

# A Performance Comparison of Microsoft<sup>®</sup> SQL Server<sup>®</sup> 2008 Transparent Data Encryption and NetLib<sup>®</sup> Encryptionizer<sup>®</sup>

P. D. Colbert and B. A. Juliano

Department of Computer Science, California State University, Chico, Chico, CA, USA

**Abstract** - *Data and information security are critical elements of the modern world. Relational database management systems (RDBMS) such as Microsoft<sup>1</sup> SQL Server are the backbone of data storage, retrieval, and manipulation. With SQL Server 2008 Enterprise, Microsoft introduces Transparent Data Encryption (TDE); also known as whole database encryption, TDE secures complete database instances using industry standard Advanced Encryption Standard (AES) and Triple Data Encryption Standard (TDES) encryption ciphers. Prior to Transparent Data Encryption, whole database encryption has been readily available for Microsoft SQL Server using Encryptionizer from NetLib. This paper details the methodology, results, and conclusions in comparing the performance of 128-bit AES encrypted TDE and Encryptionizer SQL Server 2008 instances against a baseline, unencrypted instance. Benchmark data results were collected from virtual single processor systems using Virtual PC and VMware, and a non-virtual dual processor host system. The evaluation included ten separate benchmarks in five categories: data retrieval, insertion, update, deletion, and backup. All evaluations were conducted using SQL scripts executed using Microsoft Transact-SQL against a sample Northwind database expanded in data size by a factor of 100. Virtual single processor results yielded a slight performance advantage for Encryptionizer over Transparent Data Encryption, and a significant performance advantage for Encryptionizer over Transparent Data Encryption in the non-virtual dual processor configuration. Final benchmark results followed a balanced execution pattern to minimize the impact of indeterminate errors.*

**Keywords:** Database Encryption Benchmark Performance Encryptionizer.

## 1 Introduction

Data and information security are critical elements of our modern world. Relational database management systems (RDBMS) are one of the most widely used computer mechanisms for the storage, retrieval and manipulation of

data. Theft of information from database systems often involves acquiring the physical database files containing the information [1], [2], [3], either by copying the database files, or by obtaining the computer containing the database files. Unless the physical database files are in some way encrypted, information is easily retrieved.

With the release of SQL Server 2008 Enterprise (MSSQL), Microsoft introduced Transparent Data Encryption (TDE) for their RDBMS. Enabling TDE results in the encryption of all information stored within the database system, also known as whole database encryption [4]. Without proper authentication credentials, access to the data within the database system is prevented.

While Microsoft is new to the concept of whole database encryption, the NetLib Encryptionizer (NLE) product has been providing this level of security for Microsoft SQL Server products for over 10 years. In this paper, the authors compare execution times of Transparent Data Encryption with Encryptionizer while performing routine data manipulation.

The authors detail the methodology, results, and conclusions in comparing the performance of 128-bit Advanced Encryption Standard (AES) encrypted TDE and NLE SQL Server 2008 instances against a baseline, unencrypted instance. Benchmark data results were collected from virtual single processor systems using Virtual PC and VMware, and a non-virtual dual processor host system. The evaluation included ten separate benchmarks in five categories: data retrieval, insertion, update, deletion, and backup. Initial benchmarking involved executing each individual benchmark in isolation to determine a relative iteration count to achieve a five to twenty second overall execution time for that benchmark. This unbalanced approach led to a balanced execution pattern to minimize the impact of indeterminate errors. The derived iterations were used for each benchmark, executing the benchmark sequentially across each database instance.

<sup>1</sup> Microsoft, SQL Server, and Windows are registered trademarks or trademarks of Microsoft Corporation, Inc. in the United States and/or other countries. NetLib and Encryptionizer are registered trademarks of Communication Horizons LLC. VMware is a registered trademark or trademark of VMware, Inc. in the United States and/or other jurisdictions.

