International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:1, No:3, 2007

Ensuring Data Security & Consistency in FTIMA - A Fault Tolerant Infrastructure for Mobile Agents

Umar Manzoor, Kiran Ijaz, Wajiha Shamim, and Arshad Ali Shahid

***Abstract*—**Transaction management is one of the most crucial requirements for enterprise application development which often require concurrent access to distributed data shared amongst multiple application / nodes. Transactions guarantee the consistency of data records when multiple users or processes perform concurrent operations. Existing Fault Tolerance Infrastructure for Mobile Agents (FTIMA) provides a fault tolerant behavior in distributed transactions and uses multi-agent system for distributed transaction and processing. In the existing FTIMA architecture, data flows through the network and contains personal, private or confidential information. In banking transactions a minor change in the transaction can cause a great loss to the user. In this paper we have modified FTIMA architecture to ensure that the user request reaches the destination server securely and without any change. We have used triple DES for encryption/ decryption and MD5 algorithm for validity of message.

***Keywords*—**Distributed Transaction, Security, Mobile Agents, FTIMA Architecture.

I. Introduction

TRANSACTIONmanagement is one of the most crucial requirements for enterprise application development. Most of the large enterprise applications in areas of finance, banking and electronic commerce rely on transaction processing for delivering their business functionality. Given the complexity of today's business requirements, transaction processing occupies one of the most complex segments of enterprise "level" distributed application to build, deploy and maintain. Enterprise applications often require concurrent access to distributed data shared amongst multiple components. Such applications should maintain integrity of data under the following circumstances.

- Distributed access to a single resource of data
- Access to distributed resources from a single application component.

The use of transactions is a very popular concept for the management of larger data collections. Transactions guarantee the consistency of data records when multiple users or processes perform concurrent operations. The access of distributed resources (databases on different computers) within a transaction is called a distributed transaction. The architecture under consideration has been built by Summiya et al under the title of "A *Fault Tolerance Infrastructure for Mobile Agents"*. The main goal of this architecture is to provide a fault tolerant behavior in distributed transactions and in order to achieve this, a multi-agent system has been devised which well suits distributed transactions and their processing.

A. FTIMA Architecture

In the existing FTIMA architecture there are few vulnerable points which greatly require security as shown in Fig 1. Whenever a user provides his/her login information (to the Client Application), this data needs to be protected as it may contain login ids, passwords, credit card numbers, money transaction requests in case of an ATM or banking scenario which is confidential information and can we misused if accessed by the intruder. This data flows through the network until it reaches the Transaction Manager which forwards the request to a suitable path available at that time.

The data after traveling through the appropriate path reaches the destination server which manipulates this information after interpretation. It can be seen clearly that such information which flows through the network might contain personal, private or confidential information. Moreover in financial transactions for example in banking transactions a minor change in the transaction request can cause a great loss to the user. Therefore the main concern of this paper is to ensure that the user request reaches the destination server securely and without any change in distributed transactions.

This paper is organized as follows. The first section discusses the related work done in securing distributed transactions. This section is followed by the discussion of the modified FTIMA architecture which ensures security at least to the minimum required level. Section 4 contains

Manuscript received March 13, 2006.

Umar Manzoor is with the Department of Computer Science, National University of Computer & Emerging Sciences, (FAST-NU), Islamabad, Pakistan. (phone: +92-111-128-128 e-mail: umarmanzoor@gmail.com).

Kiran Ijaz is with the Department of Computer Science, National University of Computer & Emerging Sciences, (FAST-NU), Islamabad, Pakistan (e-mail: kiranijaz@gmail.com).

Wajiha Shamim is presently working as Software Engineer at a software development company "Trivor Software" Islamabad, Pakistan (e-mail: wajiha84@gmail.com).

Dr Arshad Ali Shahid is Head of the Department of Computer Science, National University of Computer & Emerging Sciences, (FAST-NU), Islamabad, Pakistan. (e-mail: arshad.ali@nu.edu.com).

