Research Journal of Information Technology

ISSN 1815-7432

Academic Journals Inc.

www.academicjournals.com

Research Journal of Information Technology 2 (3): 174-184, 2010
ISSN 1815-7432 / DOI: 10.3923/rjit.2009.30.40

Improving the Performance of the Authorization Process of a Credit Card System Using Thread-Level Parallelism and Singleton Pattern*

S.H.Ab. Hamid, M.H. Nizam Md. Nasir, W.Y. Ming and H. Hassan
Department of Software Engineering,
Faculty of Computer Science and Information Technology,
University of Malaya, 50603 Kuala Lumpur, Malaysia

Abstract: This study presents an architectural framework and prototype of a credit card authorization system using multi-threading and shared memory pool techniques in order to improve the response time during the authorization process. Through the multi-threading technique, each worker thread will be assigned several child threads to perform online fraud validation concurrently, depending on the numbers of cryptographic elements presented in the transaction message. Meanwhile, the worker thread itself performs the card restriction validation based on the information stored in the card's shared memory pool. This approach was implemented to reduce the idle time while waiting for the slow cryptographic operation in each input/output operation that is performed through an external device and to accelerate the authorization process through a memory operation instead of accessing similar information from a database. The implementation of these techniques was then measured in terms of response time. The results showed that the performance of the multi-threaded authentication engine was almost double that of the single-threaded authentication engine and the number of credit card authorizations that can be processed using the shared memory was 10% higher than the number of credit card authorizations that can be processed using a database at a single point.

Key words: Performance, .NET, single-threaded, singleton design pattern, multi-threaded engine

INTRODUCTION

Credit card authorization is a process whereby the card issuer decides whether to approve or decline requests to accept transactions performed by cardholders, which is based on a series of validations of the card's risk management profile to verify that the cardholder's account is open, the transaction amount is within the available credit limit and comes from the legitimate card and many other related validation parameters. The validation of the card's risk management profile can be classified into two categories, namely card restriction validation and online fraud validation.

Card restriction validation includes the financial and non-financial verification related to the card whereas online fraud validation involves cryptographic operation through a Host Security Module (HSM) to verify the security aspect of the authorization in order to determine the legitimacy of the card. An HSM is an external device connected to the authorization host that keeps the card issuer's secret information in tamper-resistant hardware, which is used to perform the verification of the credit card transaction. Due to various validations during the authorization process for each credit card transaction, it will take some time to complete a whole process. Moreover, a slow and expensive input

Corresponding Author: Mohd Hairul Nizam Md. Nasir, Department of Software Engineering, Faculty of Computer Science and Information Technology, University of Malaya (UM), 50603, Kuala Lumpur, Malaysia Tel: +603-79676435 Fax: +603-21784965

Originally Published in Research Journal of Information Technology, 2009

and output operation during the card restriction and online fraud validation through the database and HSM also causes some delays. Besides, validation in terms of the security aspect of the card itself also takes some time to process due to the complexity of the algorithms involved. As a result, the performance is affected whenever the number of authorization processes is increased. In this case, it will cause some of the simultaneous authorizations accepted at a single point in time not to be able to respond within the allowed timeframe. These transactions failures are classified as timed-out in the context of electronic financial services.

This research will look into the current issues relating to the credit card authorization process. It concludes that a multi-threaded authorization system with a shared memory pool is needed in order to improve the response time of the credit card authorization process and to overcome the slow sequential authorization processing problem of a single-threaded model for the current credit card authorization system. The prototype of the multi-threaded authorization system was developed using the .NET framework, then the performance of the multi-threading implementation was measured.

There are various methods proposed in order to improve the response time of the authorization process of a credit card transaction. These include the invention of a Host Security Module (HSM), implementation of a distributed authorization system, utilization of a cardholder-initiated transactions device and deployment of a digital network access system device.