## 2 Data set and test suite availability

All data and test suites are publicly available from the author. Data and test suites employ standards appropriate for RDBMS evaluations, including but not limited to adding new data, updating existing data, and deleting data. The database used for all of the benchmark tests is based on the Microsoft Northwind database [7] expanded in size by a factor of 100 using a modified Transact-SQL (T-SQL or TSQL) script originally created by Scott Mauvais called BigNW [11]. The Microsoft Northwind database was selected as representing a readily available and well-known database file that includes extensive intra-table referential relationships, stored

Table	Northwind Row Count	BigNW Row Count
Categories	8	8
CustomerCustomerDemo	0	0
CustomerDemographics	0	0
Customers	91	9191
Employees	9	9
EmployeeTerritories	49	49
OrderDetails	2155	217655
Orders	830	83830
Products	77	77
Region	4	4
Shippers	3	3
Suppliers	29	29
Territories	53	53

Fig. 1 Row count comparisons between Northwind and BigNW database tables.

procedures, and views. Figure 1 compares the original Northwind database to the expanded BigNW database. The benchmark TSQL scripts focused on the BigNW Orders table.

## 3 Related work

No comparison between MSSQL TDE and NLE exists. Performance benchmarks are available from NetLib that indicates a negligible impact when using NLE in comparison to non-encrypted systems [5], although the results are for AES encryption only, and do not indicate number of iterations, key strength, number of processors or processor cores, or operating systems. Microsoft estimates systems with low processor and input/output load to experience a 3-5% reduction in overall performance, and systems with higher loads to reach up to 28% reduction [4].

The use of a standard benchmarking platform is preferred in any benchmarking endeavor. For RDBMS benchmarks, the industry standard product TPC Benchmark™ [9] was cost prohibitive, and led to the development of a proprietary benchmarking platform. Development of the proprietary benchmark followed the recommendations of the ANSI SQL Standard Scalable and Portable (AS3AP) benchmark guidelines: comprehensive but tractable, scalable and portable, minimize human effort, and

uniform metric [10]. The selected unit of measure was the query execution time measured in milliseconds.

The use of a virtual environment to simulate a single processor has been demonstrated to be a viable alternative for benchmark platforms [8]. Full virtualization, or running a fully installed but virtualized operating system within another host operating system, was selected to benchmark a single processor system.

## 4 Experimental procedures

### 4.1 Benchmark hardware and software

All tests were conducted using an Intel® Core™2 Duo 2.13 GHz Processor with 3.25 GB RAM running Microsoft Windows XP Professional operating system with Service Pack 3. The virtual environment configuration for Microsoft Virtual PC used version 6.0.156.0 with 1024 MB RAM running Microsoft Windows XP Professional operating system with Service Pack 3. The virtual environment configuration for VMware Workstation used version 6.5.0 build-118166 with 1024 MB RAM running Microsoft Windows XP Professional operating system with Service Pack 3. All benchmark database instances used Microsoft SQL Server 2008 Enterprise code name "Katmai" (CTP) version 10.0.1075.23. The hardware virtualization capability of the host operating system Intel® Core™2 Duo was enabled for all virtual environment tests. All virtual and host operating systems software, SQL Server software, and NetLib Encryptionizer software were 32-bit. The NetLib Encryptionizer version was 2007.101.20.8.3.4a.

### 4.2 Database instances

To evaluate the execution time differences between instances of SQL Server encrypted with TDE and an instance encrypted with NLE, two distinct instances were installed. For reference, these two instances will be referred to as the TDE instance and as the NLE instance, respectively. In addition, an unencrypted instance was created to serve as a baseline, and will be referred to as the Base instance. The TDE and NLE instances were encrypted using 128-bit AES encryption cipher. The AES encryption cipher replaced the Data Encryption Standard (DES) cipher May 26, 2002 as the United States government cryptographic security standard for all sensitive data [6].

