Journal of Advanced Research in Computing and Applications

Journal homepage: www.akademiabaru.com/arca.html
ISSN: 2462-1927

Performance Analysis of Three Classical Encryption Algorithms, Simple Substitution, Caeser, and Periodic Permutation (the Three SCP) in Encrypting Database Transactions

Open Access

Mohammed Suleiman Mohammed Rudwan[1,*], Salah Eldin Deng Al-Jack[2]

[1] Department Information Technology, College of Computer Science and Mathematics, University of Bahri, Alzohor, Mohammed Najeeb Street, Street 53, Khartoum, Sudan
[2] Department of Computer Science, College of Computer Science and Mathematics, University of Bahri, Alzohor, Mohammed Najeeb Street, Street 53, Khartoum, Sudan

ARTICLE INFO

Article history:
Received 5 March 2018
Received in revised form 12 April 2018
Accepted 10 July 2018
Available online 5 August 2018

Keywords:
Database security, database transaction, database transaction security, classical encryption

ABSTRACT

Encryption was introduced and implemented in many Database Management Systems (DBMSs) as one of the solutions for securing databases. However, Securing database transactions and data they hold while they are existing in memory or even during their transmission throuh computer networks media should be also given big attention. The core objective of this paper is that to measure the performance of the three classical encryption algorithms: Simple Substitution, Caesar cipher, and Periodic Permutation respectively for encrypting such database transactions and text data they hold which is sort of encrypting data-at-rest. Performances are measured herein in terms of time, memory usage and CPU efforts. The expirement was performed to many categories of transactions, those categories are categorized according to two issues: the size of data that the transactions hold, and the type of transactions itself. This experiment can be a base to enable Database Designers to design algorithms that has the ability to decide which one is the optimal according to type of database transaction waiting for processing as well as size of data that it consists of, and also other computing processing components such as CPUstate and RAM consumption ratios.

1. Introduction

Using Encryption in databases usually considered as the last line of defence against illegal access to them. Confidentiality is the aim in encrypting such data [1]. However, So as to guarantee the confidentiality of the in-memory database transactions as well as those are about to be transmitted through a network, classical encryption algorithms are suitable suggestion for doing that. Therefore, comparing those encryption algorithms to encrypt such transaction will be an essential issue that

[*] *Corresponding author.*
E-mail address: Mohammed Suleiman Mohammed Rudwan (m.suleiman@bahri.edu.sd)

DBMSs designers should maintain carefully in order to choose the right option in terms of time, CPU efforts, memory consumption. Many factors no-doubt affect the processing speed of the encryption algorithm, so the researchers determined certain specifications of the processing environment and tested different sizes of data and types of transactions. This paper could open eyes and minds for future researches for giving attention to design and suggest new algorithms that aim to decide which the best algorithm to be chosen from among the given options at a specific time and state – in terms of memory, speed, CPU, bandwidth, etc.

2. Overview of Database Transactions and the Encryption Algorithms under Study

2.1 Database Transaction

A transaction is "an executing program that forms a logical unit of database processing" [2]. A single database transaction can include one or more update or retrieval operations by writing Data Manipulation Language operations (DML) using high level query language such as SQL. Also, it can include Data Definition language (DDL) to manage databases, tables, columns and security policies.

2.2 Log Buffer and the System Log

Log Buffer is the area in which all the database transactions are stored in before appending them to the system log. The system log is a non-volatile file that is stored in the hard drive. It consists of all executed transactions for the sake of recovery and backup.

2.3 Simple Substitution Encryption Algorithm

This Algorithm is simply replaces each letter in the plaintext with another letter specified earlier in an array – that is shared between senders and receivers of the messages – which contains all the alphabetic letters and their corresponding letter to be substituted with as a ciphertext.

2.4 Caeser Encryption Algorithm

Ceaser algorithm is somehow same as simple substitution in that they both replaces each letter with another one, but in Caeser, each letter in the plaintext is replaced with the nth position (the order) of the letter that comes after the of that letter; where 'n' is an integer value the shared key between the senders and receivers. So, the original ordering of the alphabetics or any other custom ordering of them that is made by the parties of messages exchange should be clearly known by them. For instance, in English alphabetic if we took the original order of them, and if the key was 3, then letter 'a' in a plaintext will be transformed to 'd' as a ciphertext, 'b' will be replaced by 'e', 't' will be the ciphertext for 'q' in plaintext, and so on.