An HSM is an external device which is used to generate and store long-term secrets securely for use in cryptography and physically protect the access to and use of those secrets over time. According to Chodowiec and Gaj (2003), these secrets include the private keys used in symmetric key protection and public key cryptography. The HSM is implemented because the hardware implementation is the only way that can achieve speeds beyond the reach of general-purpose microprocessors. Therefore, the HSM is used as a cryptographic accelerator to hasten the intensity of the mathematical operation, especially in public key encryption and provide better performance than a normal software-based cryptographic system as discussed by Eslami *et al*. (2006). The functionalities of the HSM include the verification of an online Personal Identification Number (PIN) by comparing it with an encrypted PIN block, the validation of credit card transactions by checking card security codes and performing the host processing component of a Europay Master Card Visa (EMV) based transaction. In recent years, the introduction of an HSM that supports an Ethernet device is gaining in popularity because of its higher speed of data transmission during cryptographic processing. The simulation result performed by students from one of the Universities in Brazil proves that the IP performance version provides a better performance than other protocol solutions as mentioned by Panato *et al*. (2002). In short, the HSM provides an industry-leading performance in which it significantly reduces the credit card transaction processing time and lowers the cost per transaction. The only problem with the HSM is that there is apparently no global standard in the low-level communication data exchange protocol due to the re-engineering cost and market dominancy.

A patented method of a distributed authorization system has been proposed in the last decade to accelerate the authorization process. This system utilizes a host computer communicating with a network of remote electronic terminals from the host computer. It includes storing negative file data in the electronic terminal containing information used to identify accounts for which requested transactions are to be denied and storing authorization file data in order to determine the authorization of a requested transaction. The completed transaction is stored in a terminal transaction queue file residing in the terminal for subsequent transmission to the host computer and for use with a transaction request which is subsequently entered at the terminal for the same account. However, with the increasing number of terminals and credit cards, it will increase the network traffic and it is costly to maintain this information at the network level. Moreover, the card issuer has less control over the authorization profile. This would result in some information not being updated instantly in the network and may cause bad credit accounts. Besides that, there is a higher potential risk of fraudulent cases that would cause financial loss in the event of a lost card.

Then, the cardholder-initiated transaction device has been introduced. This approach allows end-user cardholders, by means of their own card devices, to authenticate POS terminal devices in a way substantially different from the existing EMV protocol. The EMV protocol is often used for authenticating user transmissions to Point-Of-Sales (POS) terminal devices. By contrast, the invention performs authentication of the parties to a prospective transaction at the same time as it transfers the message data necessary to carry out the authorization of the transaction through the POS terminal device. If both of the authentications are successful, the exchanged authentication data and transactions data sent between the devices would be used to complete the transaction. Through this technique, the authentication of the card and terminal would greatly reduce the time required to perform the transaction. This approach claimed to reduce by four times the usual time to complete an electronic transaction, which averages 15 to 30 sec.

The common emphases of the authorization process of credit card transactions are on performance and security. The performance aspect concerns the time to authorize and complete a sales transaction whereas the security aspect concerns the fraud prevention and confidentiality of the financial information. The three major factors affecting the efficiency of the authorization process are the transaction volume, the automatic authorization procedure and the authorizers as discussed by Leung and Lai (2001). With an increasing number of accounts and transaction volume, these two aspects remain a major dilemma of the credit card authorization process. Current research in recent years has focused on the security area of the authorization process of credit cards. This is because the number of fraudulent cases is growing dramatically and it is becoming a serious problem faced by credit card issuers. In 2004, credit card transactions had a total loss of 800 million dollars of fraud in the United States, while in the United Kingdom, the loss due to credit card fraud amounted to 425 million pounds, as discussed by Shen *et al*. (2007). Fraud affects all sectors of the community, extending from individuals who have responded to online offers of making quick money (Smith, 2008).

