Evaluating the Feasibility Issues of Data Confidentiality Solutions from a Data Warehousing Perspective

Ricardo Jorge Santos[1], Jorge Bernardino[1,2] and Marco Vieira[1]

[1] CISUC – Centre of Informatics and Systems of the University of Coimbra
University of Coimbra, 3030-290 Coimbra, Portugal
[2] ISEC – Superior Institute of Engineering of Coimbra
Polytechnic Institute of Coimbra, 3030-190 Coimbra, Portugal
lionsoftware.ricardo@gmail.com, jorge@isec.pt, mvieira@dei.uc.pt

Abstract. Data Warehouses (DWs) are the core of enterprise sensitive data, which makes protecting confidentiality in DWs a critical task. Published research and best practice guides state that encryption is the best way to achieve this and maintain high performance. However, although encryption algorithms strongly fulfill their security purpose, we demonstrate that they introduce massive storage space and response time overheads, which mostly result in unacceptable security-performance tradeoffs, compromising their feasibility in DW environments. In this paper, we enumerate state-of-the-art data masking and encryption solutions and discuss the issues involving their use from a data warehousing perspective. Experimental evaluations using the TPC-H decision support benchmark and a real-world sales DW support our remarks, implemented in Oracle 11g and Microsoft SQL Server 2008. We conclude that the development of alternate solutions specifically tailored for DWs that are able to balance security with performance still remains a challenge and an open research issue.

Keywords: Data masking, Encryption, Data security, Confidentiality, Database Performance, Data Warehousing.

1 INTRODUCTION

Data Warehouses (DWs) store the secrets of the business and are used to produce business knowledge, which makes them a major target for attackers [4, 7, 20]. As the number and complexity of attacks increase, efficiently securing the confidentiality of DWs is critical [15, 18, 23]. To accomplish this, data masking and encryption solutions are widely used. Although data masking routines are simpler than encryption routines, they provide lower security strength. Moreover, data masking routines provided by most commercial tools typically change data in an irreversible manner, *i.e.*, after masking data it not possible to subsequently retrieve the original true values, making them useless for real live DW databases. This has made masking solutions the main choice for protecting published data or production data, instead of real-live databases [5, 8, 11, 12, 16, 20, 21]. Published research and best practice guides have stated that encryption is the best method to protect sensitive data at the database level while maintaining high database performance [3, 7, 8, 11, 12, 13, 14, 18, 23].

Despite their security strength, encryption techniques introduce performance key costs from a data warehousing point of view:

- Large processing time/resources for encrypting sensitive data, given routine or hardware access in very large databases such as those in DWs;
- Extra storage space of encrypted data. Since DWs usually have many millions or billions of rows, even a small modification of any column size to accommodate encrypted output introduces large storage space overheads;
- Overhead query response time and allocated resources for decrypting data to process queries. Given the huge amount of data typically accessed by DW queries, this is probably the most significant drawback in DWs [3].

The two main features that differentiate one confidentiality solution from the other are its security strength and its execution speed and efficiency. Given the specificity of DW environments, we believe there are specific performance issues to evaluate and discuss, regarding the use of encryption solutions. This is the foundation of this research work. We present the state-of-the-art solutions for protecting stored data confidentiality and evaluate their feasibility from a data warehousing perspective. It is not within the scope of this paper to discuss the scientific merit or soundness of the security strength of each technique, but rather to evaluate their impact on database performance and applicability in data warehousing environments.

Thus, in this paper we analyze and discuss the technical issues involving the implementation of the available state-of-the-art data confidentiality solutions, and use storage space and query response time as measures for evaluating performance in both loading and querying DW data. To support our remarks and claims, we include experimental evaluations using the TPC-H decision support benchmark [17] and a real-world sales DW, implemented in two leading commercial DataBase Management Systems (DBMS), such as Oracle 11g and Microsoft SQL Server 2008.

The main contributions of our work are as follows:

- We enumerate and describe the current state-of-the-art techniques for protecting stored data and discuss their application from a data warehousing perspective;
- We present the results of several experimental evaluations using two leading commercial DBMS and one leading open-source DBMS on a well-known benchmark (TPC-H) and a real-world DW, analyzing the impact in database performance due to using encryption techniques for protecting data confidentiality;
- The issues discussed and the results from our experimental evaluations allow us to state that currently available encryption solutions are not suitable for most DWs. Our work shows that the development solutions specifically tailored for DWs that are able to present better tradeoffs in balancing security strength with database performance remains a challenge and a relevant research issue.