2.5 Periodic Permutation

This algorithm calculates the length for each plain text, and then a random permutation of the order of indexes (positions) of each character composing the plain text will take place until all the indexes are shuffled perfectly and completely stored in specific order. Then the index (position) of the first index generated will take place instead of the first character of the original plaintext, then the second one's value will refer to the index of the value that it holds, and so on until the last position is reached.

3. The Experiment

Here we will give a glance to the experiment; the environment, the platform, and the data sets sample in our work.

3.1 The Environment and Platform

The three algorithms, namely, Simple Substitution, Caesar, and Periodic Permutation, are programmed using java programming language. (Net beans 8.0) IDE was also used for programming the mentioned algorithms and also for measuring the time consumed for each operation.
The operating System at which the experiment was performed was Window7 Professional edition. The built-in tool in that operating system, the Task Manager, was used for measuring the memory size and CPU efforts that each algorithm took for each algorithm.The machine that the experiments were run at was a Laptop, Toshiba C50 with corei3 processor and 4 GB RAM with 2.40 GHz processor speed.

3.2 The Data Set

Data set under the experiment and study are database transactions. In order to evaluate the encryption algorithm in encrypting each transaction and for simplicity purposes as well as for defining to what extent the length of data affect the evaluation and measuring such performance accurately. Database transactions are categorized into two main parts in order to perform the experiments of our research on: the part of the transaction which identifies the transaction type as well as its id, and hereby we refer to it by 'transaction header'. The second one is the actual data that is held by the transaction and which is already had been manipulated or read from the database.

The second part of transaction is the data part. We divided it into three main categories in terms of length for the research purpose: the first one is a data length of 'VARCHAR2' data type in MySQL DBMS, that is of 65535 characters; i.e., 65.535 kilobytes. The second category of data consists of 32.768 kilobytes. And the third consists of 16.384 kilobytes.

For all three data categories we've just mentioned in the preceding paragraph, random characters had been taken using the random numbers that are generated randomly as well by the trusted scientific website: 'randomizer.org' [3]. Generated random numbers are in the range (33 - 255) and each number generated is casted into the equivalent character; that is to say, each number is considered as the ASCII code of the intended character. As it had been mentioned in just last sentence, the available characters and symbols those are possible are list in the figure1.

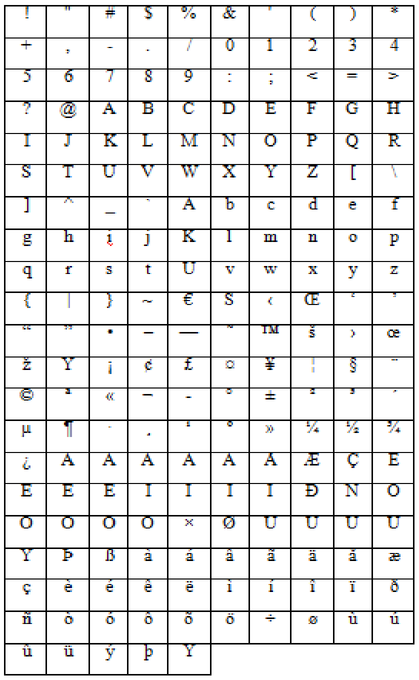
For each one of the data categories, i.e., 65.553 kilobytes, 32.768 kilobytes, and 16.384 kilobytes, a number of 5 (five) sets are prepared using the method mentioned in the preceding paragraph as they are the random sample of our data that to perform on the experiment. Each set's size will be of size of the data category to which it belongs.

4. Evaluation Methodology

4.1 Time Measurement

Generally, each transaction part (either the transaction header or even the data set) is encrypted by the encryption algorithms under study 20 (twenty) times. Then the average value for each part/set is taken as the final result for that algorithm in case of that set/part. For transaction headers, the value of time taken (in milliseconds) is determined by the following steps:

a. Determine the transaction header that going to be encrypted.
b. Determine the encryption algorithm under study.
c. Perform the encryption process 20 times and store each value (the time consumed to encrypt the transaction header) in a individual database or file.
d. Calculate the time that is taken by algorithm by getting the average of the 20 values that is adopted earlier. The average= $\frac{\sum_{i=1}^{20} time}{20}$.