All three instances were identical in configuration other than the instance name, accepting the default values as presented by the Microsoft SQL Server install program. Microsoft SQL Server 2008 Enterprise was used, as only this version of the product supports TDE. NetLib Encryptionizer does not have this limitation, and is backward compatible with earlier versions of Microsoft SQL Server, as well as all current versions.

### 4.3 Test database

The Microsoft Northwind test database was installed in each of the three instances, and then expanded by a factor of 100 using the BigNW TSQL script authored by Scott Mauvais. The script was modified to work with SQL Server 2008. The original schema of the Northwind database is maintained in the BigNW database.

### 4.4 Benchmark tables and scripts

Within each instance a benchmark results table was created to record the execution time results. The benchmark suites as TSQL scripts were stored within benchmark script tables. Each benchmark script includes parameters indicating the number of iterations to repeat the script that constitutes a measurable single run, and a repeat parameter for the number of measurable executions of the script. Following the benchmark testing, results were exported to text files for collation and analysis. During the testing, of special consideration was the issue of caching. The test suites utilized TSQL commands unique to SQL Server to clear any cached data between subsequent iterations.

Category	Benchmark Description
Retrieve	Select orders ordered on or after 1/1/1996 using index seek
Retrieve	Select orders shipped to Germany using index scan
Retrieve	Select all orders joined with customers table
Retrieve	Select all invoices from Invoices view
Retrieve	Execute stored procedure Employee Sales by Country inserted into temp table
Insert	Select all orders and insert into temporary table
Insert	Insert new orders
Update	Update orders information
Delete	Delete orders
Backup	Add a backup device and backup the entire database

Fig. 2 Benchmark descriptions by category.

The test suite consisted of ten scripts grouped into one of five main categories as shown in Figure 2: retrieval, insertion, update, deletion, and backup. A driver script was developed that retrieved the stored test scripts, repeated the script the specified number of times to constitute a single measurable unit, and then repeated this process to produce the specified number of timed executions. Using the database to store the scripts and control parameters allowed for a flexible and easily configured testing environment. The driver script included commands for clearing any cached data between script executions if specified by the benchmark control parameters.

### 4.5 Benchmark executions

With the flexibility of the driver script, different script execution iterations and repetitions were benchmarked. Initially, script parameters were configured to determine the approximate execution time of a single run of each benchmark for both virtual environments, and the dual-

processor non-virtual environment. These unbalanced benchmarks yielded execution times, but because of the length of each benchmark execution, were not comparable to executions of the same benchmark within each instance. During a lengthy execution of a benchmark, the underlying computer system may experience any number of unknown system processes that impact the execution times of that benchmark. To minimize the impact of these indeterminate errors, a balanced approach was conceived and executed on the virtual single processor systems, and the dual-processor non-virtual system. This balanced benchmark approach consisted of estimating the minimum number of iterations per benchmark that would result in a mean execution time between five and twenty seconds. Each benchmark would be iterated sequentially for each database instance, with the entire process being repeated 1000 times.

## 5 Experimental results and discussion

### 5.1 Unbalanced virtual environment

Experimental benchmark executions were conducted within each of the virtual environments, as well as the non-virtual dual-processor host operating system. These tests were

	Mean Execution Time (ms)			% Difference NLE to TDE
	Base	TDE	NLE	
1	72398	89549	83604	6.6
2	46753	66735	56110	15.9
3	39263	50016	45400	9.2
4	326105	368798	359366	2.6
5	74148	119022	108294	9.0
6	74102	86631	84210	2.8
7	10430	22115	17276	21.9
8	9232	11350	11119	2.0
9	22138	28267	27564	2.5
10	269870	265143	288022	-8.6

Fig. 3 Unbalanced benchmark results for Virtual PC.

initially conducted as a means of comparing results for a simulated single processor system versus a dual processor system, as well as give a relative performance comparison between each of the two virtual computer products. Initial benchmark tests followed a pattern of sequentially conducting each of the ten benchmarks for a single instance, followed by