Various fraud detection techniques have been proposed to combat fraudulent cases, such as using smart cards and implementing a fraud detection system using data mining techniques including neural networks, logistic regression and decision trees. However, increasing the security aspect will bring a downside to performance when it is implemented using more advanced technology (Hwang and Verbauwhede, 2004). This is the trade-off of the authorization process when it is implemented using an advanced technique like smart cards due to the higher transmission of bytes to the server and a longer processing time to perform verification. According to local news published in Motor Traders, the Managing Director of ProJET Malaysia, Matthew Selbie, has mentioned that chip-based transactions will take a second or two longer than the usual magnetic stripe transactions to complete the verification after deployment of the new devices to accept chip-based transactions in petrol stations. Besides that, the implementation of advanced risk analysis techniques using the computer intellectual will also contribute to the processing time, which may result in performance degradation. Apart from that, the size of the database to manage the authentication data is also increasing enormously with the usage of more advanced technology such as smart cards. For example, Juang *et al*. (2008) have proposed a robust and efficient user authentication and key agreement scheme using smart cards. According to Bourlai *et al*. (2005), the use of a fixed precision data type does not affect system performance very much but can speed up the verification process. As a result, the verification performance decreases monotonically and appears to saturate when the database size increases similarly to that mentioned by Bourlai *et al*. (2006).

According to Bank Negara Malaysia's (BNM) Annual Report (2004), the number of credit cards in circulation in Malaysia reached a total of 6.6 million at the end of 2004, with total transactions amounting to RM34.9 billion. Also, in recent years, there has been a dramatic growth in credit card usage among college students. It can be seen that credit card usage is not only restricted to elite groups but is spreading among the graduates.

The credit card authorization systems that most banks are using are more than 15 years old, hard-coded, rigid and time-consuming to change. Furthermore, many of these systems are at capacity and struggling to keep up with the large increase in card payment volume. Many systems lack embedded business rules or workflow engines, resulting in, among other things, inefficient risk management operations. As a consequence, some of the transactions do not have the chance to be processed using the conventional architecture design during high simultaneous transaction flow.

According to Tim Kelly, director of TSYS, the transaction delays in the COBOL-based programs running on the mainframe have affected their business tremendously when the transaction flow is high. To cater for this scenario, some of the banks have begun to upgrade the existing card processor applications to a new enhanced processing platform. For instance, one of the largest banks in Germany, VÖB-ZVD Bank, has appointed Atos Origin to implement its new authorization solution named Worldline Pay. With the implementation of the new solution, VÖB-ZVD Bank hopes to achieve a high performance authorization platform that can reliably handle all payment transactions. The Managing Director of the VÖB-ZVD Bank, Gabriele Cremer-Wichelhaus, has stated that new requirements and the constantly increasing number of transactions in card and Internet-based payments require precocious system adaptations which would enable the bank to meet the demands of the market and the clients and to handle the future number of transactions.

Many banks are using home-grown authorization of credit card systems that are more than 15 years old and in need of functional and technical upgrades. Other banks are using packaged applications that still need upgrades as well. In either case, the card authorization systems that most banks have in place are rigid, at capacity in terms of account and transaction volume and difficult to change in the face of changing regulations and market conditions. Currently, there are a few big market players in providing authorization system solutions to the credit card companies. Most of these authorization systems are parameter-driven in order to give flexibility to the authorization process and meet the demand of the market. However, there is still room for improvement as mentioned in the latest industry survey report on the payment solution to cater for the payment transaction volume.

An analysis of the current authorization systems solution in the market has been conducted and the findings of this analysis show that the current credit card authorization system does not utilize the multi-threading technique as part of its architecture design. Most of the systems are using Oracle as their database management system and none of the current credit card authorization systems use the shared memory pool for authorization purposes. Apart from that, advanced programming languages such as .NET, for example, are not the most commonly used in the current architecture of credit card authorization systems.

