Data Insecurity: A Perspective on Data Encryption

Ron Johnson

Introduction

Significant data breaches resulting in the release of personal identifiable information (PII) are regularly reported in the news media: *Laptop with NASA worker's personal data is stolen*, *State now says 3.8 million tax records were hacked* and '*Wall of Shame' exposes 21M medical records*. (McLeod, 2012) (Klotz, 2012) (Mearian, 2012) The headlining incidents are merely exemplars of a much broader issue – data insecurity – which is now being brought to light as a result of data breach reporting requirements (for a comprehensive list of data breaches the reader is referred to: The Privacy Rights Clearinghouse). In addition to the headline-making incidents there are dozens more that do not garner such attention. Howard Schmidt, a recognized security expert, estimates there have been 564 million records released in data breaches in the U.S. (Weisbaum, 2012)

To be clear, data breaches are not solely the result of insecure systems as there are a percentage of data breaches that are the result of lost or stolen computers, backup media, and other storage devices. Regardless of the cause a data breach is defined as "the intentional or unintentional release of secure information to an untrusted environment." (Wikipedia, 2012) Data security efforts attempt to protect Personal Identifying Information (PII) data that identifies a particular individual such as Social Security numbers that identify by themselves or bits of information such as maiden name, employment address, date-of-birth, place-of-birth, address, phone number, and other personal information that when aggregated together provide an individual's identity. (Sprague & Ciocchetti, 2009)

It is important to recognize that the domain of data security encompasses storage through transport; for example, database encryption ensures the security of the data while "at-rest" (as well as "in-memory") and network-layer security ensures the security of the data while "in-motion". The focus of this paper is on the protection of PII data stored within a database using encryption technologies with the assumption that the SQL Server is hardened and that regular security audits are performed as part of an overall system security plan.

Background

The first step in developing a data security plan is to perform a data inventory to understand what data exists in the database, the data types, and to develop a set of rules for classification of the data. The process of data classification for encryption is similar to the classification methodologies used for defining data access levels with, in my opinion, higher-level classes. Therefore, if your organization has already defined and implemented data access classifications then there is no need to redefine data classification levels to segment the data for encryption.

Unless predefined by a client I use three classifications when segmenting the data:

> ***Public***: data that could be publically released without compromising individual privacy or proprietary company information.

Sensitive: data that cannot be publically released without compromising individual privacy or company competitiveness or data whose protection is required by regulation.

Secret: data that is business critical with limited distribution within the company or PII data this is self-identifying such as, National ID.

There are many alternative data classification methodologies available to which the reader is referred for further study.

Developing a robust key management process is essential to the success of the encryption project. The encryption keys are as important if not more important than the data that the keys are protecting; therefore, encryption key security is paramount. Encryption key management includes limiting distribution and access of the encryption key to only those individuals who have a defined need to access the encrypted data. Additionally, key management requires back-up and key rotation on a regular basis. Both ANSI and ISO have developed models for encryption key management providing best practices to ensure the security of the protected data as well as the encryption keys to which the reader is referred.

Microsoft Windows provides support for database encryption by leveraging the Data Protection API (DPAPI) which protects the Service Master Key (SMK) which, in turn, protects the Database Master Key (DMK) which, in turn, is used to protect the certificate or asymmetric keys which, in turn, are used to protect the Database Encryption Key (DEK). These dependencies create a security chain from the operating system to the data eliminating user interaction thus strengthening security.

As a planning guide the reader is referred to my previous paper Transparent Database Encryption in SQL Server: A Planning Guide which is, obviously, targeted for TDE implementation but, nonetheless, contains useful planning suggestions regardless of encryption mechanism.