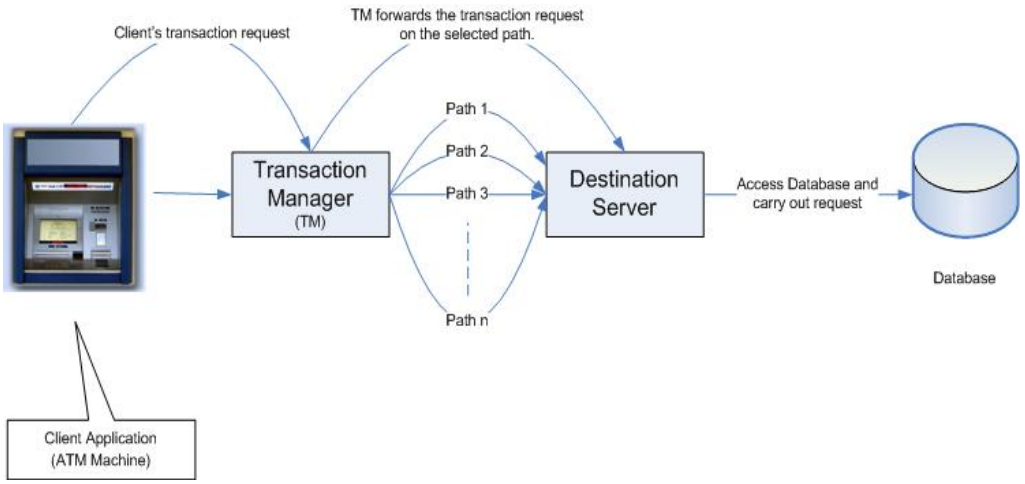

Fig. 1 A Fault Tolerance Infrastructure for Mobile Agents (FTIMA) Architecture

experimental evaluation on test cases which have been provided to understand the solution and its importance in banking transactions. Finally conclusion is presented in section 5.

II. Related Work

A few studies have been done on how to secure distributed transactions. [1] Practical techniques for securing distributed computing systems have been discussed. It reviews critical risk areas in distributed systems, including networks, operating systems, applications, middleware, and the Internet. It presents detailed information about technologies that can help users respond, including: Cryptography, The Kerberos authentication model and DCE security. Another architecture has been presented for a mobile agent system in [2] which offers fault tolerance for the whole agent system at a high level.

This architecture additionally guarantees security for the host as well as security for the agent. To handle these issues for mobile agents they have used various encryption mechanisms and other methods. More work has been done in [3] in which the authors consider techniques for designing and analyzing distributed security transactions. They have presented a layered approach, with a high-level security transaction layer running on top of a lower-level secure transport protocol. Moreover [4] presents a high-level view of existing security challenges related to clusters and proposes a structured approach for handling security in clustered servers which is worth mentioning in this context.

III. Modified Architecture

If Keeping the architecture (FTIMA) discussed in introduction in mind we can see that the client's critical data flow needs to be protected from anyone who could possibly be cracking passwords or trying to get private information with the intent to misuse the information etc. The information flowing from the Client Application to the Transaction Manager which ultimately reaches the destination server is private and confidential which needs to be encrypted to ensure this confidentiality as shown in Fig. 2.

Moreover we want to ensure that nobody tries to modify the request or information in the middle. For example changing a request "withdraw $5000" to "withdraw $50000" or someone could also try to change the source / destination account number in case of a money transaction from one account to another. To ensure that the information reaches the other end without any modification which is called integrity we have used hash codes. We have chosen MD5 (hash code function) for this purpose. The data will be encrypted and the hash code of the original message will be attached by the client application. The data decryption and hash code comparison will then take place once the message (request) reaches the destination server. Moreover to avoid the problems that could occur from stolen keys, a new key would be generated and passed to the destination server after every 100^{th} transaction.

Therefore the basic threats that we were able to identify and decided to handle immediately are confidentiality and integrity.

A. Use of Triple DES for Encryption

Organizations' growing security concerns and the increasing adoption of the Internet for business-to-business (B2B) transactions are increasing the need and importance of encryption and confidentiality. The Data Encryption Standard (DES) was developed by an IBM team around 1974 and adopted as a national standard in 1977. Triple DES is a minor variation of this standard. It is three times slower than DES but can be much more secure if used properly. Triple DES enjoys much wider use than DES because DES is so easy to break with today's rapidly advancing technology. This is major reason we decided to use Triple DES instead of DES. Here is a brief over view of how Triple DES works; Triple DES is simply another mode of DES operation.

