הצפנת נתונים ב-MS-SQL

מאת: נצר רודנפלד

הקדמה

קיימות כיום בשוק שתי תוכנות עיקריות מסוג "מערכת לניהול בסיסי נתונים": Oracle Database (DBMS/RDBMS) של חברת אורקל ו-SQL Server Management Studio (SSMS), או בשמה העממי יותר: MS-SQL Server של חברת מייקרוספט.

על ההבדלים בין התוכנות ואיזו מהן נחשבת יותר טובה אפשר לכתוב ספרים שלמים אבל ממש בקצרה אפשר להגיד שאורקל נחשבת בעלת ביצועים טובים יותר אך SQL Server נוחה יותר לשימוש ולתפעול. כמובן, יהיו רבים וטובים שיחלקו על האמירה האחרונה (מספיק לכתוב בגוגל "SQL Server VS. oracle" ולראות כמה תוצאות שונות תקבלו).

מטרת המאמר הזה היא לבחון את אפשרויות ההצפנה הקיימות ב-SQL Server בכלל ובפרט פיצ'ר "חדש" יחסית שיצא בגירסה של SQL Server 2008 Enterprise. נציג את כל תהליך ההצפנה - שאילתות, סקריפטים, הצגת המידע לפני ההצפנה ואחריה, וכן בעיות נפוצות בשימוש במנגנוני ההצפנה השונים.

התקנת SQL Server management server:

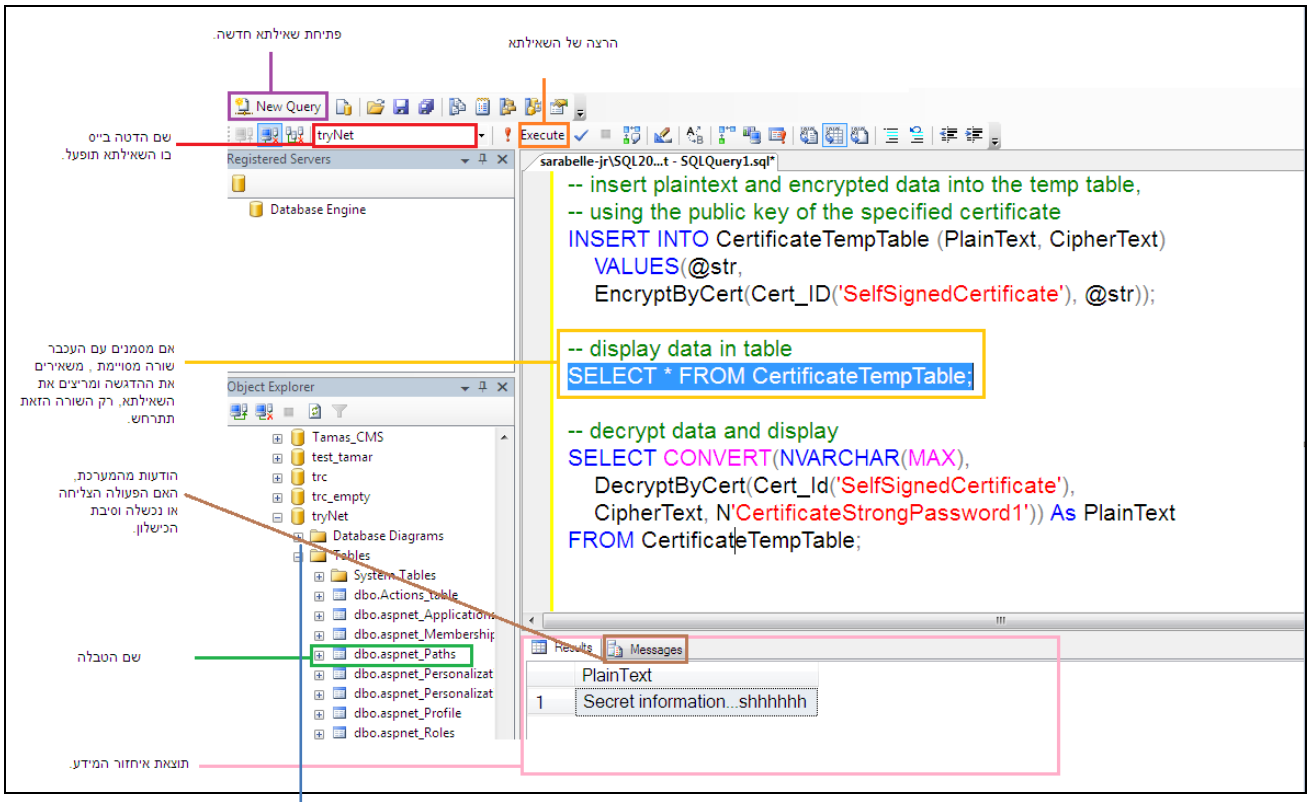
הגרסה הזאת אפשרית להורדה ולשימוש ל-3 חודשים. לאחר שלושת החודשים ה-SQL Server הופך להיות מסוג express ועדיין אפשר להשתמש בו אבל בגירסה "רזה":