The remainder of the paper is structured as follows. In section 2, we describe state-of-the-art data masking and encryption solutions for databases and discuss their issues from the DW perspective. Section 3 provides experimental evaluations of those solutions using the well-known TPC-H decision support benchmark and a real-world sales DW implemented in two leading commercial DBMS. In section 4 we point out research challenges and opportunities regarding specific confidentiality solutions for DWs, from the lessons learned. Finally, section 5 presents our conclusions.

2 STATE-OF-THE-ART DATA CONFIDENTIALITY

2.1 Data Masking Solutions

An extensive survey on data masking is given in [16]. Many organizations have strived to solve confidentiality issues with hand-crafted solutions within the enterprise to solve the problem of sharing sensitive information. The most common solution is probably to use scripts with triggers in order to mask and unmask each value, or to embed the masking/unmasking logic within the user applications themselves.

Many commercial data masking packages have also been developed, such as the Oracle Data Masking (ODM) pack [11, 12], protecting data by replacing real values with realist-looking data with the same type and characteristics as the original data. Once applied, the ODM masking process is irreversible. Oracle states ODM is to be used as a fast and easy way to generate production databases for supporting outsourcing and software application development. It can also be used to mask Microsoft SQL Server and DB2 databases for the same purpose. ODM requires new data to be loaded into the database first, and only applies the masking procedures afterwards. It is not possible to load previously masked data; masking in ODM is an *a posteriori* process. Most commercial data masking solutions work in a similar fashion as ODM.

Recently, research has proposed non-deterministic methods for masking data, such as **perturbation techniques** [4, 14, 19]. The work in [4] proposes a solution based on data perturbation techniques and explains data reconstruction for responding to queries, in a DW environment. Recent similar work proposed data anonymization solutions relying on perturbation or differential techniques [14] and [19].

2.2 Data Encryption Algorithms

Typical encryption algorithms include executing bit shifting and exclusive Or (XOR) operations within a predefined number of rounds. These operations rely on a key, which influences the "data mix" output of each round. There are mainly two types of encryption techniques: **Block Ciphers** and **Stream Ciphers**.

A block cipher is a type of symmetric-key encryption algorithm that transforms a fixed-length block of *plaintext* (unencrypted text) data into a block of *ciphertext* (encrypted text) data of the same length, under the action of a user-provided secret key. Decryption is performed by applying the reverse transformation to the ciphertext block using the same secret key. Stream ciphers take a string (the encryption key) and deterministically generate a set of random-seeming text (called *keystream*) from that key. That keystream is then XORed against the message to encipher. To decipher the text, the recipient hands the same key to the stream cipher to produce an identical keystream and XORs it with the ciphertext, thus retrieving the original message.

The **Data Encryption Standard (DES)** became the first encryption standard [6]. DES is a 64 bit block cipher, which means that data is encrypted and decrypted in 64 bit chunks, and uses a 56 bit encryption key. As an enhancement of DES, the **Triple DES (3DES)** standard was proposed [1]. The 3DES encryption algorithm is similar to the original DES, but it is applied three times to increase the encryption level, using three different 56 bit keys. Thus, the effective key length is 168 bits. Since the algorithm increases the number of cryptographic operations it needs to execute, it is a well known fact that the 3DES algorithm is one of the slowest block cipher methods.

The **Advanced Encryption Standard (AES)** is a symmetric block cipher algorithm [2]. These algorithms are the latest generation of block ciphers, and have a significant increase in the block size, 128 bits. AES provides three approved key lengths: 128, 192 and 256 bits. AES is considered fast and able to provide stronger encryption than other well-known encryption algorithms such as DES [10, 25]. Brute force attack (where the attacker tries all the possible key combinations to unlock the encryption) is the only known effective attack known against it.

In [3] an Order Preserving Encryption Scheme (**OPES**) for numeric data is proposed, flattening and transforming plaintext distributions onto target distributions defined from value-based buckets. This solution allows any comparison operation to be directly applied on encrypted data, such as equality and range queries, as well as SUM, AVG, MAX, MIN and COUNT queries. A lightweight database **encryption scheme for column-oriented DBMS** is proposed in [7].

The **Blowfish encryption algorithm** [24] is one of the most common public domain encryption algorithms. It is a 64 bit block cipher, allowing a variable key length. Each round of the algorithm consists of a key-dependent permutation and a key-and-data-dependent substitution. All operations are based on XORs and additions on 32-bit words. The key has a variable length (maximum of 448 bits). Though it suffers from weak keys problem, no attack is known to be successful against it [10].