Example Setup

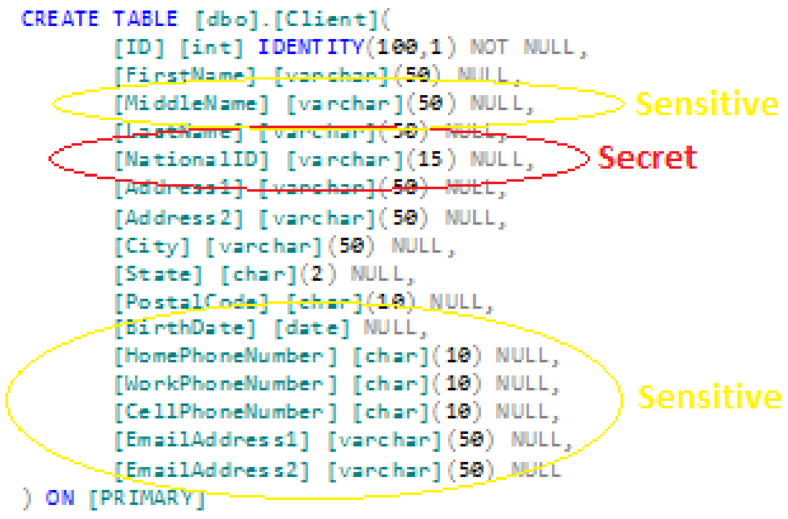
As a "real-world" example we will create a database containing a single table: *Client* which contains PII that must be protected. Using the created database the example will demonstrate and document the process for implementing TDE and Column-level encryption. The steps for creating the example environment are listed below:

Step #1 Create the database

```
CREATE DATABASE [EncryptionTest]
```

Step #2 Create the table

Using the three (3) classification categories discussed above I designate the *NationalID* and other data fields to their respective security classifications. This example will only manage the security of the **Secret** data. The *Client* table is created with the following script:



```
CREATE TABLE [dbo].[Client](
        [ID] [int] IDENTITY(100,1) NOT NULL,
        [FirstName] [varchar](50) NULL,
        [MiddleName] [varchar](50) NULL,
        [LastName] [varchar](50) NULL,
        [NationalID] [varchar](15) NULL,
        [Address1] [varchar](50) NULL,
        [Address2] [varchar](50) NULL,
        [City] [varchar](50) NULL,
        [State] [char](2) NULL,
        [PostalCode] [char](10) NULL,
        [BirthDate] [date] NULL,
        [HomePhoneNumber] [char](10) NULL,
        [WorkPhoneNumber] [char](10) NULL,
        [CellPhoneNumber] [char](10) NULL,
        [EmailAddress1] [varchar](50) NULL,
        [EmailAddress2] [varchar](50) NULL
) ON [PRIMARY]
```

Comment

The first issue that should be recognized is the poor design of the *Client* table. In a more appropriate design the data contained in the *Client* table would be distributed by security classification across multiple tables and potentially multiple servers. The simplified table design is used to maintain the focus of this paper on the implementation of encryption rather than on the myriad of ancillary considerations that should be taken into account in a data security project.

Furthermore, an explanation regarding the classification of the data is warranted; whereas the classification of the *NationalID* as **Secret** is intuitive (or, should be; apparently it isn't in much of the government), the classification of *MiddleName*, *BirthDate*, etc. as **Sensitive** may not be as intuitive. This ancillary data needs to be protected as **Sensitive** based on research conducted by the author and others demonstrating the ease with which **Secret** information about an individual (i.e. Social Security Number) may be derived from innocuous, seemingly unrelated information. (Acquisti & Gross, 2009)

Step #3 Populate the table

The *Client* table is populated with data using a "random" string generator. Feel free to adjust the counts as you see fit, in this example the table is populated with 1M rows.

```
DECLARE @cnt INT
DECLARE @ten INT
DECLARE @nine INT
DECLARE @seven INT
DECLARE @six INT

SET @six = 100000
SET @seven = 1000000
SET @nine = 100000000
SET @ten = 1000000000
SET @cnt = 0
```

```
WHILE @cnt < @seven
  BEGIN
      INSERT INTO CLIENT
      SELECT dbo.udf_StringGenerator('A', 8),
             dbo.udf_StringGenerator('P', 6),
             dbo.udf_StringGenerator('A', 12),
             Cast(Rand(Checksum(NewID ())) * @nine AS INT),
             dbo.udf_StringGenerator('A', 7),
             dbo.udf_StringGenerator('A', 5),
             dbo.udf_StringGenerator('A', 8),
             dbo.udf_StringGenerator('A', 2),
             Cast(Rand(Checksum(NewID())) * @six AS INT),
             GetDate() - ( ( 21 * 365 ) + Rand() * ( 39 * 365 ) ),
             Cast(Rand(Checksum(NewID ())) * @ten AS INT),
             Cast(Rand(Checksum(NewID ())) * @ten AS INT),
             Cast(Rand(Checksum(NewID ())) * @ten AS INT),
             dbo.udf_StringGenerator('A', 8) + '@'
             + dbo.udf_StringGenerator('A', 4) + '.com',
             dbo.udf_StringGenerator('A', 6) + '@'
             + dbo.udf_StringGenerator('A', 6) + '.com'

      SET @cnt = @cnt + 1
  END
```