http://www.microsoft.com/downloads/en/details.aspx?FamilyID=265f08bc-1874-4c81-83d8-0d48dbce6297&displaylang=en

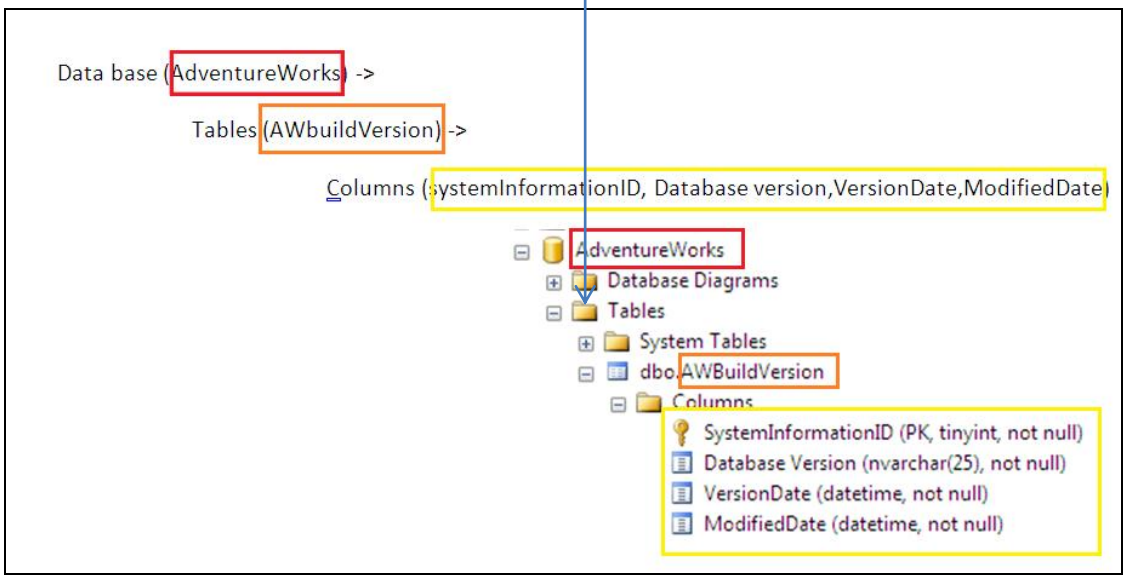
חשוב לזכור שלפני כן יש להוריד את SQL Server 2008 service pack 1 ולהתקין אותו עוד לפני ההתקנה של ה-SQL Server עצמו:

http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=20302

SSMS הוא כלי רב עוצמה וניתן לבצע בעזרתו פעולות רבות, אך אנו נתמקד רק במה שרלוונטי למאמר הזה:



בחלון של ה-Object Exporer ניתן לראות את ההיררכיה של המידע. ההיררכיה של המידע בנויה באופן הבא:



סוגי ההצפנה

הצפנה בעזרת סיסמה (EncryptByPassPhrase):

ההצפנה הראשונה שנכיר במאמר זה היא הצפנה ברמת המחרוזת הספציפית, בעזרת שימוש בפונקציה המובנת "EncryptByPassPhrase". הפונקציה מקבלת כקלט שני פרמטרים: מחרוזת שבעזרתה היא תחולל את מפתח ההצפנה ("PassPhrase") ואת המידע אותו אנו מעוניינים להצפין. הפונקציה תחולל מפתח באורך 128bit, ותצפין איתו את המידע בעזרת אלגוריתם ההצפנה Triple DES.