In this case, performance still remains an issue that requires improvement with the increasing number of transactions and the implementation of greater security features. Moreover, there are many home-grown credit card authorization systems still using old technologies to perform authorization that can not support high transaction flow. Therefore, multi-threading should be deployed as one of the techniques to improve the response time of the credit card authorization process since modern operating systems with advanced multi-core processors have good support of multi-threading implementation.

MATERIALS AND METHODS

The prototype of the credit card authorization system was developed in January 2008 at the Department of Software Engineering, Faculty of Computer Science and Information Technology, University of Malaya, using .NET programming language in order to measure the performance of the authorization process. The functionalities of the prototype authorization credit card system are categorized into two main broad components, namely the front engine component and the back office component.

The front engine component is the authentication engine of the credit card authorization system. This component consists of four modules, namely the listener module, worker thread module, authorization module and shared memory module. The listener module contains functionalities that include activating the listener service, activating the worker thread-pool, activating the child thread-pool, activating the shared memory pool and accepting the socket connection. The worker thread module contains functionalities that include handling the socket connection, parsing the authorization message, displaying the authorization message, updating the authorization message, building the authorization message, saving the authorization message, updating the card balance, saving the card changes and closing the socket connection. The authorization module contains functionalities related to card restriction validation and online fraud validation. Card restriction validation consists of checking the card's existence, checking the card status, checking the card activation status, checking the card expiration date, checking the card usage and checking the card balance whereas online fraud validation consists of checking the card security code, checking the card identification number, checking the personal identification number and checking the chip application cryptogram. The functionalities of online fraud validation are performed through child threads. The shared memory module contains functionalities that include activating the synchronization service, searching the modified card information and updating the card information.

On the other hand, the back office component stores the authentication data used in the authorization of the credit card system. This component consists of user management and card management. The functionalities related to the user management include displaying the user information, saving the user information and validating the user information, whereas card management consists of displaying the card information, displaying the card activity, displaying the card history, saving the card information, updating the card information, searching the card information and saving the card changes.

Architectural Design

As shown in Fig. 1, the architectural design of the authorization of the credit card system consists of a front engine and a back office. These two components will interact with the system database to store and retrieve application-related data. Apart from the system main components, there are a few sub-systems that have communication with the authorization of the credit card system including the Host Security Module (HSM) server, Point-Of-Sale (POS) server, Automated Teller Machine (ATM) server and electronic commerce (e-commerce) server.

All these sub-systems will communicate with the authorization of the credit card through TCP/IP protocol. The message format that is used for communication between the authorization system and the HSM server is specific proprietary command, whereas for the other sub-systems, the message format that is used to communicate with the authorization system is ISO 8583. ISO 8583 is the standard interchange message specification defined by the International Organization for Standardization (ISO) for electronic transactions made by cardholders using payment cards.

The multi-threading technique is adopted into the architectural design of the authorization engine of the credit card system. Through this technique, multiple threads can be run simultaneously within the single memory space of the process and all the threads share the same system resources during the authorization process of the credit card transaction. In this study, the thread-pool model is used to handle the concurrent authorization requests from the payment gateway and the shared memory pool is implemented in conjunction with the multi-threading technique to hasten the authorization processing. The shared memory pool is implemented in this study to reduce the time searching for card information from the system database, which involves an expensive I/O operation as compared with obtaining the similar information through a shared memory pool stored in random access memory using a binary search. There are two thread-pools implemented in the system, namely the worker