!	"	#	$	%	&	'	(	)	*
+	,	-	.	/	0	1	2	3	4
5	6	7	8	9	:	;	<	=	>
?	@	A	B	C	D	E	F	G	H
I	J	K	L	M	N	O	P	Q	R
S	T	U	V	W	X	Y	Z	[	\
]	^	_	`	A	b	c	d	e	f
g	h	i	j	K	l	m	n	o	p
q	r	s	t	U	v	w	x	y	z
{	|	}	~	€	S	‹	Œ	‘	’
“	”	•	–	—	˜	TM	š	›	œ
ž	Y	¡	¢	£	¤	¥	¦	§	¨
©	ª	«	¬	-	°	±	²	³	´
µ	¶	·	¸	¹	º	»	¼	½	¾
¿	A	A	A	A	A	A	Æ	Ç	E
E	E	E	I	I	I	I	Đ	N	O
O	O	O	O	×	Ø	U	U	U	U
Y	Þ	ß	à	á	â	ã	ä	å	æ
ç	è	é	ê	ë	ì	í	î	ï	ð
ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú
û	ü	ý	þ	Y					

Fig. 1. List of Possible Characters considered in the experiment

For Data, determine the size of data first, then select the encryption algorithm under study and then calculated the averages by the following method:

a. Determine new set of the given sets that belong to that data category.
b. Perform the encryption process 20 times in each set. Store each value (the time consumed to encrypt that set in milliseconds) in a separate database or file.
c. Get the average of those 20 values and store the value, let's refer to it by Ai. The average $=\frac{\sum_{i=1}^{20} time}{20}$
d. Return to step (a) and perform the steps following it on the other set applying the following five sets.
e. When calculating five averages finished, consider the average of those averages, then the calculated value will be considered as the measured time for encrypted data. The average will be calculated as: $\sum_{i=1}^{5} A$

4.2 Determination of CPU Efforts Usage by the Algorithms under Study

The CPU efforts used during the execution of each algorithm under study were captured through the 'Task Manager' tool that is available in Windows7 Operating systems.

4.3 Memory Size Measurement

Size of memory reserved for running each one of algorithms under study was captured also through the 'Task Manager' tool that is available in Windows7 Operating system.

5. Result

Results that the researchers came out with are translated in both numeric representations in tabular manner as well as in statistical graphs to clearly identify the variances in results in comparing algorithms under study with each others in terms of time they took to operate, CPU efforts, and memory consumption. Results that will be expressed are measuring the mentioned encryption algorithms to encrypt the targeted transactions for encrypting the transaction header, then encrypting the data part individually in terms of the three pre-mentioned measurement attributes.

5.1 Results and Evaluation of Algorithms in encrypting Transaction Header (the transaction type and its ID parts) in Terms of Time

In this section we explore the evaluation and results of processing time that are taken by the algorithms under study: Periodic Permutation, Caesar, and Simple Substitution algorithm. In each subsection a statistical graph will be conduct to clarify typically the differences among each algorithm from the others and how each one acts according to the given type of each transaction under processing.

Table 1 represents average times taken in milliseconds to process each type of transaction assuming that the transaction has an ID of: 12345678910123456778910 (22 digits). The encryption included the transaction type which can either be a commit, an abort, starting a new transaction, read only transaction, or read-write transaction. Then a comma followed by it, then the transaction id comes finally.

Table 1
Times in milliseconds that are spent to encrypt transaction headers by different encryption algorithms

Type of Transaction	Encrypted data	Time spent by "Simple Substitution" Algorithm	Time spent by "Caeser" Algorithm	Time spent by "Periodic Permutation" Algorithm
Read-only transaction	*read_item,12345678910123456778910*	0.0879628	0.02069715	0.0456698
'Start' transaction	*start_transaction,12345678910123456778910*	0.09786245	0.01610025	0.0593548
Commit	*commit,12345678910123456778910*	0.0748133	0.0123584	0.0482362
Abort	*abort,12345678910123456778910*	0.05255535	0.0173404	0.0558486
Read-write transaction	*write_item,12345678910123456778910,database.emp.name*	0.08625235	0.0558486	0.0974134

Obviously, Caesar Algorithm had recorded the minimum time taken to encrypt the 5 types of transaction headers, then Periodic Permutation algorithm recorded the second score, then lastly the simple substitution set place. Figure 2 illustrates the differences in times taken by each algorithm in encrypting each type of transaction header.