לשם הדוגמא ניצור טבלה:

```
create table MyPassTable(
      data varchar(122),
      password varchar(122),
      encryptedData varbinary(max)
)
```

- **encryptedData** - ישמש לאיחסון המידע המוצפן.
- **Data** - המידע שאותו נרצה להצפין.
- **Password** - מפתח ההצפנה שבעזרתו נשתמש בעת ההצפנה.

נזין את הטבלה בצורה ידנית:

data	password	encryptedData
משה	13234 487974	NULL
יצחק	79846 468749	NULL
Alice	שדכ1654 fasf	NULL

ניתן לראות שהשדה encryptedData ריק. כעת, נצפין אותה:

```
UPDATE MyPassTable
   SET  encryptedData = EncryptByPassPhrase(password, data)
```

- **EncryptByPassPhrase(password, data)** - הפונקציה להצפנה בעזרת סיסמה, הפרמטר הראשון הוא הצופן, והשני הוא המידע אותו נרצה להצפין. לדוגמא נרצה להצפין את "משה" בעזרת הסיסמה "13234 48794".

והתוצאה:

	data	password	encryptedData
1	משה	13234 487974	0x01000000751351067867 45DBF72AB8ACB58D520E22129F92949FB99B
2	יצחק	79846 468749	0x010000009C968435E51325D989E0F19379A1B8CBE02D0C66AE778A05
3	Alice	שדכ1654 fasf	0x010000005F48579C02869DF0D93091FFD16EB521ED351529EB68B2C8

כעת, נפענח את התוצאה (EncryptedData), באופן הבא:

```
select encryptedData,
convert(varchar,decryptByPassPhrase(passward,encryptedData)) as
decrypted
from MyPassTable
```

- **encryptedData** - המידע שהוצפן.
- **Convert** - המרה מ-varbinary (מופע בינארי) ל-varchar (מופע שמתאים למחרוזת).
- **decryptByPassPhrase(password, encryptedData)** - פונקציית הפיענוח, מקבלת שני פרמטרים, הראשון הוא מפתח ההצפנה שבעזרתו יפוענח המידע, והשני זה המידע שברצוננו לפענח.

והתוצאה:

	encryptedData	decrypted
1	0x0100000075135106786745DBF72AB8ACB58D520E22129F92949FB99B	משה
2	0x010000009C968435E51325D989E0F19379A1B8CBE02D0C66AE778A05	יצחק
3	0x010000005F48579C02869DF0D93091FFD16EB521ED351529EB68B2C8	Alice

ניתן לראות כי המידע אכן פוענח בהצלחה.

דוגמאות לחולשות בהצפנה:

- **מעקב בעזרת Profiler:**

ה-Profiler הוא כלי לניהול, אשר קיים בתוך ה-SQL Server ומאפשר ניטור של כלל השאילתות והפעולות שמתבצעות ברקע. משתמש עם הרשאה מתאימה למחשב אך בלי הרשאה למידע המוצפן יוכל לעקוב ולגלות את הפעלת הפונקציה בידי המשתמש שיצפין את המידע (הפעלת פונקציית ההצפנה כוללת גם את כתיבת הסיסמה ב-Hard Code או את שליחתה מפונקציה חיצונית).

לשם הדוגמה הקמתי אתר על השרת הביתי ובו עמוד הזדהות המתקשר מול מסד נתונים:

לפני מילוי הפרטים, התחברתי עם ה-Profiler לשרת והפעלתי New Trace. לאחר מילוי הנתונים ושליחתם, ניתן להסתכל ב-Trace Log ולראות את פרטי הטופס שנשלח:

Untitled - 1 (NETSER-PC\SQLNETSER)