It takes three 64-bit keys, for an overall key length of 192 bits. In Private Encryptor, you simply type in the entire 192-bit (24 character) key rather than entering each of the three keys individually [10].

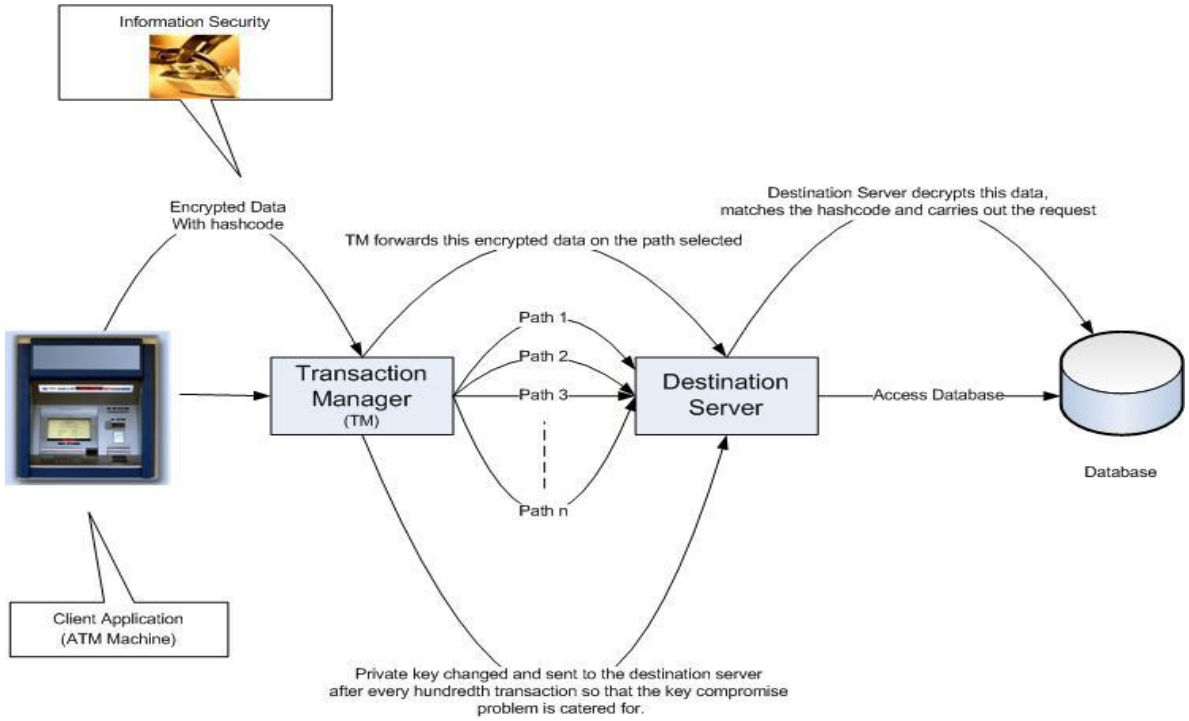

Fig. 2 A Fault Tolerance Infrastructure for Mobile Agents (FTIMA) Modified Architecture to provide Security

The Triple DES DLL then breaks the user provided key into three sub keys, padding the keys if necessary so they are each 64 bits long. The procedure for encryption is exactly the same as regular DES, but it is repeated three times hence named Triple DES. The data is encrypted with the first key, decrypted with the second key, and finally encrypted again with the third key as shown in Fig. 3.

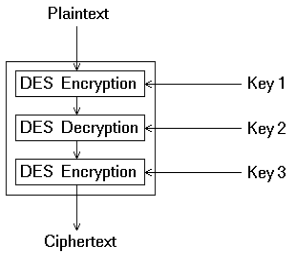

Fig. 3 Triple DES Encryption

B. Use of MD5 for Hash Code

MD5 is a message-digest algorithm developed by Rivest. It is used where a large message has to be "compressed" in a secure manner before being signed with the private key. It takes a message of arbitrary length and produce a 128-bit message digest. It is used to ensure that the information being sent has not been tampered with and thus helps to ensure integrity.