More recently, the **Snuffle 2005** encryption algorithm (also known as **Salsa20**) was proposed [22]. It is a stream cipher based on a hash function with a long chain of simple operations (32-bit additions, 32-bit XORs, and constant distance 32-bit rotations), instead of a short chain of more complex operations (typical in standard encryption algorithms). Salsa20 produces a 64-bit block given a key, nonce and block counter. Salsa20 simply works by running the hash function in counter mode, generating the keystream by hashing the key with a message based nonce and sequential integers (1, 2, 3, etc) appended. This solution is relatively simple when compared with other standard encryption algorithms and has been recognized by the cryptology research community as an interesting alternative in certain contexts.

Since we focus on discussing if current data encryption algorithms are too slow or not for DWs, we are not interested in discussing the security details of each algorithm, but rather in pointing out their generic guidelines and how their performance is affected. In cryptography, an S-box (Substitution-box) is a basic component of symmetric key algorithms. They are typically used to obscure the relationship between the keys and the generated ciphertext. In general, an S-box takes a number of input bits, m, and transforms them into a number of output bits, n; an $m \times n$ S-box can be implemented as a lookup table with $2m$ words of n bits each. In many cases, the S-boxes are carefully chosen to resist cryptanalysis. Fixed tables are normally used, as in the Data Encryption Standard (DES) [6], but in some ciphers the tables are generated dynamically from the key; *e.g.* the Blowfish encryption algorithm [18].

The argument in favor of using complicated operations such as S-boxes is that a single table lookup can mangle its input quite thoroughly – more thoroughly than a chain of simple integer operations – in fewer rounds. This provides a large amount of mixing at reasonable speed on many CPUs, reaching many desired security levels more quickly than simple operations. The counterargument is that potential speedup is fairly small and is accompanied by huge slowdowns on other CPUs. On the other hand, simple operations such as bit additions and XORs are consistently fast, independently from the CPU. It is also not obvious that a series of S-box lookups (even with rather large S-boxes, as in AES, increasing L1 cache pressure on large CPUs and

forcing different implementation techniques for small CPUs) is generally faster than a comparably complex series of simpler integer operations.

Table 1 shows the number of rounds for achieving minimum and recommended security strength, respectively, along with block size and encryption key lengths, for Salsa20, 3DES and AES. The performance of Salsa20 in the ENCRYPT's test framework reports the speeds (in CPU cycles per encrypted byte) for encrypting a 576-byte packet (or a long stream) on several CPUs [25] and are shown in Table 2. The values for the AES encryption algorithm are from [26].

Table 1. Encryption algorithm variables w/ performance impact

	Salsa20	**3DES**	**AES**
Recommended nr. of rounds	20	16	14
Minimum nr. of rounds	8	12	10
Block size	512 bits	64 bits	128 bits
Encryption key length	128 or 256 bits	168 bits	128 or 256 bits

Table 2. Encryption algorithms CPU Cycles p/ Encrypted Byte

	CPU Cycles p/Encrypted Byte	
CPU/Algorithm	**Salsa20**	**AES128**
AMD64 3GHz Intel Xeon 5160 (6f6)	4.3	9.2
Intel Core 2Duo 2.1GHz (6f6)	4.3	9.2
AMD64 3GHz Intel Pentium D (f64)	11.7	16.2
Intel Pentium 4 3GHz (f41)	13.4	19.8

Other encryption solutions, such as [3], distribute data in well-defined groups to allow direct operations on encrypted data. However, the impact in performance produced by these solutions, in response time and storage space overhead, depends on the skew in the target distributions, which can be a very serious problem in DWs. There is no easy way around this. The proposal from [23] also suffers from the same problem. The lightweight encryption in column-oriented DBMS proposed in [7] aims on providing low decryption overheads. However, their experiments show at least 50% of response time overhead to retrieve the encrypted tuples, which is still extremely high for many DW scenarios, such as long running queries.

Analyzing the features of the referred encryption solutions that influence performance (and security tradeoff), we have found the following conclusions:

- Specific DW encryption solutions still show large performance overheads;
- The type and number of operations for producing the "data mix" output in each round of the algorithm, the length of the used encryption keys, the size of the input and output blocks, and the number of rounds to execute, are all variables that affect both security and performance;
- Typically, a secure encryption algorithm will execute between 8 and 20 rounds against 64, 128 bit (or more) sized blocks, using a 128 or 256 bit key;
- Encryption algorithms which make use of chains of simple operations such as bit additions and XORs scale better and have reduced CPU dependency than solutions that make use of more complex operations such as S-box lookups;
- Salsa20 seems to provide consistent speed in a wide variety of applications across a wide variety of platforms. It is faster and simpler than other complex approaches such as the standard algorithms 3DES and AES, while granting significant security

strength. However, most commercial vendors just include AES and 3DES routines. The AES became a standard only after a five-year long standardization process that included extensive benchmarking on a variety of platforms ranging from smart cards to high end parallel machines. Thus, the adoption of encryption standards is probably only due to legal impositions and public reliability issues.