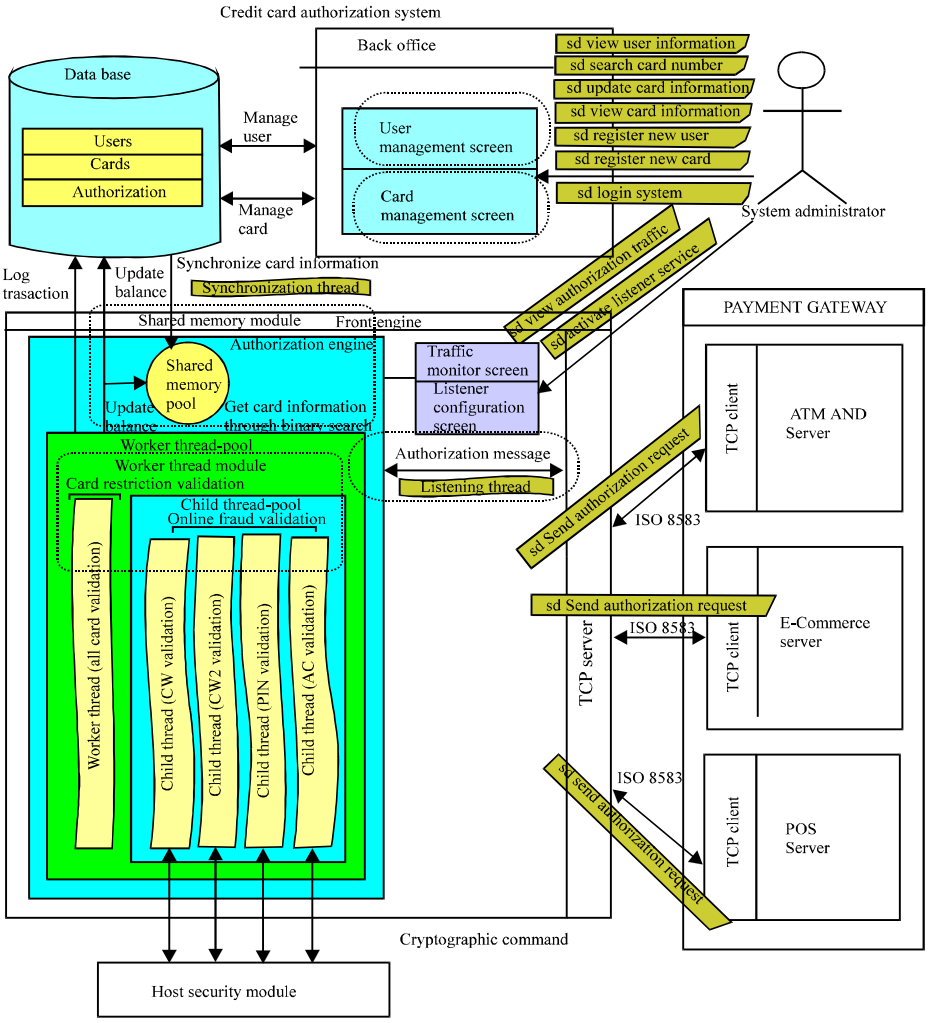

Fig. 1: Overall system architecture

thread-pool and the child thread-pool. When the listener service is activated, all the worker threads and child threads are constructed and started in their related thread-pools through the listening thread. Besides that, all the card information is also loaded to the shared memory pool before the authorization request can be serviced.

The worker threads in the pool are combined with a work queue. Each accepted client socket through the listening thread from the payment gateway will be put in the work queue. The work queue will signal the waiting worker threads each time a new authorization job arrives to make the relative waiting threads process the authorization request immediately. Each authorization job is mapped to a client connection. The assigned worker thread acquires a socket from the queue and serves the request on that socket until the connection is closed. Once the authorization job is accepted, the worker thread will acquire a mutex lock not only to synchronize the access to the shared data area but also to accelerate the processing in the thread-pool environment. To avoid a starvation situation, a timer has been set to release the mutex after a pre-defined period has elapsed.

The worker thread assigned to each authorization process of the credit card transaction will begin to read the raw buffer message in ISO 8583 format accepted from the socket connection and proceed with message parsing to obtain all the elements. Once the message is parsed, the worker thread will perform the card restriction validation and online fraud validation based on the elements present in the message. The worker thread begins to assign several child threads to perform cryptographic operations in online fraud validation and the number of child threads assigned for online fraud validation is in accordance with the number of cryptographic elements present in the credit card transaction itself. Similar to the worker threads, the child threads in the pool are also combined with a child queue. Each assignment of a child thread is put in the child queue and the child queue will signal available waiting child threads each time a cryptographic task is added. The assigned child thread will remove the cryptographic task and proceed with its validation through HSM. These cryptographic operations encompass card security code validation, card identification number validation, personal identification number validation and chip application cryptogram validation.