MD5 processes a variable length message into a fixed-length output of 128 bits. The input message is broken up into chunks of 512-bit blocks; the message is padded so that its length is divisible by 512.

The padding works as follows: first a single bit, 1, is appended to the end of the message. This is followed by as many zeros as are required to bring the length of the message up to 64 bits fewer than a multiple of 512. The remaining bits are filled up with a 64-bit integer representing the length of the original message.

The main MD5 algorithm operates on a 128-bit state, divided into four 32-bit words, say *A*, *B*, *C* and *D*. These are initialized to certain fixed constants. The main algorithm then operates on each 512-bit message block in turn, each block modifying the state. The processing of a message block consists of four similar stages, termed *rounds*; each round is composed of 16 similar operations based on a non-linear function *F*, modular addition, and left rotation.

Therefore in short the MD5 (Message Digest number 5) checksum for a file is a unique 128-bit 'fingerprint'. It can be used to verify that the file has not been tampered with. Security conscious users for example in the case of financial transactions who are concerned that their request may have been tampered with should use this checksum to validate integrity and hence we chose to use this message digest.

IV. Test Cases

In order to check the reliability and security of the architecture discussed in section 3, we conducted a few test cases. Two separate test cases were designed to ensure integrity and confidentiality. The test cases have been made taking scenarios from the real life banking ATM machine

