Choosing Encryption for Microsoft® SQL Server
<table>
<thead>
<tr>
<th>CONTENTS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Database Security Issues</td>
<td>3</td>
</tr>
<tr>
<td>Balancing Database Security with Usability</td>
<td>5</td>
</tr>
<tr>
<td>Optimizing the Solution</td>
<td>6</td>
</tr>
<tr>
<td>Evaluating Potential Security Solutions</td>
<td>6</td>
</tr>
<tr>
<td>Summary</td>
<td>8</td>
</tr>
</tbody>
</table>
Database Security Issues

The news on the cyber security front is grim. Attacks on enterprises and government agencies of all sizes are common and increasingly successful. At the same time, businesses, their customers, and their employees require access to information from a multitude of devices and locations. Enterprises need to balance productivity and availability with security. As the need for data storage continues to grow, businesses of every size struggle with the costs and complexity of maintaining their stored and rapidly growing data, especially in databases. Whether you're managing data locally, remotely, or in the cloud, securing that data has never been more important.

Market adoption of database encryption has been hindered for three primary reasons:

• Performance
• Complexity
• Cost

An overview of database encryption choices and the inherent issues with each follows below.

Database Encryption Types

There are a number of different database types available to consumers today, such as Relational, Open Source, NoSQL, and Hadoop. They all have different characteristics, each optimized for certain data uses and types.

SQL is a relational database, and as such is very sensitive to additional operations on its stored data that require more processor cycles to complete. For this reason, encryption has been a problem for DBAs and users of SQL for many years. There are three types of traditional encryption solutions for SQL databases; Cell Level Encryption (CLE), Transparent Data Encryption (TDE) and Encrypting File System (EFS). Each have their associated advantages and disadvantages.

Cell Level Encryption:

CLE is designed for environments with very little data to encrypt. It allows you to encrypt only the data that needs to be encrypted. i.e. a single column in a specific row or rows. You can also encrypt entire columns.

There are, however, several significant drawbacks to CLE. You must alter your application code, which could be a prohibitive amount of work. Indexes are not useful on these columns and that can result in poor performance for queries that filter or join on encrypted columns. The retrieve-and-convert process is necessarily slower than direct access of non-encrypted data.
Transparent Data Encryption (TDE):

Microsoft’s Transparent Data Encryption is the built-in option for MS databases, but it is only available in the Enterprise Editions of database products (including SQL Server). Though TDE requires minimal code changes, and encryption is performed on the entire set of data and log files, there are some significant drawbacks. Primarily, TDE renders database compression functions virtually unusable. It also requires management of two types of keys (Service Master Keys and Database Encryption Keys) and encryption certificates, which can be complicated. From a performance perspective, TDE has the least impact to the database environment, but the costs associated with TDE for each database make it an option for very few enterprises.

Encrypting File System (EFS):

EFS is file system level encryption, and is almost exclusively used for unstructured data. It has excellent performance characteristics, but requires significantly more CPU and RAM. You can encrypt at the file, directory or disk level. For databases, it is sometimes used for OLAP applications (On-line Analytical Processing), but rarely for OLTP (On-line Transaction Processing) environments. EFS is rarely used in production environments due to the significant increase in compute power required.

Recent innovations in the data protection/survivability space have enabled a new type of database protection capability that offsets the disadvantages of these traditional solutions. In particular, SPx SHARC from SecurityFirst Corp reduces the performance impacts associated with CLE and TDE without the additional processing and memory requirements demanded by EFS solutions.

Performance

As described above, performance is a critical issue when choosing an encryption solution. Simply put, the encryption operation places a very large burden on the processor (up to a 45% increase) unless it is designed into the operating system at the core/kernel level. Additionally, most encryption products require an external key management system that adds to the operations, a penalty that is often too hard to overcome with most database security solutions.

Complexity

Approaching database security the right way is about more than just persistent data security – it also means properly defining database access policies, ensuring the database’s security controls are correctly configured, audited and enforced. Databases have many different flavors, so similar ideas are implemented in different ways and similar concepts are named differently. Identical database software packages can be dramatically different in operation depending on the DBA who installed and configured it.
Database encryption requires a DBA with significant expertise. Historically, encryption could be a daunting task unless the DBA was comfortable with the certificate process and a proliferation of keys. Get any of these things wrong, and you can create a self-inflicted denial of service to your customers, or worse, lose the data completely.

**Cost**

The last and arguably the most significant issue concerning database security is cost. Costs come in the form of encryption software licenses, key management hardware, consulting to plan and deploy the encryption solution, and specially trained administrators to operate and maintain. Additionally, most enterprises have more than a single type of database, so each will often require customized deployments and individual licenses, etc.

**Balancing Database Security With Usability**

Enterprises and agencies continue to rely heavily on information systems to do business, sharing information internally and externally, enabling better customer and constituent experiences, and enabling employees and partners to get data when they are on edge devices. This new access and collaboration environment has resulted in a similar rise in vulnerabilities to security breaches as they gain productivity and efficiency advantages. There are a number of methods and techniques currently available to protect data, but these have entirely focused on network-centric means, trying to prevent access to data and information, with little regard to protecting the data itself. Data centric protection methods have been overlooked in comparison to firewalls, DDoS and antivirus software, identity management systems, etc. due to the issues listed in the previous section.

Data and information is critical to every organization. Protecting the data itself is an essential part of a truly comprehensive approach, and must include mechanisms for enforcing access control policies for a variety of users. Over time the database security community has developed a number of different techniques and approaches to assure data confidentiality, integrity, and availability, all with limited success. The best solutions have all centered on encryption, but the limiting factor has always been a direct trade-off of usability. If you wanted security, you had to limit access.