Note: the udf_StringGenerator function was developed by Vadivel Mohanakrishnan and is included for reference in Appendix A

Transparent Database Encryption (TDE) Example

TDE implementation is simple and straightforward; its simplicity belies its strength in protecting a database “at-rest”. It should be noted that, although TDE may meet regulatory and / or in-house data security requirements, it does not provide a complete solution because the data it protects is decrypted when moved into memory. In short, the data is stored on disk with encryption but when moved to memory is in plain text – the security ramifications should be clear. The steps for encrypting the *EncryptionTest* database are listed below:

```
/*
Create Database MASTER KEY
*/
USE MASTER;
GO
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'PLZ_D0nt-H@ck_M3';
/*
Create Certificate
*/
USE MASTER
GO
CREATE CERTIFICATE EncryptionTestCert WITH SUBJECT = 'Encryption Test Certificate';
GO
/*
Create the Encryption Key
*/
USE EncryptionTest
GO
CREATE DATABASE ENCRYPTION KEY WITH ALGORITHM = TRIPLE_DES_3KEY ENCRYPTION BY SERVER
CERTIFICATE EncryptionTestCert;
```

```
/*
Turn on Transparent Data Encryption
*/
USE [EncryptionTest]
GO
ALTER DATABASE [EncryptionTest] SET ENCRYPTION ON;
GO
```

Column-level Encryption Example

In contrast to TDE, Column-level encryption allows for encryption, as the name implies, of column data whose security is maintained when the data is moved into memory. However, Column-level encryption requires that the data be stored as type *VARBINARY* necessitating, in most cases, application modifications. The example below demonstrates the ease-of-use of the Column-level encryption mechanism. The complete steps for encrypting the *NationalID* column in the example table are listed below (note: the Encryption Key is protected by a password rather than a generated certificate as in the TDE example) :

```
/*
Create Database MASTER KEY
*/
USE EncryptionTest;
GO
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'PLZ_D0nt-H@ck_M3';
/*
Create SYMMETRIC KEY
*/
USE EncryptionTest;
GO
CREATE SYMMETRIC KEY SecretDataKey WITH ALGORITHM = TRIPLE_DES_3KEY
ENCRYPTION BY PASSWORD = 'PLZ_D0nt-H@ck_M3';
/*
Create a column for the encrypted value
*/
USE EncryptionTest;
GO
ALTER TABLE [dbo].[Client] ADD EncryptedNationalID VARBINARY(MAX) NULL
/*
open the SYMMETRIC KEY
*/
USE EncryptionTest;
GO
OPEN SYMMETRIC KEY SecretDataKey DECRYPTION BY PASSWORD = 'PLZ_D0nt-H@ck_M3';
GO
/*
Encrypt the NationalID data
*/
UPDATE [dbo].[Client] SET [EncryptedNationalID] =
EncryptByKey(Key_GUID('SecretDataKey'),[NationalID])
FROM [dbo].[Client];
GO
/*
Close SYMMETRIC KEY
*/
CLOSE SYMMETRIC KEY SecretDataKey;
GO
```

Discussion

Any encryption process is computationally intensive; TDE and Column-level encryption are no exceptions. In my experience TDE results, as Microsoft claims, in a less than ten percent (10%) performance penalty; however, Column-level encryption has, in some situations, resulted in as much as a twenty-five percent (25%) performance penalty. In addition to the cryptographic performance penalty, data encrypted using Column-level encryption cannot be indexed and, as is obvious from the example, must be stored as and cast to / from a *VARBINARY* type.