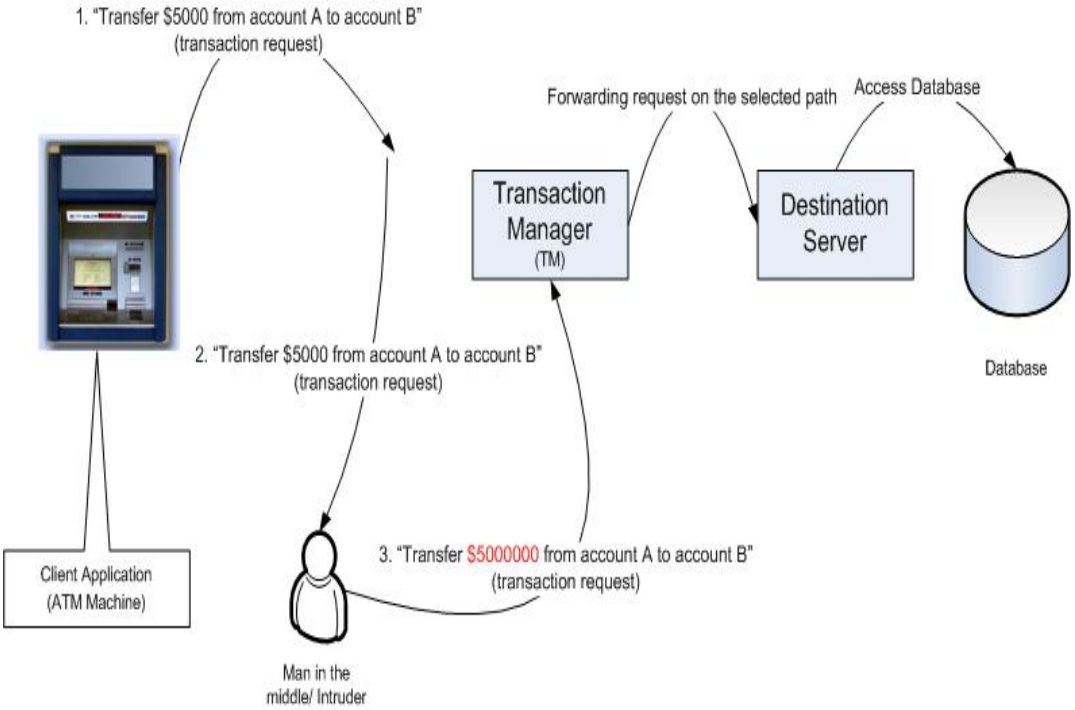

Fig. 4 Threat to Integrity of Data

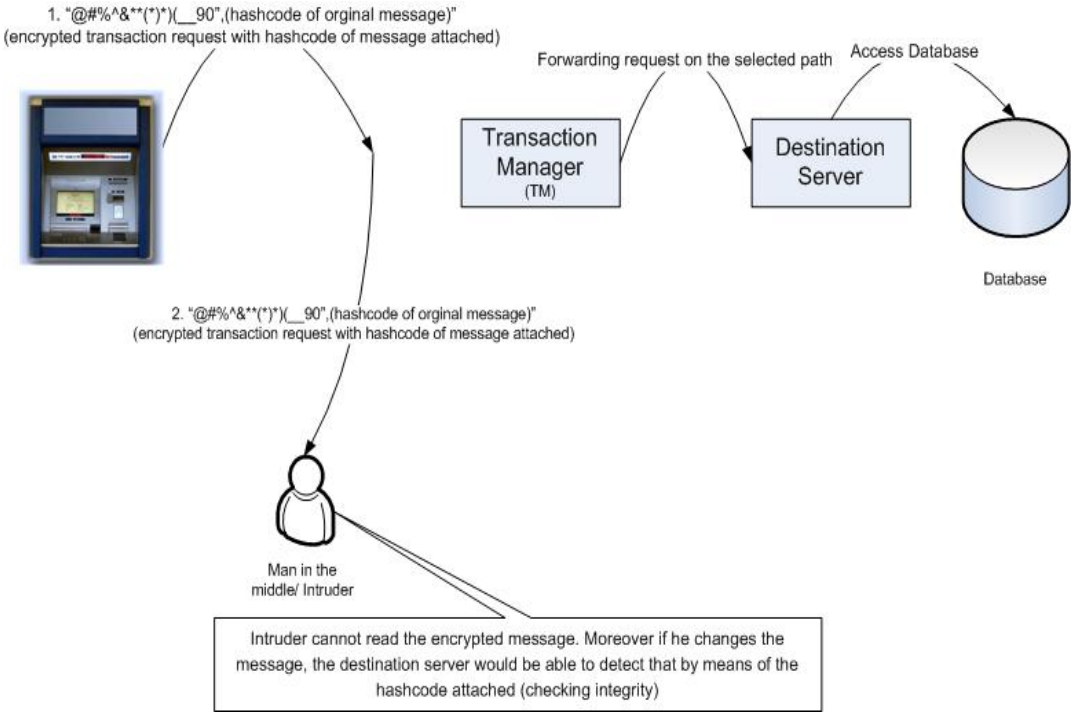

Fig. 5 Ensuring Integrity of Data

transactions. Each test case first discusses the threat identified for each case and then shows with the help of a diagram how that threat is removed to satisfy the user as well as the management of bank.

A. Test Case 1

In an ATM banking scenario let's suppose that a transaction request to transfer $5000 of a client from one account (A)