2.3 Data Masking/Encryption Architectures

There are mainly two types of architecture for data masking and encryption at the database level: 1) masking/unmasking and encryption/decryption is executed by the DBMS server itself directly on the database; or 2) all masking/unmasking and encryption/decryption is executed by a third party, typically an application or service acting as a middleware tier between user applications and the encrypted/masked database. The first type of architecture is typically used with built-in packages provided by DBMS vendors. These routines run in the DBMS kernel and are optimized to work against their data structures and across a large diversity of platforms.

Major DBMS such as Microsoft SQL Server and Oracle provide standard encryption routines. Oracle has developed TDE (Transparent Data Encryption) [11, 13] incorporating both AES and 3DES, providing column and tablespace encryption. These routines can be used transparently without requiring user application source code modifications. As Oracle, Microsoft SQL Server also provides column and datafile 3DES and AES encryption routines.

When using Oracle TDE tablespace encryption, all data in the tablespace's physical datafiles is encrypted and nearly no storage space overhead is generated. When using column encryption, a storage space overhead between 1 and 52 bytes per encrypted value is added. The generation of independently encrypted values for each column is done by using an optional feature (SALT) in the encryption, which implies adding 16 bytes of the storage space per encrypted column to each row. If NO SALT is used, those extra 16 bytes are saved, but all encrypted values in the column rely on one key only in the encryption algorithm. Tablespace encryption uses only the database master key and the tablespace's encryption key, which makes its security level lower than column encryption. Oracle recommends the use of tablespace encryption when there is no way of determining which columns are sensitive and which are not, or when the majority of the data in the tablespace is sensitive [8]. They state that column encryption should be preferred when a small number of well defined columns are sensitive. This last scenario is typical in data warehousing environments, which makes column encryption the recommended solution according to Oracle. However, as referred before, when using TDE column encryption in DWs the storage overhead will be very significant. On the other hand, since DWs store business secrets, we can assume that most of its data is sensitive. In this sense, we may also state that TDE tablespace encryption should also be highly considered.

In most enterprises, data used for analyzing business performance is mostly stored in numerical attributes, called facts [9]. Fact tables typically take up 90% or more of the DW's total storage space [9]. Standard encryption algorithms were designed for general-purpose data. Thus, they were designed for encrypting blocks of text, *i.e.*, sets of character-values by default. This has led DBMS to implement encryption routines that just output textual or binary attributes. Since most DW columns store numerical values, using encryption means they need to be converted to textual format. When the values are decrypted for query processing, they need to be converted back into numerical format in order to process sums, averages, etc. Since most decision support que-

ries process mathematical functions and calculus against numerical attributes, conversion operations are a significant and potentially critical drawback, adding computational overheads with considerable performance impact.

Topologies involving middleware solutions such as [15] typically request the encrypted data from the database *a priori* and execute the decrypting actions themselves locally. The proposal in [15] aims to ensure efficient query execution over encrypted databases, by evaluating most queries at the application server and retrieving only the necessary records from the database server. Only one query (Q6) of the TPC-H benchmark is used in their experimental evaluation, against a very small data subset (ranging from 10MB to 50MB, where query execution time rises up to 5 times for the last). This is not a realistic dataset for DWs. In a DW environment, previously transporting all the required data from the database to the middleware is unreasonable, since the amount of data accessed for processing decision support queries is typically much larger than a few tens of MB. This would strangle the network due to bandwidth consumption of data roundtrips between middleware and database, jeopardizing data throughput and consequently, response time. Thus, all encrypted data should be processed at the DBMS itself, eliminating network overhead from the critical path.

In this sense, we have found the following conclusions:

- All major DBMS provide encryption to be used transparently by user applications;
- When using tablespace encryption, the requested data is decrypted loaded into RAM memory (in the database cache) as clear text, while column encryption does not and is thus more secure;
- Tablespace encryption does not create significant storage space overhead, while column encryption does;
- Despite well-known pros and cons, the best choice between tablespace encryption and column encryption isn't obvious;
- Leading DBMS use standard encryption algorithms AES and 3DES, producing alphanumeric or binary values as a result of the encryption process, even for numerical-typed attributes;
- In DWs, transporting encrypted data to third party decrypting agents would create unbearable communication bandwidth consumption and compromise throughput.