	Mean Execution Time (ms)			% Difference NLE to TDE
	Base	TDE	NLE	
1	59396	86534	78037	9.8
2	37563	58069	48035	17.3
3	37901	47930	45130	5.8
4	321518	362240	351275	3.0
5	69604	117810	101805	13.6
6	57358	84087	75750	9.9
7	9524	15297	10425	31.8
8	5788	9945	8462	14.9
9	13056	25320	22944	9.4
10	209970	211280	232178	-9.9

Fig. 4 Unbalanced benchmark results for VMware.

the next instance until all three instances were benchmarked. This pattern would then be repeated. Such an approach yielded results with statistically unacceptable distributions and sample means when analyzing all execution times for a single benchmark for each database instance. This method of isolated benchmarking is considered an unbalanced method. Error percentages for the sample mean ranged from  $\pm 0.5\%$  to  $83.1\%$  at the 95% confidence level. Without a balanced approach to executing the benchmarks, any system anomalies that might impact the execution speed produced localized indeterminate errors. Figure 3 reports the unbalanced experimental execution speed mean times for the Virtual PC virtual platform. Figure 4 reports the unbalanced experimental execution speed mean times for the VMware virtual platform.

## 5.2 Unbalanced non-virtual dual-processor environment

Following the unbalanced virtual environment benchmarks, rigorous executions were undertaken to determine the relative execution times on a non-virtual dual-processor system for each benchmark. These experimental results, shown in Figure 5, yielded a mean average time for

	Mean Execution Time (ms)			$\pm$ % At 95% Confidence Level		
	Base	TDE	NLE	Base	TDE	NLE
1	56594	82177	57553	1.6	5.0	1.5
2	33211	60207	36752	4.0	6.7	3.7
3	19992	28019	22609	0.9	7.9	1.1
4	332533	366869	328543	0.3	0.4	0.6
5	99759	117574	91712	0.8	0.4	0.8
6	56530	76831	56420	1.8	0.7	1.6
7	4373	8554	6023	5.6	2.5	5.4
8	3000	5361	3927	6.1	2.5	5.1
9	3990	9187	5836	3.5	2.4	2.7
10	293895	279958	27532	2.3	0.2	0.4

Fig. 5 Unbalanced benchmark results for a dual-processor system at the 95% confidence level.

each benchmark used to determine the minimum number of iterations necessary to enable each benchmark to execute within a five to twenty second execution timeframe. Figure 6 lists the derived iterations that were subsequently used in conducting the balanced benchmark experiments. The minimum threshold of five seconds was selected to ensure a

Benchmark	Iterations
1	12
2	19
3	32
4	2
5	7
6	12
7	143
8	10
9	100
10	3

Fig. 6 Derived iterations for balanced benchmark testing.

reasonable and comparable benchmark execution time.

## 5.3 Balanced virtual environment

The unbalanced approach demonstrated the need for a sequential methodology as a superior benchmarking technique. The experimental results of the balanced benchmarking approach for the Virtual PC platform are depicted in Figure 7, reporting that NLE executes up to  $6.9\%$  faster ( $\pm 2.8\%$  at the 95% confidence level) when compared to TDE. The experimental results of the balanced

	Mean Execution Time (ms)			$\pm$ % At 95% Confidence Level		
	Base	TDE	NLE	Base	TDE	NLE
1	7963	9524	8523	2.8	1.1	0.9
2	6531	8118	6685	1.6	1.2	1.6
3	5202	6974	6757	0.8	3.1	3.5
4	6838	7881	7679	0.4	1.0	2.7
5	6092	8796	7894	3.4	2.5	1.0
6	6314	7684	7163	1.5	0.5	4.0
7	7414	10151	8741	3.6	5.1	3.4
8	3532	4698	4298	1.0	0.8	2.2
9	8691	11953	11501	8.9	6.4	9.3
10	8135	7968	8624	3.0	2.7	2.9

Fig. 7 Balanced Virtual PC benchmark results, with an average sample mean decrease in execution speed for NLE compared to TDE of  $6.9\%$  ( $\pm 2.8\%$  at the 95% confidence level).

benchmarking approach for the VMware platform are depicted in Figure 8, reporting that NLE executes up to  $4.1\%$  faster ( $\pm 1.3\%$  at the 95% confidence level) when compared to TDE. Both virtual platforms consisted of 100 executions of the benchmark scripts.