Once all the child threads have been assigned to these cryptographic operations, the worker thread itself will perform an operation pertaining to the card restriction validation. This operation is done in parallel with the child threads handling the cryptographic processing. The card restriction validation includes card existence validation, card status validation, card activation status validation, card expiration date validation, card usage validation and card balance validation. All the operations related to card restriction validation are carried out through the shared memory pool without accessing the system database.

Once the worker thread finishes its card restriction operation, it waits for a completion signal from the child threads that perform the online fraud validation. Upon receiving all the completion signals from the child threads, all the assigned child threads are put back into the child thread-pool for the next assignment while the worker thread will be working on providing the final response code to the cardholder on whether to approve or decline the transaction based on the result of the entire validation. If there were any declines during the validation, the final response code will be based on the first occurrence of the decline. Otherwise, the transaction will be approved and a unique authorization number will be randomly generated as part of the authorization response message that will be used as a reference. Next, the assigned worker thread will proceed with building the authorization response message in ISO 8583 format. Once the response message has been built, the worker thread will write the message to the socket and this authorization response will be sent back to the payment gateway that originated the transaction.

After the authorization response has been sent, the assigned worker thread will drop the socket connection and proceed with the internal processing. This internal processing includes saving the authorization message into an authorizations table for record purposes and performing balance updating for the particular card. The balance adjustment will be updated in both the shared memory pool and the system database. Next, the acquired mutex is released and the pending timer set earlier is cancelled before the worker thread is put back into the worker thread-pool for its next assignment. In this project, an additional synchronization thread is started against the background of the authorization engine to update any changed information of the card made through the back office component in the shared memory pool. This is implemented to ensure the data kept in the shared memory is always synchronous with similar information stored in the system database.

In a single-threaded credit card authorization system, both card restriction validation and online fraud validation have to be done one after another. Thus, the system resources are not fully optimized because the waiting time of the slow I/O operation, especially during the validation of the cryptographic element, is wasted. This not only causes the authorization of the credit card transaction to take a longer time to process but it also degrades the performance of the server, especially during the heavy traffic during peak hours. In that case, the cardholder might encounter a problem in gaining authorization from the system because of the slow response time from the credit card authorization system.

In this study, the multi-threaded credit card authorization system is used to accelerate the authorization process. Multiple tasks of the authorization process can be executed concurrently through multiple threads. If there were two or more cryptographic operations to be performed during the authorization process, the ideal time of the waiting I/O operation could be reduced to at least half of the total time required in processing those operations sequentially. Apart from time, the thread-pool model is applied to minimize the system resources spent in creating and destroying this type of recyclable thread.

Besides that, the response time could be further reduced by loading all the cards' information into random access memory to let the authorization system obtain information from the shared memory pool through a binary search instead of accessing similar data from the database for authorization processing. Through all these methods, the response time of the credit card authorization process could be significantly improved.

In this study, a singleton design pattern was applied to the card object which is acting as a shared memory pool that holds all the cards' information for authorization purposes. When the listener service is activated, the listening thread will load all the information of the cards into random access memory through a configurable array. After an authorization is received, a worker thread will obtain the only instance of the card's object and perform a binary search through the related array of the card's object in order to retrieve the information of the card related to the transaction from the shared memory for authorization purposes. A separate synchronization thread is initialized in the background of the authorization engine to browse the system database for any modified card information requiring updating in the shared memory pool. This is implemented to ensure the data kept in the database is synchronized with the data in the shared memory pool. Once modified, the card information is loaded into the shared memory pool and the synchronization thread will update the system database to mark that the card has been processed.