3 EXPERIMENTAL EVALUATION

We implemented the TPC-H decision support benchmark (TPC-H) [17] with 1GB and 10GB scale sizes, and a real-world sales DW with 2GB of storage size. The sales DW has a star schema [9] with four dimension tables and one fact table (Sales). Its dimensional features are shown in Table 3. Each DBMS was installed on separate machines, Pentium 2.8GHz CPU with a 1.5TB SATA hard disk and 2GB RAM, with 512MB of RAM devoted to the database memory cache (SGA). The Oracle machine ran Windows XP Professional, SQL Server ran Microsoft Windows Server 2003. The TPC-H database schema has one fact table (*LineItem*) and seven dimension tables, where four columns of *LineItem* were chosen for encryption (*L_Quantity*, *L_ExtendedPrice*, *L_Tax* and *L_Discount*), given they are the fact columns used in the benchmarks queries to analyze the business. In Sales DW, five numerical fact columns were encrypted (*S_ShipToCost*, *S_Tax*, *S_Quantity*, *S_Profit*, and *S_SalesAmount*), for the same reasons. In our tests, we used the following encryption algorithms: AES with 128 bit and 256 bit keys, and 3DES168 (which uses triple DES with a 168 bit key), provided by

each DBMS, in both tablespace (Tab) and column (Col) encryption modes [8, 11, 13]. Salsa20 [22] and OPES [3] were implemented in C++ and also tested.

Table 3. Dimensional features of the Sales Data Warehouse

	Times	Customers	Products	Promotions	Sales
Number of Rows	8 760	250 000	50 000	89 812	31 536 000
Storage Size	0,12 MB	90 MB	7 MB	10 MB	1 927 MB

3.1 Fact Table Loading Time

Figure 1 shows the loading time overhead percentages concerning of the fact table in the TPC-H 1GB and Sales DW for all the tested scenarios. The results in the TPC-H 10GB scenarios are similar to those of the TPC-H 1GB, with absolute values approximately proportional (10 times bigger), and due to lack of space are not included.

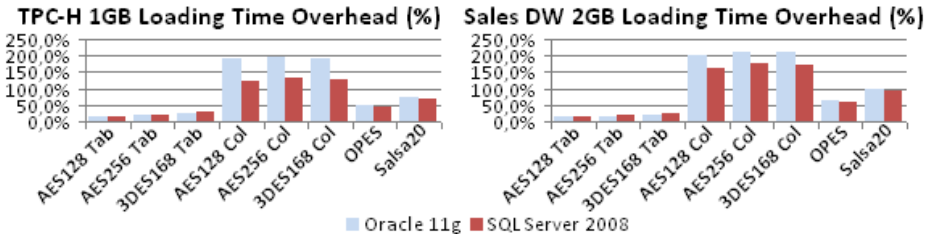

Fig. 1. TPC-H 1GB and Sales DW loading time overheads per DBMS

It can be seen that loading time overheads range from 14,8% (more 46 seconds) to 191,6% (more 594 seconds) in Oracle 11g, from 16,5% (more 35 seconds) to 129,2% (more 274 seconds) in SQL Server 2008, for loading the TPC-H 1GB LineItem fact table (approximately 800MB of original data). The results show that most overheads are indeed very considerable, although tablespace encryption present better results than column encryption, as it would be expected. AES also has better results than 3DES, since it has been proven a faster algorithm. OPES shows overheads of 43,6% and 48,7%, while Salsa20 of 70,4% and 73,3%. OPES and Salsa20 show better results than column encryption, but worse than tablespace encryption.

Loading time overheads for the Sales DW fact table (approximately 1,9GB of data) range from 13,3% (more 159 seconds) to 209,5% (more 2512 seconds) in Oracle 11g, from 15,4% (more 192 seconds) to 171,1% (more 2139 seconds) in SQL Server 2008. As in TPC-H 1GB, the results show that most overheads are indeed very considerable, tablespace encryption presents the best results and column encryption the worst, AES also has better results than 3DES, and OPES better than Salsa20.

3.2 Fact Table Storage Space

Figure 2 shows storage space overhead percentages concerning the fact table in the TPC-H 1GB and Sales DW. As with loading time results, storage space results in the TPC-H 10GB scenarios are similar those of TPC-H 1GB, with absolute values approximately proportional (10 times bigger), and due to lack of space are not included.