You can upgrade your security posture by locking down the data. However, this directly alienates users and impacts their productivity. At what point do we maximize the benefit to both usability and security? That's a complicated question without an easy answer; and of course, the caveat is that it depends on the organization. Every security control requires some kind of trade-off.
Employees just want to do their jobs; customers just want convenience. IT wants them to be happy, but must meet the security needs of the organization, regulations and compliance requirements.

**Optimizing the Solution**

In light of the database security issues and options detailed in this paper along with the usability concerns, there are a couple of options to pursue. It’s clear that for virtually every organization out there it’s not a matter of if they will be attacked, it’s when.

IT can go continue with a “Business as Usual” outlook, but it is clear that nefarious hackers and bad actors are becoming more sophisticated every day. Attacks come in all forms, via the user network (“vertical” threat vectors) and through emerging storage network attacks (“horizontal” threat vectors). The business as usual approach is shown to be ineffective when it comes to protecting the data-- once you’re in, you’re in-- but it is essential for keeping the bad guys out.

On the other side, there is complete data lock down; all data everywhere is encrypted regardless of relative value or need for protection. This has the disadvantages of not only being incredibly expensive, it also devastating productivity. Employees and customers have no ability to effectively take advantage of technology to be more efficient and effective. Additionally, the cost to protect the variety of platforms (SQL, Oracle, NFS, cloud platforms, back-up data, archive, etc.) and their unique applications would quickly lead to a decision to abandon this approach.

A balance is needed. Network-centric defenses must work hand in glove with data centric protection solutions. This dynamic duo must also deliver strong collaboration capability and scale to meet the growth requirements of today’s enterprise and agency.

**Evaluating Potential Security Solutions**

**Complexity**

Any evaluation of solutions for database protection should weigh and measure the following critical aspects to help reduce complexity:

- Works with existing security defenses
- Does not require change to end user experience
- Does not require “rip and replace” of current IT components
- Is cloud ready
- Does not require 3rd party access or control for key security
- Scales to meet edge device adoption and enterprise growth
Cost

From a cost perspective, database security solutions should use the following simple equation to determine relative costs, regardless of specific database product (within the same database type). These costs are by no means exhaustive, but a database security solution should account for these aspects. The components of each factor are given in more detail below.

\[
\text{HARDWARE} + \text{SOFTWARE} + \text{SERVICES} + \text{MAINTENANCE} = \text{TCO}
\]

- **HARDWARE** = Server, Key Management System, Off-site key bunker
- **SOFTWARE** = Encryption Software License, Server License(s), Key Management License
- **SERVICES** = Planning, Discovery, Baseline, Code Alterations, Implementation Testing, Operation
- **MAINTENANCE** = Key Management System, Server, Key Bunkering System and Maintenance (3rd party)

If a third party consultant is not brought on to maintain and operate the solution, training will be required, usually at an additional cost.

Performance

When evaluating an encryption solution, performance is often the most significant factor in determining its usefulness. Ensuring employees and customers have quick access to the information they need, even when it is protected, is priority number one in many IT environments. With that in mind, it is important that an SQL database security solution:

- Has no impact on the inherent functionality of the SQL database
- Has no measurable impact on the day-to-day use of data, and
- Does not create an I/O bottleneck
Summary

Cyber security threats, privacy concerns and compliance requirements have generated increased interest in data security, but market adoption of database encryption has been blocked for three primary reasons:

- Cost
- Complexity
- Performance Impact

<table>
<thead>
<tr>
<th>Protection Type</th>
<th>Complexity</th>
<th>Performance</th>
<th>Cost</th>
<th>Protection Profile</th>
<th>TCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLE</td>
<td>F</td>
<td>D</td>
<td>F</td>
<td>B</td>
<td>Very High</td>
</tr>
<tr>
<td>TDE</td>
<td>D</td>
<td>B</td>
<td>C</td>
<td>C</td>
<td>High</td>
</tr>
<tr>
<td>EFS</td>
<td>B</td>
<td>D</td>
<td>B</td>
<td>D</td>
<td>Low</td>
</tr>
</tbody>
</table>

As demonstrated in the table above, there haven’t been good choices for database encryption. Of the three types of data protection traditionally available for database applications (CLE, TDE, and EFS) TDE is the most frequently used, but it is also very expensive and has significant restraints involved with its use. Usability requirements based on data access, employee mobility and customer convenience have to be balanced with security needs to ensure data is protected and available. In order to decide the best solution for an IT environment, a simple TCO calculation can be made that applies to any solution or product in the marketplace today:

**HARDWARE + SOFTWARE + SERVICES + MAINTAINENCE = TCO**

SPxSHARC™ is software-defined security that combines provably secure data encryption with cryptographic splitting at the bit level to render data invulnerable and impervious to today’s cyber security threats. SPxSHARC’s simplified key management, discrete protection footprint, transparent operation and affordable cost make database protection available to everyone from small business owners to global enterprises. It can cut comparative TCO measures by 50% or more based on the IT environment.

<table>
<thead>
<tr>
<th>Protection Type</th>
<th>Complexity</th>
<th>Performance</th>
<th>Cost</th>
<th>Protection Profile</th>
<th>TCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPx SHARC</td>
<td>A</td>
<td>B</td>
<td>A</td>
<td>B</td>
<td>Very Low</td>
</tr>
</tbody>
</table>

Full functional testing of MS SQL and SPxSHARC was performed to include configurations on separate servers, load testing and failure testing. SPxSHARC was able to operate seamlessly within the SQL environment, and scaled to easily meet performance loads.

SPxSHARC should be a consideration for any data protection requirement in an SQL environment, especially if there is any additional need for cloud enablement, disaster recovery or business continuity.