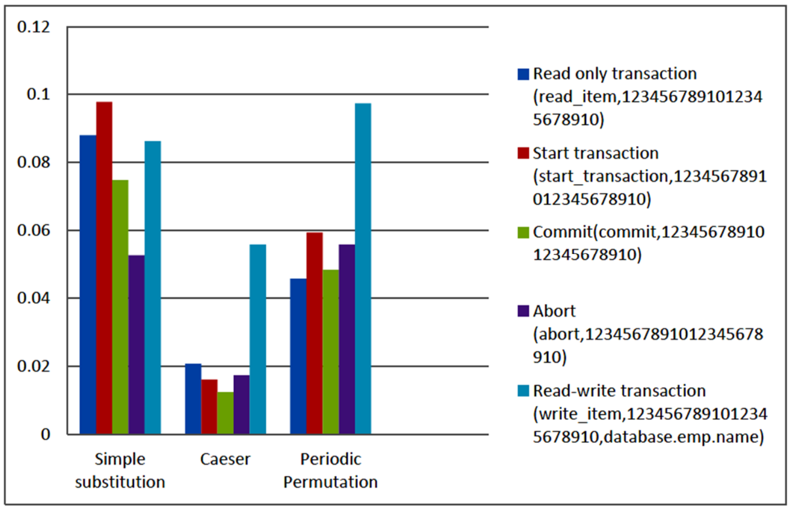


Fig. 2. Time elapsed in milliseconds for encrypting different database transaction headers types

In another hand, it's clear that type "started transaction" has recorded the biggest amount of time it took to encrypt by both simple substitution and, Periodic Permutation algorithms. Whereas by Caesar, it took less time than the read-only type did, but more time than the commit type did take. Also, it is clearly observed that the read-write transaction in all three algorithms consumed more time than the other transaction headers' types. While the start transaction encryption by the simple substitution algorithm recorded the biggest amount in all experiments from other types, Time consumed by commit transaction by Caesar algorithm had recorded the most little time than all other transaction headers that are encrypted by the same algorithm as well as by other algorithms.

Finally, according to both table 1 and figure2 above, Caesar had won the competition than other encryption algorithm, and we can ensure that it is the optimal one for encrypting such transaction header data; that is to say it is the most suitable option to be considered in terms of time in encrypting the first part of the transaction header (the part that consists of the transaction type and its id). But for Security issues, a regular change of the key each must be done several times to harden the process of cryptanalysis that can be performed by black hat hackers (crackers).

5.2 Evaluation of the Algorithms in Encrypting Data Part in the Transaction in Terms of Time

Data that is considered to obtain our results are categorized into three categories in terms of length. As we referred to that in the preceding chapter, 1st category is of size 65535 which is typically 65.535 kilobytes which is the maximum number of characters that the Varchar2 data type has in MySQL DBMS. The 2nd one is of the half of the maximum size of varchar2 which is 32768 characters. The 3rd category is of 16.384 kilobytes.

Table 2
Times spent for encrypting different data sizes

Type of Transaction	**65.535 kilobytes**	**32.768 kilobytes**	**16.384 kilobytes**
Simple substitution	3434.94	862.12	217.16
Caesar	3337.13	843.87	213.49
Periodic Permutation	3340.82	829.21	210.03