So, which approach is best? Neither. Both.

A comprehensive data security plan would include the use of Column-level encryption for granular protection while TDE would be used for full scope protection.

Conclusion

The good news is that the reader now knows what the South Carolina Department of Revenue only recently discovered - the importance of encryption. According to their spokesperson, "The state agency is now working to encrypt taxpayers' Social Security numbers" (McLeod, 2012). Too late for the 3.8 million South Carolina taxpayers (81% of the 4.7 million total residents) who have had their Social Security Numbers and other PII released. However, as those involved in the PII data community from government officials, data scientists, to hackers know all too well releases of the kind suffered in South Carolina are not a primary source of PII. For hackers; in fact, the PII of almost any individual may be purchased from nefarious Eastern European and Chinese websites or derived from publically available information.

Comments by two technology pioneers help to define an appropriate perspective regarding PII: "The privacy you're concerned about is largely an illusion. All you have to give up is your illusions, not any of your privacy." (Larry Ellison) "You have zero privacy anyway. Get over it." (Scott McNealy). In the interest of full disclosure I must say that I agree with both statements with regards to personal privacy or, more precisely, the lack thereof. The statements and my own beliefs are well-supported in studies concluding that the publically available information collected on every American by data aggregators (and others) as well as self-published on social networking sites provides a computationally trivial problem for deriving personal identifying information such as Social Security Number. (Hotaling, 2008) (Krishnamurthy & Wills, 2009) (Smith, 2005) (Acquisti & Gross, 2009) For a complete discussion of this unrelated yet interesting topic visit the Identity Theft Awareness and Prevention site (IDTAP.org).

Notwithstanding my personal opinion, as a DBA, it is my responsibility to ensure that my employer **NEVER** makes the Data Breach list. To that end I am a disciple of, and evangelist for, encryption of all PII data stored in my employers' databases. Although this paper does not provide detailed discussion of all aspects of developing a data security plan, the discussion and example provided do demonstrate the ease with which data can be encrypted. As the saying goes, the devil is in the details - key management, key security, application modifications, and other issues must be addressed.

About the Author

Ron is a Senior DBA (MCDBA) who specializes in performance optimization, replication, and security.

Bibliography

Acquisti, A., & Gross, R. (2009, July). Predicting Social Security numbers from public data. *Proceedings of the National Academy of Sciences of the United States of America*.

Hotaling, A. (2008). Protecting Personally Identifiable Information on the Internet: Notice and Consent in the Age of Behavioral Targeting. *CommLaw Conspectus: Journal of Communications Law and Policy, 16*(2), 529.

Klotz, I. (2012, November 14). *Laptop with NASA workers' personal data is stolen*. Retrieved November 15, 2012, from NBCNEWS.com: http://www.nbcnews.com/technology/technolog/laptop-nasa-workers-personal-data-stolen-1C7079170

Krishnamurthy, B., & Wills, C. E. (2009). On the leakage of personally identifiable information via online social networks. *Proceeding WOSN '09 Proceedings of the 2nd ACM workshop on Online social networks* (pp. 7-12). New York: Association for Computing Machinery.

McLeod, H. (2012, November 8). *State now says 3.8 million tax records were hacked .* Retrieved November 8, 2012, from NBC News - Technology: state-now-says-3-8-million-tax-records-were-hacked-1C6891691

Mearian, L. (2012, August 7). *'Wall of Shame' exposes 21M medical record breaches.* Retrieved August 8, 2012, from COMPUTERWORLD.com: http://www.computerworld.com/s/article/9230028/_Wall_of_Shame_exposes_21M_medical_record_breaches

Smith, M. S. (2005). *Identity Theft: The Internet Connection.* Washington, D.C.: Congressional Reserch Service.