EventClass	TextData	ApplicationName	NTUserName	LoginName	CPU	Reads	Writes	Duration
RPC:Completed	declare @p1 nvarchar(64) set @p1=N...	Report Server	SYSTEM	NT AUT...	0	2	0	1
SQL:BatchStarting	...	Report Server	SYSTEM	NT AUT...				
SQL:BatchCompleted	...	Report Server	SYSTEM	NT AUT...	0	0	0	1
SQL:BatchStarting	...	Report Server	SYSTEM	NT AUT...				
SQL:BatchCompleted	...	Report Server	SYSTEM	NT AUT...	0	0	0	0
Audit Login	-- network protocol: LPC set quote...	.Net SqlClie...	netser	netser...				
RPC:Completed	exec dbo.aspnet_CheckSchemaVersion ...	.Net SqlClie...	netser	netser...	0	2	0	0
RPC:Completed	exec dbo.aspnet_CheckSchemaVersion ...	.Net SqlClie...	netser	netser...	0	2	0	0
RPC:Completed	declare @p12 uniqueidentifier set ...	.Net SqlClie...	netser	netser...	0	24	0	14
Audit Logout		.Net SqlClie...	netser	netser...	0	28	0	16
RPC:Completed	exec sp_reset_connection	.Net SqlClie...	netser	netser...	0	0	0	0

```
declare @p12 uniqueidentifier
set @p12='7C60C937-9981-4100-B686-38011D51A520'
exec dbo.aspnet_Membership_CreateUser
@ApplicationName=N'MyApplication',@UserName=N'nn3',@Password=N'zxcvbn!',@PasswordSalt=N'rmzH+l7Mz8KWkYKBL5Wsug==',@Email=N'tyrrt@gmail.com',@Passw
18:05:01',@UserId=@p12 output
select @p12
```

Ready. Rows: 6

במצב ברירת המחדל של קובץ ה-Web.config במקרה של שימוש בפונקציות ההתחברות המובנות של ויז'ואל סטודיו מוגדר שתוכן השדות ישלח כמוצפן:

```
passwordFormat="Hashed"
```

על מנת שהמידע שנשלח לא יהיה מוצפן צריך לכתוב זאת במפורש:

```
passwordFormat="Clear"
```

ולכן רק התחברות בצורה פרטית נחשבת פחות בטוחה.

במידה ונקבע כי הסיסמאות ישלחו באופן מוצפן ונשלח את הטופס שנית, נוכל לראות ב-Profiler:

Untitled - 5 (NETSER-PC\SQLNETSER)

EventClass	TextData	ApplicationName	NTUserName	LoginName	CPU	Reads	Writes	Duration
RPC:Completed	declare @p12 uniqueidentifier set ...	.Net SqlClie...	netser	netser...	0	24	1	66
Audit Logout		.Net SqlClie...	netser	netser...	0	439	3	70
RPC:Completed	exec sp_reset_connection	.Net SqlClie...	netser	netser...	0	0	0	0
Audit Login	-- network protocol: LPC set quote...	.Net SqlClie...	netser	netser...				
RPC:Completed	exec dbo.aspnet_Membership_GetPassw...	.Net SqlClie...	netser	netser...	0	21	0	2
Audit Logout		Report Server	SYSTEM	NT AUT...	16	1086	0	10003
RPC:Completed	exec sp_reset_connection	Report Server	SYSTEM	NT AUT...	0	0	0	0
Audit Login	-- network protocol: LPC set quote...	Report Server	SYSTEM	NT AUT...				
RPC:Completed	declare @p1 nvarchar(64) set @p1=N...	Report Server	SYSTEM	NT AUT...	0	2	0	0
RPC:Completed	declare @p1 nvarchar(64) set @p1=N...	Report Server	SYSTEM	NT AUT...	0	2	0	0
SQL:BatchStarting	...	Report Server	SYSTEM	NT AUT...				

```
declare @p12 uniqueidentifier
set @p12='9A3E7B4E-3A2B-408A-8A63-DABF4083E07E'
exec dbo.aspnet_Membership_CreateUser
@ApplicationName=N'MyApplication',@UserName=N'nn',@Password=N'R/HPtJf3JE2x6RPF5ypPlg0xFXU=',@PasswordSalt=N'PiPC29qmLD3CMm3TXzRCkA==',@Email=N'ty@
18:08:12',@UserId=@p12 output
select @p12
```