	Mean Execution Time (ms)			$\pm$ % At 95% Confidence Level		
	Base	TDE	NLE	Base	TDE	NLE
1	6301	8719	8138	1.9	1.1	1.3
2	4886	7083	6313	1.0	0.5	0.7
3	5398	6435	6630	1.3	0.9	0.8
4	6112	6600	6882	1.0	1.0	1.1
5	4999	8079	7227	0.6	0.4	0.4
6	4904	6896	6373	1.6	1.3	1.3
7	7820	8039	7002	4.7	5.5	3.1
8	10565	13606	12576	1.0	0.9	0.8
9	16283	15785	15493	0.4	1.5	0.9
10	5751	5694	6246	1.0	0.9	0.9

Fig. 8 Balanced VMware benchmark results, with an average sample mean decrease in execution speed for NLE compared to TDE of  $4.1\%$  ( $\pm 1.3\%$  at the 95% confidence level).

## 5.4 Balanced non-virtual dual-processor environment

The experimental results of the balanced benchmark executions for the non-virtual dual processor platform are depicted in Figure 9, reporting that NLE executes up to  $17.2\%$  faster ( $\pm 0.4\%$  at the 95% confidence level) when compared to TDE, and only  $10.9\%$  slower ( $\pm 0.6\%$  at the 95% confidence level) than an unencrypted MSSQL instance.

The balanced results are graphically represented in Figure 10, and include the error margins at the 95% confidence level as bar extensions to the right of each benchmark. The non-virtual, dual-processor consisted of 1000 executions of the benchmarking scripts.

	Mean Execution Time (ms)			± % At 95% Confidence Level		
	Base	TDE	NLE	Base	TDE	NLE
1	7066	9324	7014	0.4	0.2	0.3
2	6763	11173	7172	0.6	0.3	0.5
3	6580	9000	7715	0.2	0.2	0.3
4	6618	7318	6749	0.1	0.1	0.1
5	7093	8283	7107	0.2	0.2	0.2
6	7108	9317	7280	0.4	0.2	0.3
7	3958	9653	6580	1.1	0.7	1.7
8	15377	15581	15568	0.1	0.2	0.1
9	4651	10684	7345	3.4	0.3	1.5
10	8783	8564	9419	0.1	0.1	0.1

Fig. 9 Balanced dual-processor benchmark results, with an average sample mean decrease in execution speed for NLE compared to TDE of 17.2% ( $\pm 0.4\%$  at the 95% confidence level).

## 5.5 Execution runtime versus CPU ticks

In addition to recording the execution time, the number of CPU ticks MSSQL spent during the processing of the TSQL benchmark was also recorded during the same time period. The mechanism for determining the execution time involved recording the time at the beginning of a measurable unit, then again at the conclusion of a measurable unit, and taking the time difference as the execution time. System processes that utilize the computer core may also prevent the benchmark from executing, but allow the time differential to continue to grow, suggesting that CPU ticks might be a more accurate measurement for comparison. However, at the heart of any RDBMS is the hard disk, a heavily used system resource. MSSQL does provide a means for tracking how much CPU time was spent doing input and output functionality. This information was not recorded for the benchmarks, and is suggested for future work.