Sprague, R., & Ciocchetti, C. (2009, June 10). Preserving Identities: Protecting Personal Identifying Information through Enhanced Privacy Policies and Laws. *Albany Law Journal of Science and Technology, 19*(1).

Weisbaum, H. (2012, October 19). *Federal agencies don't do enough to protect your data.* Retrieved October 19, 2012, from NBCNEWS.com: http://www.nbcnews.com/business/federal-agencies-dont-do-enough-protect-your-data-1C6563242

Wikipedia. (2012, October 16). Retrieved October 20, 2012, from Wikipedia: http://en.wikipedia.org/wiki/Data_breach

Appendix A

```
-- =============================================
-- Author:           Vadivel Mohanakrishnan
-- URL:       http://vadivel.blogspot.com
-- Create date: Oct 8th, 2011
-- Description:      Function to return random string.
-- =============================================
CREATE FUNCTION [dbo].[udf_StringGenerator]
(
        -- A = only Alphabets,
        -- AN = alpha numeric,
        -- P = AN + special characters
       @Type VARCHAR(2),
       @MaxLength INT
)
RETURNS VARCHAR(500)
AS
BEGIN
       DECLARE @randomString VARCHAR(500)

       DECLARE @counter SMALLINT
       DECLARE @Length INT
       DECLARE @strPattern VARCHAR(150)
       DECLARE @isType VARCHAR(2)
       DECLARE @rand REAL

       SET @isType = UPPER(@Type)

       SELECT
         @strPattern = CASE
              WHEN @isType = 'A' THEN 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz'
              WHEN @isType = 'P' THEN 'ABCDEFGHIJKLMNOPQRST
UVWXYZabcdefghijklmnopqrstuvwxyz0123456789 -=+&$'
              WHEN @isType = 'AN' THEN 'ABCDEFGHIJKLMNOPQRST
UVWXYZabcdefghijklmnopqrstuvwxyz0123456789'
         END

       SET @Length = LEN(@strPattern)
       SET @randomString = ''
       SET @counter = 1

       WHILE @counter <= @MaxLength
       BEGIN
                     SET @rand = (SELECT rnd FROM [dbo].[vw_Rand])
                     SET @randomString = @randomString + SUBSTRING(@strPattern,
CONVERT(TINYINT, ((@Length - 1) * @rand + 1)), 1)
                     SET @counter = @counter + 1
       END

       RETURN @randomString
END

GO
```

Glossary of Security / Encryption Terms

Authorization – The official management decision given by a senior agency official to authorize operation of an information system and to explicitly accept the risk to agency operations (including mission, functions, image, or reputation), agency assets, or individuals, based on the implementation of an agreed-upon set of security controls.
SOURCE: SP 800-37

Authenticate – To confirm the identity of an entity when that identity is presented.
SOURCE: SP 800-32

Authentication – Verifying the identity of a user, process, or device, often as a prerequisite to allowing access to resources in an information system.
SOURCE: SP 800-53; FIPS 200

FIPS - Federal Information Processing Standard

IT Security Goals – The five security goals are confidentiality, availability, integrity, accountability, and assurance.
SOURCE: SP 800-27A

Information Security – Protecting information and information systems from unauthorized access, use, disclosure, disruption, modification, or destruction in order to provide—
1) integrity, which means guarding against improper information modification or destruction, and includes ensuring information nonrepudiation and authenticity;
2) confidentiality, which means preserving authorized restrictions on access and disclosure, including means for protecting personal privacy and proprietary information; and
3) availability, which means ensuring timely and reliable access to and use of information.
SOURCE: SP 800-66; 44 U.S.C., Sec 3541

Key – A value used to control cryptographic operations, such as decryption, encryption, signature generation or signature verification.
SOURCE: SP 800-63

Key Establishment – The process by which cryptographic keys are

securely distributed among cryptographic modules
using manual transport methods (e.g., key loaders),
automated methods (e.g., key transport and/or key
agreement protocols), or a combination of
automated and manual methods (consists of key
transport plus key agreement).
SOURCE: FIPS 140-2

Key Management – The activities involving the handling of
cryptographic keys and other related security
parameters (e.g., IVs and passwords) during the
entire life cycle of the keys, including their
generation, storage, establishment, entry and
output, and zeroization.
SOURCE: FIPS 140-2