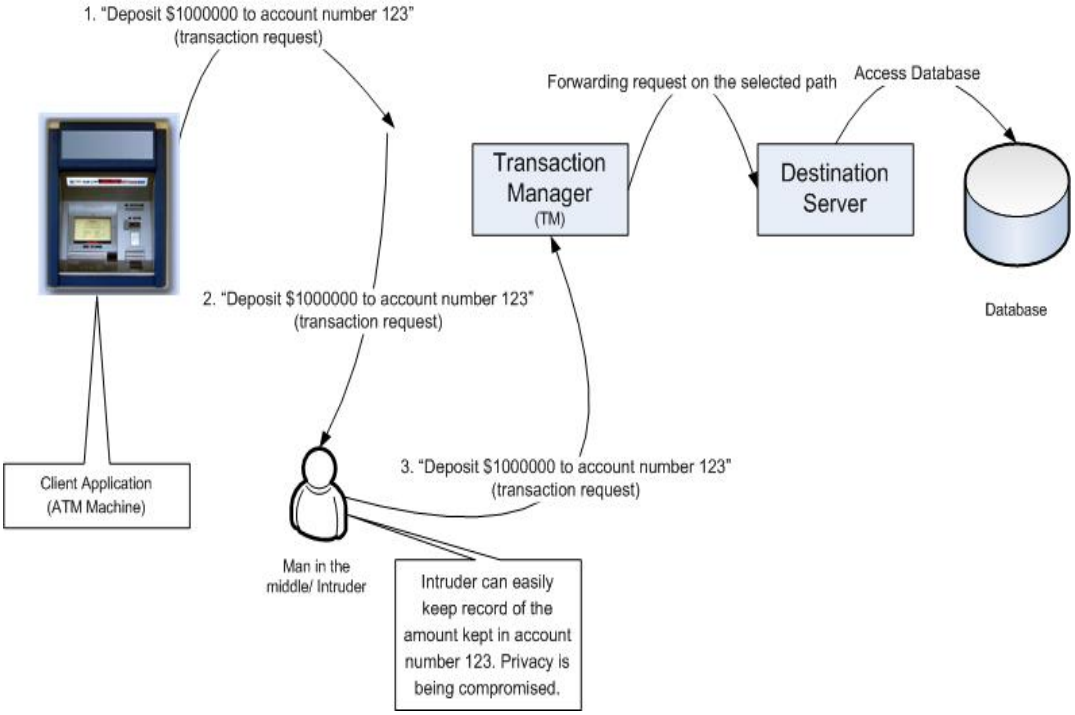

Fig. 6 Threat to Confidentiality

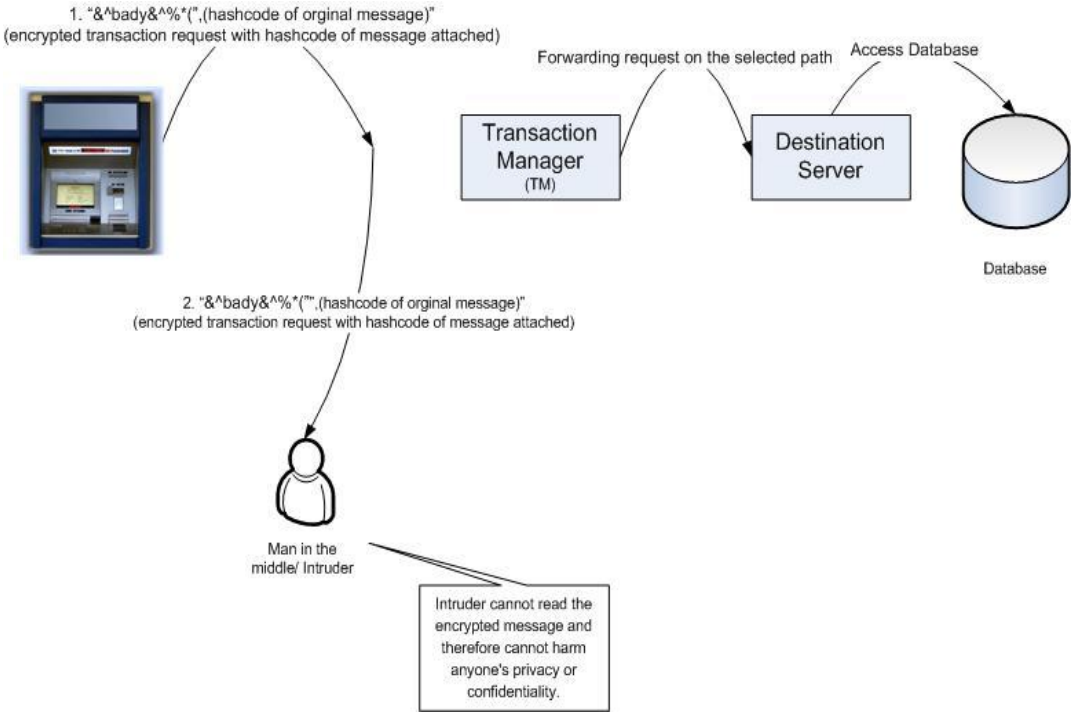

Fig. 7 Ensuring Confidently

to another account (B) is on its way as shown in Fig. 4.

