing*, Monte Carlo Resort, Las Vegas, Nevada, USA, CSREA Press, 2000.

[12] S.A. Ehikioya, K. Hiebert, A Formal Model of Electronic Commerce, First International Conference on Software Engineering, Networking, and Parallel and Distributed Computing, Champagne-Ardenne, France, ACIS, 2000.

[13] S.A. Ehikioya, Design of E-commerce Systems, CS 74.786 Lecture Notes, Winter Term 2002, Computer Science Department, University of Manitoba, Canada, 2002.

[14] B. Ola, A Comparative Evaluation of Specification Techniques for Electronic Commerce System, MSc Thesis, Department of Computer Science, The University of Manitoba, Winnipeg, MB, Canada, 2004.

[15] S.A. Ehikioya, A formal perspective to modelling electronic commerce transactions, Colomb. J. Comput. 2 (2001).

[16] W.E. Rajput, E-Commerce Systems Architecture and Applications, Artech House, Boston, MA, USA, 2000, pp. 231–239, 315–363.

[17] S.A. Ehikioya, Specification of Transaction Systems Protocols, Ph.D. Thesis, Department of Computer Science, The University of Manitoba, Winnipeg, MB, Canada, 1997.

[18] J.D. Tygar, Atomicity versus Anonymity: Distributed Transactions for Electronic Commerce, Proceedings of the 24th International Conference on Very Large Data Bases, VLDB'98, Morgan Kaufmann Publishers Inc., New York, NY, USA, 1998, pp. 1–12.

[19] C.R. Das, J.T. Kreulen, M.J. Thazhuthaveetil, L.N. Bhuyan, Dependability modeling for multi-processors, Computer, IEEE, 1990, pp. 7–19.

[20] V.P. Nelson, Fault-tolerant computing: fundamental concepts, Computer, IEEE, 1990, pp. 19–25.
[21] Y. Gurevich, N. Soparkar, C. Wallace, Formalizing database recovery, J. Univers. Comput. Sci. 3 (1997), 320–340.
[22] J.C. Laprie, Dependability: a unifying concept for reliable computing and fault tolerance, in: T. Anderson, Dependability of Resilient Computers, Chapter 1, Blackwell Scientific Publications, Oxford, 1989, pp. 1–28.
[23] T.C. Lindquist, Security considerations for distributed web-based e-commerce applications in Java, Proceedings of the 35th Annual Hawaii International Conference on Systems Sciences, IEEE Big Island, HI, USA, 2002.
[24] Q. Mahmoud, Securing component-based e-commerce applications, Security Advisor Zone, Component Advisor Magazine, 2000, pp. 2–5. Available at: http://www.advisor.com/Articles.nsf/aid/MAHMQ03.
[25] S.A. Ehikioya, K.E. Barker, Towards a Formal Specification Methodology for Transaction Systems Protocols, 3rd Annual IASTED International Conference on Software Engineering and Applications (SEA'99), Scottsdale, Arizona, ACTA Press, USA, 1999.
[26] A. Tanenbaum, M. van Steen, Distributed Systems: Principles and Paradigms, Prentice Hall, New Jersey, 2002, pp. 413 –489.
[27] J. Whitten, L. Bentley, Systems Analysis and Design Methods, McGraw-Hill, New York, USA, 1998, pp. 23–60.
[28] M. Ozsu, Next Generation Distributed DBMSs: Some Views and Research Directions, Laboratory for Database Systems Research, Department of Computing Science, University of Alberta, Canada, 2002. Available at: http://db.uwaterloo.ca/~tozsu/presentations/future_ddbms.pdf.
[29] A. Hyde, M. Peirano, J. Oliver, D. Vasiliev, A. Wolf, J. Gerber, et al., CryptoParty Handbook, 2013. Available at: https://github.com/cryptoparty/handbook.
[30] R. Oppliger, Contemporary Cryptography, second ed., Artech House, Boston, MA, USA, 2011.
[31] R. Chopra, Operating Systems: A Practical Approach, Revised Edition, S. Chand and Company Ltd, New Delhi, India, 2012.
[32] J. McCabe with the Windows Server Team, Introducing Windows Server 2016, Microsoft Press, Microsoft Corporation, Richmond, Washington, USA, 2016.
[33] The Essential Guide to Windows Server 2016, PRO+ Content, Tech Target, 2016. Available at: http://pro.techtarget.com/ProLP/.
[34] R. Doemer, P. Ezers, P. Gustavsson, C. May, G. Shull, Building E-commerce Solutions with Net.commerce: A Project Guidebook, IBM International Technical Support Organisation, 1999. Available at: http://www.redbooks,ibm.com/pubs/pdfs/redbooks/sg245417.pdf.
[35] Microsoft Corporation, The Windows Interface Guidelines for Software Design, Microsoft Press, Richmond, Washington, USA, 1995.
[36] J. Moule, Killer UX Design, SitePoint Pty Ltd, Victoria, Australia, 2012.
[37] T. Firdaus, Responsive Web Design by Example: Beginner's Guide, Packt Publishing, Birmingham, England, 2017.
[38] The Basics of User Experience Design, The Interaction Design Foundation. Available at: http://www.interaction-design.org. Accessed on June 29, 2017.
[39] I. Sommerville, Software Engineering, tenth ed., Addison-Wesley, New York, USA, 2015.
[40] M. Reynolds, Beginning E-commerce with Visual Basic, ASP, SQL Server 7.0 and MTS, Wrox Press Ltd., Birmingham, England, 2000.