## 5.6 Statistical analysis of data

Meaningful statistical comparison of data requires that data elements to be compared be synchronized as close as possible in time. The balanced approach rigorously followed as part of these benchmark experiments allows for the use of common statistical measurements: sample mean, sample standard deviation, sample confidence level, and a  $\pm$  percent of the sample mean at the 95% confidence level. All non-virtual experimental results were below  $\pm 2\%$  margin of error at the 95% confidence level.

## 5.7 Experimental anomalies and data outliers

When analyzing the experimental results, statistical efforts must be undertaken to identify and resolve the impact of outliers on the experimental data. By definition,

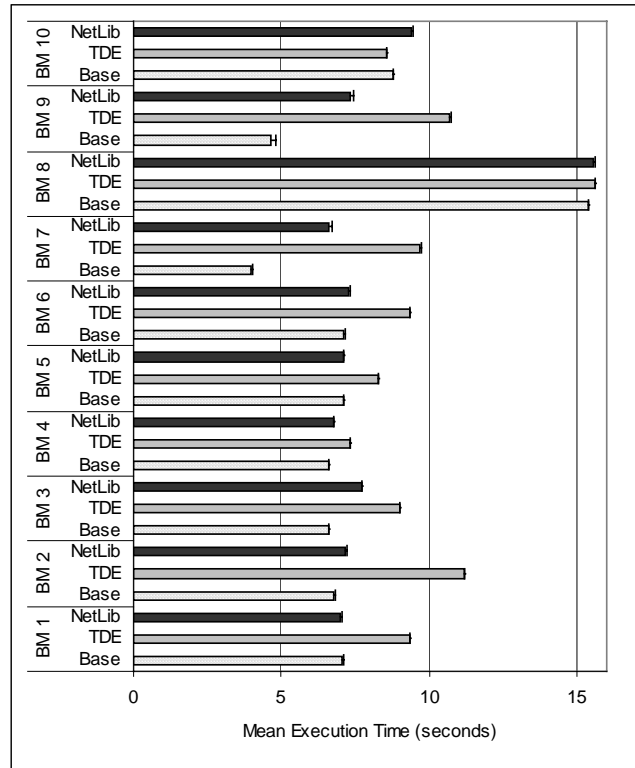


Fig. 10 Balanced dual-processor benchmark results, with an average sample mean decrease in execution speed for NLE compared to TDE of 17.2% ( $\pm 0.4\%$  at the 95% confidence level).

indeterminate system activities are of unknown duration and unknown quantity and may impact the execution time of one or more benchmark tests. The goal is to execute a single benchmark in as short a time as possible, yet a minimum threshold must be targeted to avoid reaching a minimum that is recording the minimum and not the actual execution time. In this study, a minimum target time of five seconds, but no more than twenty seconds, was selected as a reasonable range. Minimizing the measurable unit of time will limit the impact per instance for a single pass of a benchmark test. Outside of the measured time, the benchmarking process consists of a time overhead during which system activities may occur, thus elevating the interval switch time for a single benchmark to potentially over 60 seconds. System activities that extend beyond the current interval switch time will result in outliers across more than one instance, and must be addressed statistically.

## 6 Conclusion

The experimental results clearly indicate that both Transparent Data Encryption and Encryptionizer increase the execution time of SQL Server. On a non-virtual, dual-processor system, experimental results demonstrate that Encryptionizer executed the benchmarks faster than TDE an average sample mean of 17.2% ( $\pm 0.4\%$  at the 95% confidence level). Running within a virtual environment simulating a single processor, experimental results for Virtual PC demonstrate that Encryptionizer executed the benchmarks

faster than TDE an average sample mean of 6.9% ( $\pm 2.8\%$  at the 95% confidence level), and for VMware an average sample mean of 4.1% ( $\pm 1.3\%$  at the 95% confidence level). TDE mean execution time increased by 25.2% ( $\pm 0.4\%$  at the 95% confidence level) over the baseline instance on a non-virtual, dual-processor system.