Principal – An entity whose identity can be
authenticated.
SOURCE: FIPS 196

Private Key – A cryptographic key, used with a public
key cryptographic algorithm, that is
uniquely associated with an entity and
is not made public. In an asymmetric
(public) cryptosystem, the private key
is associated with a public key.
Depending on the algorithm, the private
key may be used to—
1) Compute the corresponding public
key,
2) Compute a digital signature that may
be verified by the corresponding public
key,
3) Decrypt data that was encrypted by
the corresponding public key, or
4) Compute a piece of common shared
data, together with other information.
SOURCE: SP 800-57

Privileged Accounts – Individuals who have access to set "access
rights" for users on a given system. Sometimes
referred to as system or network administrative
accounts.
SOURCE: SP 800-12

Public Key – A cryptographic key that is used with a public key
cryptographic algorithm. The public key is
uniquely associated with an entity and may be
made public. In an asymmetric (public)
cryptosystem, the public key is associated with a

private key. The public key may be
known by anyone and, depending on the
algorithm, may be used to—
1) Verify a digital signature that is signed by the
corresponding private key,
2) Encrypt data that can be decrypted by the
Transparent Database Encryption in SQL Server Page 10
Ron Johnson
corresponding private key, or
3) Compute a piece of shared data.
SOURCE: SP 800-57
Public Key – A cryptographic key used with a public key
cryptographic algorithm, uniquely associated with
an entity, and which may be made public; it is used
to verify a digital signature; this key is
mathematically linked with a corresponding private
key.
SOURCE: FIPS 196

Public Key Certificate – A set of data that unambiguously identifies an
entity, contains the entity's public key, and is
digitally signed by a trusted third party
(certification authority).
SOURCE: FIPS 196

Public Key (Asymmetric) Cryptographic Algorithm – Public key cryptography uses "key pairs," a
public
key and a mathematically related private key.
Given the public key, it is infeasible to find the
private key. The private key is kept secret while the
public key may be shared with others. A message
encrypted with the public key can only be
decrypted with the private key. A message can be
digitally signed with the private key, and anyone
can verify the signature with the public key.
SOURCE: SP 800-46

Public Key Infrastructure – (PKI)
A set of policies, processes, server platforms,
software and workstations used for the purpose of
administering certificates and public-private key
Transparent Database Encryption in SQL Server Page 11
Ron Johnson
pairs, including the ability to issue, maintain, and
revoke public key certificates.
SOURCE: SP 800-32

Public Key Infrastructure – An architecture which is used to bind public keys
to entities, enable other entities to verify public key
bindings, revoke such bindings, and provide other
services critical to managing public keys.

SOURCE: FIPS 196

Secret Key – A cryptographic key that is used with a secret key
(symmetric) cryptographic algorithm, that is
uniquely associated with one or more entities and is
not be made public. The use of the term “secret” in
this context does not imply a classification level,
but rather implies the need to protect the key from
disclosure.
SOURCE: SP 800-57

Secret (Symmetric) Key Encryption –
This is the traditional method used for encryption.
The same key is used for both encryption and
decryption. Only the party or parties that exchange
secret messages know the secret key. The biggest
problem with symmetric key encryption is securely
distributing the keys. Public key techniques are
now often used to distribute the symmetric keys.
SOURCE: SP 800-46

Security Requirements – Requirements levied on an information system that
are derived from laws, executive orders, directives,
policies, instructions, regulations, or organizational
(mission) needs to ensure the confidentiality,
integrity, and availability of the information being
processed, stored, or transmitted.
SOURCE: SP 800-53

Symmetric Key – A cryptographic key that is used to perform both
the cryptographic operation and its inverse, for
example to encrypt and decrypt, or create a
message authentication code and to verify the code.
SOURCE: SP 800-63

Verification – The process of affirming that a claimed identity is
correct by comparing the offered claims of
identity with previously proven information
stored in the identity card or PIV system. See
Identity Verification.
SOURCE: FIPS 201