A singleton design pattern is applied to ensure all the workers threads can access the shared memory pool for card information during authorization. Without a singleton design pattern, shared memory pool implementation is not possible in an object-oriented environment. Through the shared memory pool, the access time is faster and, hence, improves the response time of the credit card authorization process.

RESULTS AND DISCUSSION

Timing testing has been used to evaluate the response time of the authorization process under different circumstances. The response time was measured using the embedded testing tools that were built in as part of both the authorization system and the payment gateway to obtain the time taken before and after the transaction was sent and received. The measurement unit for the response time was recorded in seconds.

In this study, the response time was evaluated from two major aspects. These aspects are the authorization system using the multi-threaded authentication engine against the authorization system using the single-threaded authentication engine and the multi-threaded authentication engine accessing the shared memory pool for authentication data against the multi-threaded authentication engine accessing the system database for authentication data. For both aspects, an incremental testing approach has been chosen as the technique to obtain the result.

In the comparison between the multi-threaded authentication engine and the single-threaded authentication engine, incremental testing was performed to evaluate the response time of a group of authorizations performed sequentially. For this evaluation, there is no simultaneous authorization

Table 1: Test result of multi-threaded and single-threaded authentication engines

No. of sequential authorizations	Multi-threaded authentication engine (sec)	Single-threaded authentication engine (sec)
10	4.7	8.1
20	9.4	16.5
30	14.2	24.8
40	18.8	33.0
50	23.4	41.1
60	28.2	49.4
70	32.7	57.7
80	37.6	65.9
90	42.1	74.7
100	46.9	82.3

Table 2: Test result of authentication engine using shared memory and database

No. of sequential authorizations	Authentication data via shared memory (sec)	Authentication data via database (sec)
10	04.1	04.4
20	08.1	08.9
30	12.1	13.4
40	16.4	17.7
50	20.3	22.0
60	24.5	26.7
70	28.7	31.4
80	32.8	36.2
90	36.9	41.0
100	40.5	45.9

performed. The next authorization will be sent upon receiving a response from the previous transaction. The number of worker threads and child threads that were used in the multi-threaded authorization system is 3 and 9, respectively. In this testing, the result is recorded based on the best response time taken in 5 attempts for each category. This is done to minimize the impact of the context switching between multiple threads running in the system over the result obtained and to ensure the accuracy of the testing performed.

Based on the test result as shown in Table 1, it has been confirmed that the performance of the multi-threaded authentication engine is better than the single-threaded authentication engine in the .NET platform. From the result, the performance of the multi-threaded authentication engine is almost double that of the single-threaded authentication engine.

In the second aspect, the testing was carried out to assess the response time of a group of authorizations performed one after another using the multi-threaded authentication engine accessing the shared memory pool for authentication data and the multi-threaded authentication engine accessing the system database for authentication data. Similar to the first aspect, the next authorization will be sent upon receiving a response from the previous transaction and there is no simultaneous authorization performed. The number of worker threads and child threads that were used in the multi-threaded authorization of the credit card system is also similar, which is 3 and 9, respectively. The test result is recorded based on the best response time taken in 5 attempts for each category. This is done to minimize the impact of the context switching between multiple threads running in the system over the result obtained and to ensure the accuracy of the testing performed.

Based on the test result as shown in Table 2, the performance of the multi-threaded authentication engine using the shared memory for authentication data is better than the multi-threaded authentication engine using the database for authentication data in the .NET platform. The difference is insignificant at the earlier stage, but it becomes more significant when the number of authorizations increases. From the test result, the number of credit card authorizations that can be processed using the shared memory is 10% more than the number of credit card authorizations that can be processed using the database at a single point in time.