Since tablespace encryption affects the datafiles' contents as a whole, it is known that they do not increase storage space (this is because they use a sole key for encrypt-

ing data in the entire datafile as a single entity). OPES also minimally increases storage space, with almost no overhead (ranging from 1,9% to 3,7%), while Salsa20 has a space storage overhead of 26,4% to 94,1%. For column encryption in the TPC-H 1GB fact table, storage space overhead is 153,9% (more 1188MB) and 103,6% (more 800MB) in Oracle 11g, 94,8% (more 1173MB) and 76,3% (more 994MB) in SQL Server 2008. Column encryption storage space overhead for the Sales DW fact table is 461,5% (more 7680MB) and 307,7% (more 5120MB) in Oracle 11g, 591,3% (more 11424MB) and 389,9% (more 7532MB) in SQL Server 2008. The results show that storage space overheads for column encryption are very considerable. As expected, 3DES presents better results than AES because it outputs a smaller block size.

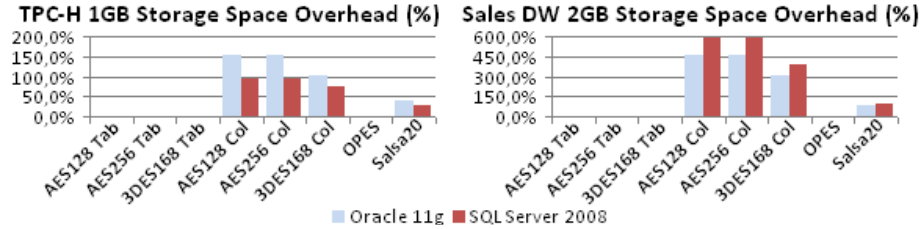

Fig. 2. TPC-H 1GB and Sales DW storage space overheads per DBMS

3.3 OLAP Query Execution

For TPC-H, the test workload was benchmark's queries 1, 3, 6, 7, 8, 10, 12, 14, 15, 17, 19, and 20, all queries accessing the *LineItem* fact table. For the Sales DW, the workload was a set of 29 queries, all processing facts in the *Sales* fact table, representing typical decision support queries such as customer product and promotion sales daily, monthly and annually values, including actions such as selection, joining, aggregates, and ordering. All results shown in this section are the average response time (in seconds) values obtained from six executions (with standard deviations between [0.52, 54.65] and [0.64, 70.10] for 1GB and 10GB TPC-H, respectively, and [0.32, 33.11] for the Sales DW, for individual query response times).

Figure 3 shows total query workload response time overhead percentages in the TPC-H 1GB and the Sales DW. The TPC-H 10GB results are similar to those of the TPC-H 1GB, with absolute values approximately proportional (10 times bigger), and due to lack of space are not included. It can be seen that for TPC-H 1GB the response time overheads range from 28,3% (more 177 seconds) to 203,0% (more 1271 seconds) in Oracle 11g, from 35,2% (more 204 seconds) to 195,2% (more 1132 seconds) in SQL Server 2008. For the Sales DW, response time overheads range from 79,5% (more 497 seconds) to 810,5% (more 5069 seconds) in Oracle 11g, from 82,8% (more 518 seconds) to 758,9% (more 4746 seconds) in SQL Server 2008.

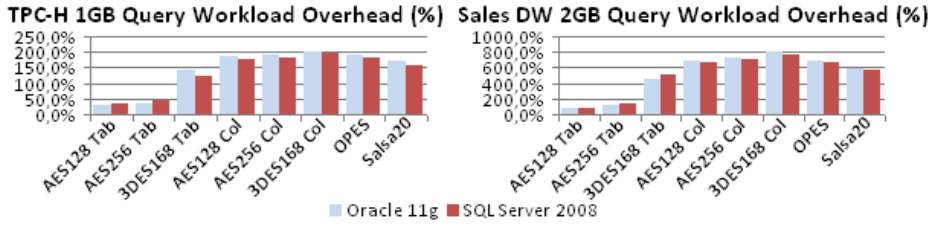

Fig. 3. TPC-H 1GB and Sales DW query workload response time overheads per DBMS

The results show most overheads are extremely high, and that tablespace encryption presents much better performance than column encryption and OPES and Salsa20. AES also has better results than 3DES, since it is a faster algorithm.