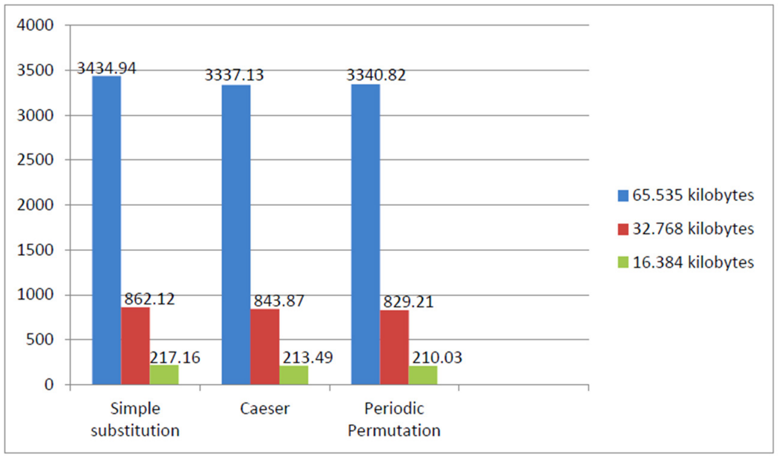


Fig. 3. Times elapsed in milliseconds to encrypt data part by different encryption algorithms

As it had appeared above in Figure 3, the three algorithms under study have nearly the same results. Most efforts and time spent to encrypt the data cluster of 65.535 kilobytes was recorded by the Simple Substitution method, It took3434.94 Milliseconds. While the least time spent for encrypting such cluster was 3337.13 Milliseconds by Caesar algorithm.

On the other hand, it has been found that data of 32.768 kilobytes took the largest period of time to be processed was recorded by Simple Substitution algorithm, and least time that the same size of data spent to be encrypted was recorded by Periodic Permutation algorithm.

The last data group with size 16.384 had been processed in nearly the same time via the three of algorithms. But accurately, Periodic Permutation consumed the least time from other with 210.03 milliseconds, while simple substitution had took the biggest plenty of time among the three with value of 217.16. Then, Caesar had got the second score with an average time of 213.49 millisecond consumed for encrypting that size of data. The variance between time spent in processing by Simple Substitutions and Caesar was just 4 milliseconds, and between Caesar & Periodic Permutation was only 3 milliseconds, this is an indicator of stability in data processing with mentioned size, so deciding which to choose cannot be easy, therefore, including other methods to support decision making such as machine learning will be valuable for more accurate decisions

5.3 Results and Evaluation of Performance of Encrypting the 'Data' Transaction Section in Terms of CPU Efforts Usage

After examining the pre-mentioned categories of data, the category of 65.535 kilobyte, 32.768, and of 16.384 kilobytes The results that our experiment has shown:

Table 3
Effort of CPU (in percentage) taken for encrypting different data sizes

Type of algorithm	**65535 kilobytes of data**	**32768 kilobytes of data**	**16384 kilobytes of data**
Simple substitution	19.47%	14.46%	0.5%
Caesar	23.97%	24.03%	20.725%
Periodic Permutation	25.02%	23.84%	20.825%

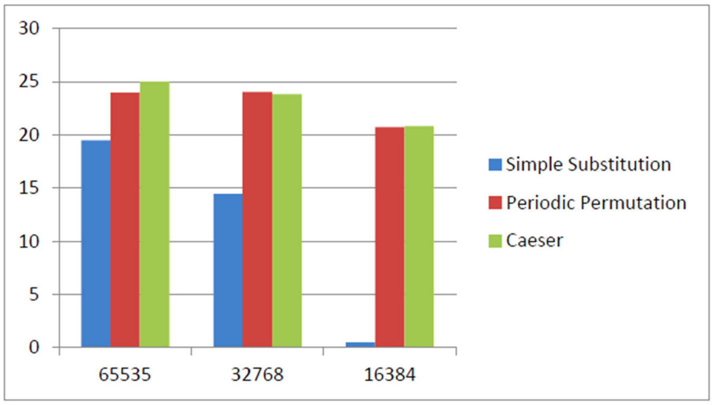


Fig. 4. CPU effort consumed (in percentage) for encrypting data part by different encryption algorithms

From the Chart above, i.e., figure4, the reader can see that Caesar algorithm had scored the last position in the competition between it and the other two algorithms in encrypting the largest size of data under study – 65.535 kilobytes of data. Whereas Simple Substitution algorithm was the optimal one in encrypting such size of data. Simple Substitution algorithm also came at the first place when encrypting the data of size 32.768 kilobytes, and regarding to the other two, the CPU efforts used for both was nearly the same but 'Periodic Permutation' had consumed a little more bits efforts than that Caesar did.