If the data is not encrypted an intruder could easily read this data and modify it to transfer $5000000 instead of $5000 without the knowledge of the client. Therefore the client would be at loss and would not even be able to find out until he checks his account again at a later stage.

These kinds of losses are not affordable in critical transactions such as financial transaction requests and definitely require security. In order to avoid this threat, if the data is encrypted as shown in Fig. 5, the man in the middle (intruder) firstly would not be able to read the data and moreover if he makes any change randomly, this would be detected with the help of the hash code attached.

The destination server after decrypting the message would calculate its hash code and when it compares this hash code with the hash code attached, it would be found out that the data has been tempered and is not in its original form.

B. Test Case 2

In this case an intruder could easily sniff request packets and keep track of the amount being deposited, withdrawn or transferred from/to account number 123 which is clearly a violation of confidentiality of information as shown in Fig. 6.

Nobody deserves to be tracked about what amount goes into his account and what amount is deposited neither should his account number be known by anyone just like that. This is no doubt confidential information and requires privacy.

In order to avoid this kind of situation sending encrypted messages would be useful as shown in Fig. 7. The intruder would not simply be able to read any request messages and therefore can not track what amount is being a deposited, withdrawn or transferred at all time.

Moreover, the clients account number being personal data would also be saved from any leakage. Therefore we see how encryption serves the purpose of securing private information which is the right of every user especially in the case of financial transactions as mentioned in the scenarios discussed.

V. Conclusion

In this paper we have modified FTIMA architecture to ensure that the user request reaches the destination server securely and without any change. We have used triple DES for encryption/decryption and MD5 algorithm for validity of message. Existing Fault Tolerance Infrastructure for Mobile Agents (FTIMA) provides a fault tolerant behavior in distributed transactions and uses multi-agent system for distributed transaction and processing but data flows through the network with out encryption and contains personal, private or confidential information. In banking transactions a minor change in the transaction can cause a great loss to the user. We have evaluated the modified architecture on larger number of test cases and conclude that using proposed modification we can protect the transactions as well as personal information.

References

[1] Summiya, Kiran Ijaz, Umar Manzoor, Arshad Ali "A Fault Tolerance Infrastructure for Mobile Agents" IEEE Intelligent Agents,Web Technologies and Internet Commerce (IAWTIC 06) Sydney, Australia, 29 Nov – 01 Dec, 2006.

[2] Hartmut Vogler, Thomas Kunkelmann, Marie-Louise Moschgath, *"Distributed Transaction Processing as a Reliability Concept for Mobile Agents,"* ftdcs, 6th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS '97), 1997.

[3] Philippa Broadfoot, Gavin Lowe, *"On Distributed Security Transactions that Use Secure Transport Protocols" csfw*, 16th IEEE Computer Security Foundations Workshop (CSFW'03), 2003.

[4] Makan Pourzandi, David Gordon, William Yurcik, Gregory A. Koenig, , *"]Clusters and Security: Distributed Security for Distributed Systems*", Open Systems Laboratory, Ericsson Research,(NCSA).

[5] Gerhard Weiss, *"A Modern Approach to Distributed Artificial Intelligence"*,Chapters: 1-4, the MIT Press Cambridge, Massachusetts London, England, 1999.

[6] Chris Mayers, *"ANSAwise - Transactions in Distributed Systems"*, a variant of APM.1461, produced for CNET, 1st April 1996.

[7] Michael R. Lyu, Xinyu Chen, and Tsz Yeung Wong, Research Paper, *"Design and Evaluation of a Fault Tolerant Mobile Agent System"*, Chinese University of Honk Kong.

[8] Andrew S. Tanenbaum, Maarten van Steen, Book, *"Distributed Systems, Principles and Paradigms"* Chapters 1, 5 and 7.

[9] Mark Greaves, Victoria Stavridou-Colemen, Robert Laddaga, *"Dependable Agent Systems"*, 2004.

[10] http://www.tropsoft.com/strongenc/des3.htm

[11] http://en.wikipedia.org/wiki/MD5

[12] William Stallings, Book, *"Cryptography and Network Security, Principles and Practices", Third Edition.*

[13] http://www.informit.com/bookstore/product.asp?isbn=0131829084&rl=1