3.4 Discussion and Conclusions

Regarding the TPC-H 1GB results, notice that for TPC-H 10GB (which is ten times bigger), since all overheads are approximately proportional, absolute values of storage space, loading and response time overheads are nearly ten times bigger. This means that TPC-H 10GB has approximately 8GB to 12GB of increased storage space, for column encryption scenarios; tablespace encryption does not introduce extra storage space. In what concerns loading time, TPC-H 10GB can expect an extra 6 to 13 minutes to load the data in tablespace encryption, and 44 to 99 minutes for the column encryption scenarios. In the TPC-H 10GB column encryption, OPES and Salsa20 setups, query workload response time rises up from 2 to 3 hours. Given that 10GB is actually a small size for a DW database, it is easy to conjecture that the introduced overheads due to encryption in DWs are extremely significant and may in fact be unacceptable. Although Oracle argues that TDE will only increase response time an average of 5% to 10% [13], this has shown not to be true. The results show that response time overhead is, on average, many orders of magnitude higher. The same occurs in SQL Server. Conclusively, the findings were the following:

- Using encryption does in fact introduce huge storage space, data loading time and query response time overheads;
- Given that decision support environments typically execute long running queries (*i.e.*, queries that run for many minutes up to hours), those response time overheads represent high absolute values that can easily make query responses overdue and jeopardize the usefulness of the DW itself;
- Storage size and data loading time overheads are also very large, mainly in column encryption, with implications in database availability and storage management;
- Although security best practice recommends using column encryption in DW environments, tablespace encryption presents better performance results.

4 RESEARCH CHALLENGES & OPPORTUNITIES

In a traditional DW, data is static, *i.e.*, there is no data loading when the databases are available to its end users. In these environments, the main performance issue is not encryption, but decryption overhead for querying. Since data loading occurs in well-defined time windows in which the database is offline, there is no impact in user query response time; it only affects DW maintenance time. Nevertheless, this static data state paradigm has been changing, with the increasing implementation of real-time DW solutions. Thus, given the size of DWs and the amount of data typically processed by decision support queries, the overheads introduced by both encryption and decryption need to be dealt with, for the sake of their feasibility. The development of future solutions must consider the performance of both encryption/masking and decryption/unmasking as critical.

To improve CPU performance and scalability, using long chains of simple operations instead of short chains of complex operations may allow developing faster solutions while being able to maintain significant security strength, as argued in Salsa20.

The basic argument for increasing the block size of the standard 16 bytes to a higher size of 256 bytes, for example, is that we do not need as many cipher rounds to achieve the same security level. Using a larger block size should provide just as much mixing as the first few cipher rounds and thus, saves time. The basic counterargument is that a larger block size also loses time in CPU models. On most CPUs, the communication cost of sweeping through a 256-byte block is a bottleneck, because they have been designed for computations that do not involve so much data. However, CPU trends show that evolution will allow computing larger amounts of bits. Thus, future algorithms should take advantage of this, increasing the currently typical 128 bit block size. Parallel processing is also a performance booster in speed and scalability.

Some ciphers sacrifice security strength attempting to obtain higher speed. Nowadays, 256 bit keys are used and considered secure, since the computational efforts in trying to break their security are considered nearly impracticable. However, the recent multi-core CPU trends indicate this key length will be rapidly surpassed as hardware processing power evolves. Thus, to avoid rapidly becoming useless, at least 256 bit or higher key lengths should be used in the development of new solutions. Although higher keys should, in principle, bring worse performance, in our opinion the problem is not centered on the key length, but on the used block size and the algorithm itself.

There is always a tradeoff between performance and security; research will probably lead to solutions that are better in database performance, but have less security strength. The main issue is to significantly decrease storage space, resource consumption and response time, while maintaining substantial security strength. A possibility is to develop variable-based dynamic algorithms that enable the user to choose between different key lengths and block sizes, the number of encryption/masking rounds, and other parameters allowing DBAs and application developers to fine tune the security-performance tradeoff's balance according to the specific features and requirements of each DW.

5 CONCLUSIONS

We have presented the available confidentiality solutions for databases and described the performance issues concerning their use in DWs. Experimental evaluations included in state-of-the-art standards and published research show that the storage space and response time overheads introduced by encryption algorithms dramatically degrade database performance to a magnitude that jeopardizes their feasibility in data warehousing environments. Our experiments have also confirmed this.

A data confidentiality solution may be useless if it assures a high level of protection, but is too slow to be considered acceptable in practice. Since database performance is a critical issue in data warehousing scenarios, we conclude that current encryption solutions are not suitable for DWs. DWs function in a well-determined specific environment with tight security, performance and scalability requirements and, therefore, need specific solutions able to cope with these directives. Since there is always a tradeoff between security strength and performance, developing specific data confidentiality solutions for DWs must always balance security requirements with the desire for high performance, *i.e.*, ensuring a strong level of security while keeping database performance acceptable. This is a critical issue and remains a challenge, which makes ground for opportunities in this domain, given the lack of specific solutions for data warehousing environments.