Regarding the third category of data –size 16.384 kilobytes – Simple Substitution algorithm won the competition once again with a very big difference in efforts that it did require from the CPU to perform comparing to the other two, it required just 0.5 of the CPU efforts. And once again, the other remaining algorithms had nearly a similar percentage of CPU efforts consumption but Caesar algorithm this time came at the tail of the list.

5.4 Results and Evaluation of Performance of Encrypting 'Data' Part of Transaction in Terms of Memory reserved

'Task Manager' tools were also used to measure the memory usage rate in kilo bytes for each algorithm to perform. Following table shows size of memory reserved and used in encrypting our data.

Table 4
Sizes of memory reserved for encrypting different data sizes using encryption algorithms mentioned below

Type of algorithm	**65535 kilobytes of data**	**32768 kilobytes of data**	**16384 kilobytes of data**
Simple substitution	353153.3333	293313.5333	18367.54545
Caesar	234924	351613.3333	353001.3333
Periodic Permutation	298140	295021.3333	295746

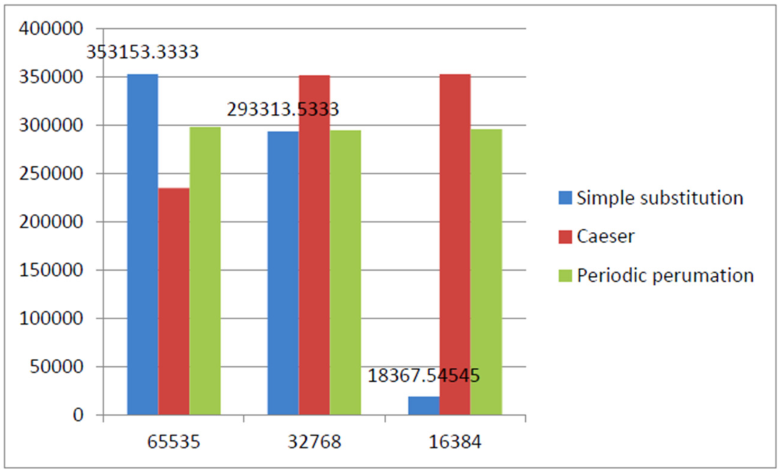


Fig. 5. Memory sizes reserved for encrypting the data part of the transaction by different encryption algorithms

As it is shown above, it's clearly that Caeser algorithm had won the competition and registered the first place in terms of memory consumption in large data – of size 65535 kilobytes. Whereas Simple substitution reserved the smallest memory size when encrypting 32.768 kilobytes of data while Caesar consumed the largest size of memory comparing to the other two, so it came at the tail – the third place. Regarding data of size: 16.384, Simple Substitution came also at the first place while Caesar had lost the competition once again and took extremely large size of memory in order to perform.

5. Discussion, Conclusion and Future Work

From the illustrated results above, in terms of time spending when encrypting the types of transactions, we can clearly see that Caeser algorithm is the best solution since it doesn't required intensive or further calculations and data referencing for encryption.

Regarding the encryption of the different categories of data sizes that the researchers referred to above, Caeser spent the smallest amount of time when encrypting big sizes of data, so it came at the first place compared to other algorithms. But in case of encrypting moderate and small sizes of data, periodic permutation was the winner while Simple Substitution came at the tale of the list since referencing to the array that holds the letters and their substitutions takes time, hence, an improvement of arrays referencing method is an essential issue that should be considered when trying to encrypt transaction's data with it.

In terms of CPU efforts consumption, Simple Substitution is generally the best in encrypting small pieces of data, moderate and extremely big size due to the stability of its method for referencing to the substitution array, while Caeser and Periodic Permutation are nearly the same in most cases.The optimal algorithm in memory sizes consumption for performing its work in case of small sizes of data is the "Simple Substitution" compared to Caeser and Simper Substution. In the other hand, Caeser is the worst in moderate and small sizes of data but it is the best in encrypting large data compared to Simple Substitution. But generally, we can observe that Periodic Permutation is stable in all sizes of data and hence it is the optimal algorithm to be used when the server's CPU was stable in usage, and since no additional data or referencing to them are required except the plaintext itself.