## 7 Future work

A significant limitation in conducting the benchmark experiments was the lack of a diverse set of computers with differing number of processors. Additional experiments should be conducted using non-virtual single-processor computers to verify the relative similarity of both encryption mechanisms. Quad-processor and larger configurations should be conducted to more closely simulate larger corporate database environments. Additionally, 64-bit versions of both operating systems and SQL Server 2008 should be tested, along with the different server operating systems available from Microsoft. For all tests, the balanced benchmark approach is preferred to attempt to negate any unknown system processes that might impact the benchmark execution times on a per benchmark basis. Additionally, 256-bit encryption strength should be conducted. NetLib Encryptionizer does not support the 192-bit encryption strength. Because of the length of time to execute each benchmark for all three instances, multiple machines would greatly assist in conducting all experiments in a more reasonable amount of time, although care must be taken to ensure that for a single benchmark, the same machine is used.

Another avenue to explore would be to use the recorded CPU ticks rather than execution time, and exclude the cache clearing from the measurable unit. Benchmark experiments should be conducted to deduct the amount of CPU ticks spent by MSSQL engaged in input and output operations. The use of CPU ticks without input and output measurements may result in more statistically accurate sample mean comparison values.

Additional future research should consider increasing the database size, increasing the number of iterations, especially within the virtual environments, to increase the statistical accuracy of the results, and increasing the quantity and complexity of the benchmark queries. Subjecting the three instances to the industry standard TPC benchmarking software is also strongly encouraged.

## 8 References

- [1] The Canadian Press, "National Bank reports theft of laptop with mortgage loan database," September 23, 2008.
- [2] C. Boss, "Reynoldsburg student information stolen," The Columbus Dispatch, August 28, 2008.

[3] D. Migoya, "Stolen state database puts 1.4 million at ID-theft risk," Denver Post, November 2, 2006.

[4] S. Hsueh, "Database Encryption in SQL Server 2008 Enterprise Edition," SQL Server Technical Article, Feb. 2008. Retrieved from the World Wide Web Sep. 24, 2008, <http://msdn.microsoft.com/en-us/library/cc278098.aspx>.

[5] NetLib, "Performance Benchmarks – Whole Database Encryption," Communications Horizon, LLC., Retrieved from the World Wide Web Sep. 24, 2008, [http://www.netlib.com/files/performance\\_benchmarks\\_whole\\_db.pdf](http://www.netlib.com/files/performance_benchmarks_whole_db.pdf).

[6] National Institute of Standards and Technology (NIST), "Announcing the Advanced Encryption Standard (AES)," Federal Information Processing Standards Publication (FIPS) 197, United States of America Federal Government, November 26, 2001. Retrieved from the World Wide Web Oct. 4, 2008, <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>.

[7] Microsoft Corporation, Inc., "Access 2000 Tutorial: Northwind Traders Sample Database," June 22, 1999. Retrieved from the World Wide Web Sep. 24, 2008, <http://www.microsoft.com/downloads/details.aspx?familyid=C6661372-8DBE-422B-8676-C632D66C529C&displaylang=en>.

[8] M.L. Catalan, R. Ludena, A. Dennis, H. Umeno, "VM-Based Benchmark and Analysis System for Testing Online Transaction Processing," Second International Conference on Innovative Computing, Information and Control, ICICIC 2007, 2008, p. 4427665.

[9] Transaction Processing Performance Council, "TPC Benchmark E, Standard Specification, Version 1.6.0," Technical White Paper, 2008.

[10] C. Turbyfill, C. Orji, D. Bitton, "AS3AP – A Comparative Relational Database Benchmark," IEEE Computer Society International Conference, pp. 560-64, Feb. 1989.

[11] S. Mauvais, "Big Northwind Sample Code," May 23, 2007. Retrieved from the World Wide Web Sep. 24, 2008. <http://www.mauvais.com/Download/ZD-BigNW.htm>.