REFERENCES

1. 3DES. Triple DES. National Institute of Standards and Technology (NIST), Federal Information Processing Standards (FIPS), Pub. 800-67, ISO/IEC 18033-3, 2005.
2. AES. Advanced Encryption Standard. NIST, FIPS-197, 2001.
3. Agrawal, R., Kiernan, J., Srikant, R. and Xu, Y. "Order-Preserving Encryption for Numeric Data". ACM SIG Int. Conference on Management Of Data (SIGMOD), 2004.
4. Agrawal, R., Srikant, R. and Thomas, D. "Privacy Preserving OLAP". ACM Int. Conference of the Special Interest Group on Management Of Data (SIGMOD), 2005.
5. Bertino, E. and Sandhu, R. "Database Security – Concepts, Approaches, and Challenges". IEEE Transactions on Dependable and Secure Computing, Vol. 2, No. 1, 2005.
6. DES. Data Encryption Standard. National Inst. of Standards and Technology (NIST), FIPS Pub. 46, 1977.
7. Ge, T. and Zdonik, S. "Fast, Secure Encryption for Indexing in a Column-Oriented DBMS". International Conference on Data Engineering (ICDE), 2007.
8. Huey, P. Oracle Database Security Guide 11g. Oracle Corporation, 2008.
9. Kimball, R. and Ross, M. The Data Warehouse Toolkit. 2nd Ed, Wiley & Sons, Inc., 2002.
10. Nadeem, A. and Javed, M. Y. "A Performance Comparison of Data Encryption Algorithms", Int. Conf. on Information and Communication Technologies (ICICT), 2005.
11. Natan, R. B. Implementing Database Security and Auditing. Digital Press, 2005.
12. Oracle Corporation. "Data Masking Best Practices", Oracle White Paper, 2010.
13. Oracle Corporation. "Oracle Advanced Security Transparent Data Encryption Best Practices", Oracle White Paper, 2010.
14. Procopiuc, C. M. and Srivastava, D. "Efficient Table Anonymization for Aggregate Query Answering". Int. Conf. on Data Engineering (ICDE), 2011.
15. Radha, V. and Kumar, N. H. "EISA – An Enterprise Application Security Solution for Databases". Int. Conf. on Information Systems Security (ICISS), S. Jajodia and C. Mazumdar (Eds), Springer LNCS 3803, 2005.
16. Ravikumar, G. K., Manjunath, T. N., Ravindra, S. H. and Umesh, I. M. "A Survey on Recent Trends, Process and Development in Data Masking for Testing". International Journal of Computer Science Issues, Vol. 8, Issue 2, 2011.
17. TPC-H. The TPC Decision Support Benchmark H. http://www.tpc.org/tpch/default.asp
18. Vimercati, S. C., Foresti, S., Jajodia, S., Paraboschi and Samarati, P. "Over-encryption: Management of Access Control Evolution and Outsourced Data". International Conference on Very Large DataBases (VLDB), 2007.
19. Xiao, X., Bender, G., Hay, M. and Gehrke, J. "iReduct: Differential Privacy with Reduced Relative Errors". ACM SIG Int. Conf. on Management Of Data (SIGMOD), 2009.
20. Yuhanna, N. "Your Enterprise Database Security Strategy 2010", Forrester Research,2009.
21. Gartner Inc. "Selection Criteria for Data-Masking Technologies". Research Report ID G00165388, Feb 2009.
22. Bernstein, D. J. "The Salsa20 Family of Stream Ciphers". New Stream Cipher Designs - The eSTREAM Finalists 2008, Springer LNCS 4986, 2008.
23. Hacigumus, H., Iyer, B., and Mehrotra, S. "Efficient Execution of Aggregation Queries over Encrypted Relational Databases". Int. Conf. on Database Systems for Advanced Applications (DASFAA), 2004.
24. Schneier, B. The Blowfish Encryption Algorithm. http://www.schneier.com/blowfish.html.
25. Elminaam, D., Kader, H., and Hadhoud, M. "Evaluating the Performance of Symmetric Encryption Algorithms", Int. Journal of Network Security, Vol. 10, No. 3, 2010.
26. Bernstein, D. J., and Schwabe, P. "New AES Software Speed Records", International Conference on Cryptology in India (INDOCRYPT), 2010.