In conclusion, this paper can be one of the basic strategies that can be considered in order to enable database designers make decisions on what algorithms are more adequate in terms of time, CPU efforts, and memory size require at a time based on the amount of data to be encrypted.
Future work can briefly identify and recommend in next points:

- Other classical encryption algorithms should take part in such performance analysis and comparisons.
- Measuring the performance of non-text data should take place using special algorithms for doing so.
- Usually, Systems differ in capabilities in processing equipments, i.e., CPU, Cache, RAM …etc. And for further security guarantees a DBMS can use more than one encryption algorithm, one at a time. Therefore, introducing machine learning techniques and other decision support systems as well as OLAP methods for making such decision can benefit a lot, but this needs strong and high performance of the processing units.
- Using non-traditional encryption such as 3DES, AES, and so on to encrypt such transactions and to analyze their performance should also take chance to be under study in order to complete the big picture of complete performance analysis for encrypting database transactions.

References

[1] Natan, Ron Ben. *Implementing database security and auditing*. Elsevier, 2005.
[2] Ramiz Emasri and Shmkant B. Navathe. Fundamentals of Database Systems. UK, Addison-Wesley press., 745. 2011.
[3] Urbaniak, Geoffrey C., and Scott Plous. "Research Randomizer (Version 4.0)[Computer software]. Retrieved on June 22, 2013." (2013).
[4] Robling Denning, Dorothy Elizabeth. *Cryptography and data security*. Addison-Wesley Longman Publishing Co., Inc., 1982.
[5] Connolly, Thomas M., and Carolyn E. Begg. *Database systems: a practical approach to design, implementation, and management*. Pearson Education, 2005.
[6] Gangwar, Reena, and M. K. Sharma. "Database Security Measurements Issues in Adhoc Network." In *International Conference of Advance Research and Innovation*. 2015.

[7] Roselin Selvarani, D., and T. N. Ravi. "A survey on data and transaction management in mobile databases." *arXiv preprint arXiv:1211.5418* (2012).

[8] Weider, D. Yu, Tamseela Amjad, Himani Goel, and Tanakom Talawat. "An approach of mobile database design methodology for mobile software solutions." In *Grid and Pervasive Computing Workshops, 2008. GPC Workshops' 08. The 3rd International Conference on*, pp. 138-144. IEEE, 2008.

[9] AminuBaba, Murtala, Abdulrahman Yusuf, Aminu Ahmad, and Ladan Maijamana. "Performance Analysis of the Encryption Algorithms as Solution to Cloud Database Security." *International Journal of Computer Applications* 99, no. 14 (2014): 24-31.

[10] Asole, S. S., and Ms SM Mundada. "A survey on securing databases from unauthorized users." *International Journal of Scientific & Technology Research* 2, no. 4 (2013): 228-230.

[11] Basharat, Iqra, Farooque Azam, and Abdul Wahab Muzaffar. "Database security and encryption: A survey study." *International Journal of Computer Applications* 47, no. 12 (2012).

[12] Sathiya, D., S. Albert Rabara, and J. Ronald Martin. "A Framework for Secure Mobile Database Transactions using Cryptographic Co-processor." *International Journal of Computer Applications* 105, no. 5 (2014).

[13] Kaur, Mandeep, and Manish Mahajan. "Using encryption algorithms to enhance the data security in cloud computing." *International journal of communication and computer technologies* 1, no. 12 (2013): 56-59.

[14] Selvarani, D. Roselin, and Dr TN Ravi. "A Review on the role of Encryption in Mobile Database Security." *Internation Journal of Application or Innovation in Engineering and Management*(2014).

[15] Ghorbanzadeh, Parviz, Aytak Shaddeli, Roghieh Malekzadeh, and Zoleikha Jahanbakhsh. "A survey of mobile database security threats and solutions for it." In *Information Sciences and Interaction Sciences (ICIS), 2010 3rd International Conference on*, pp. 676-682. IEEE, 2010.

[16] Sharma, Gaurav, and Ajay Kakkar. "Cryptography Algorithms and approaches used for data security." *International Journal of Scientific & Engineering Research* 3, no. 6 (2012): 1-6.