CONCLUSION

This research provides a solution to optimize the performance of credit card authorization systems through the multi-threading technique in the .NET platform. This technique enables the authorization of credit card transactions to be processed in a shorter length of time. From the business point of view, a fast and reliable authorization process will generate more revenue to the organization, whereas from the customer point of view, the authorization process in time builds the confidence of the cardholder to use the credit card as a payment method. In short, this study provides a win-win situation for both organization and community as both parties will gain the benefits from the implementation of the multi-threaded authorization of credit card systems.

Besides that, the multi-threaded authorization of the credit card system implemented in this study enables several tasks related to the card's risk management profile validation to be executed concurrently during the authorization process. This will not only provide a better response time for the authorization process but also enables more credit card transactions to be processed in a multi-threaded authorization system in a shorter amount of time. A shared memory pool is also used in conjunction with the multi-threading technique. Since multiple threads are running in a single process space, the shared memory pool is implemented to keep all the card information that will be used for the credit card authorization process in the random access memory area. Besides that, the multi-threaded architectural design presented in this study supports the dynamic tuning of the size of the thread-pool running at runtime. The number of fixed worker threads and child threads can be adjusted to ensure the utilization of the multiple threads to its optimal level. This is implemented to ensure the capacity of the thread-pool matches the necessities of the application based on the estimated volume and velocity of the credit card transactions processed in a specified period. Besides, the multi-threaded credit card authorization systems implemented in this study can accept multiple connections from the payment system at a single port number. This is implemented to allow more simultaneous authorizations to be received through these multiple links for load balancing usage in the future.

ACKNOWLEDGMENTS

First and foremost, we would like to express our gratitude to Almighty hat gave us the possibility to complete the research work successfully. Secondly, we would like to forward our deepest thanks to our colleagues, lecturers and technical staff from the Department of Software Engineering for their endless assistance, technical advice and co-operation.

REFERENCES

BNM, 2004. Annual Report, 2004. http://www.bnm.gov.my/files/publication/ar/en/2004/ar2004.complete.pdf.

Bourlai, T., J. Kittler and K. Messer, 2005. Scenario based performance optimisation in face verification using smart cards. Audio and Video Based Biometric Person Authentication, 3546: 289-300.

Bourlai, T., J. Kittler and K. Messer, 2006. Database size effects on performance on a smart card face verification system. Proceedings of the 7th International Conference on Automatic Face and Gesture, April 10-12, Recognition, pp: 61-66.

Chodowiec, P. and K. Gaj, 2003. Very compact FPGA implementation of the aes algorithm. Lecture Notes Comput. Sci., 2779: 319-333.

Eslami, Y., A. Sheikholeslami, P.G. Gulak, S. Masui and K. Mukaida, 2006. An area-efficient universal cryptography processor for smart cards. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Jan. 2006, IEEE, pp: 43-56.

Hwang, D.D. and I. Verbauwhede, 2004. Design of portable biometric authenticators energy, performance and security tradeoffs. IEEE Trans. Consumer Elect., 50: 1222-1231.

Juang, W.S., S.T. Chen and H.T. Liaw, 2008. Robust and efficient password-authenticated key agreement using smart cards. IEEE Trans. Ind. Elect., 55: 2551-2556.

Leung, W.K. and K.K. Lai, 2001. Improving the quality of the credit card authorization process-a quantitative approach. Int. J. Service Ind. Manage., 12: 328-341.

Panato, A., M. Barcelos and R. Reis, 2002. An IP of an advanced encryption standard for altera devices. The 15th Symposium on Integrated Circuits and Systems Design, Sept. 9-14, IEEE, pp: 197-202.

Pinto, M.B., D.H. Parente and T.S. Palmer, 2001. College student performance and credit card usage. J. College Stud. Dev., 42: 49-58.

Shen, A.H., R.C. Tong and Y.C. Deng, 2007. Application of classification models on credit card fraud detection. International Conference on Service System and Service Management, June 9-11, IEEE, pp: 1-4.

Smith, R.G., 2008. Coordinating individual and organizational responses to fraud. Crime Law Soc. Change, 49: 379-396.