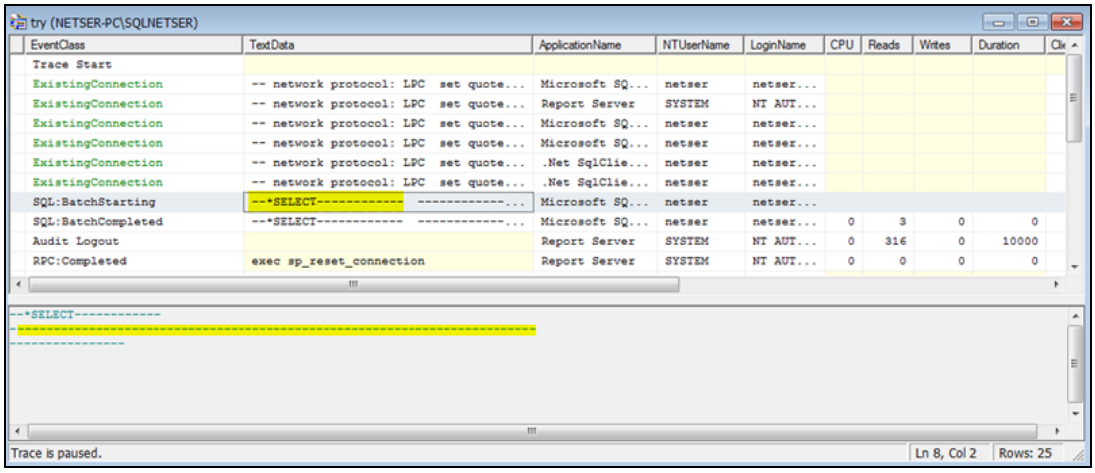
Trace is stopped. Ln 97, Col 1 Rows: 111

במקרה זה ה-Profiler לא עזר לנו.

נחזור לדוגמה שלנו לשימוש ב-EncryptByPassPhrase. נתחבר עם ה-Profiler ונריץ את השאילתה הקודמת:

```
select encryptedData,
convert(varchar,decryptByPassPhrase('1324 5644',encryptedData)) as
decrypted
from MyPassTable
```

ברור שהמידע הסודי שלנו הוא ה-PassPhrase - המחרוזת אשר בעזרתה מחוללים את מפתח ההצפנה. אם נסתכל ב-Profiler, נראה את הדבר הבא:



גם כאן- ה-Profiler אכן מזהה את השאילתא אבל השרת מונע מהמידע לזלוג וממלא בקווים את המידע הרגיש. אז היכן ה-Profiler כן יכול לעזור לנו?

במקרים בהם יש שימוש ב-Stored Procedure! (תזכורת: Stored Procedure אלו מספר פקודות המקובצות בתהליך אחד שתפקידו לבצע דבר מה מוגדר מראש. התהליך נשמר על השרת ובכל פעם שנרצה לבצע את אותן הפעולות- נריץ את ה-Stored Procedure עם הפרמטרים המתאימים.)

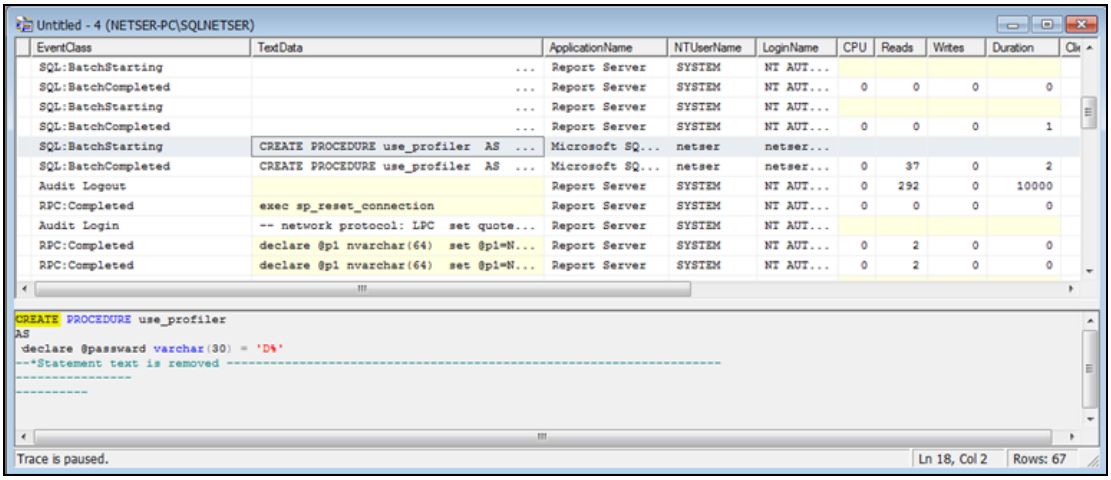
לדוגמא, הכנתי Stored Procedure בשם "use_profiler", שתפקידו לפענח מידע בעזרת ה-decryptByPassPhrase ו-PassPhrase שמגיע מגורם חיצוני:

```
CREATE PROCEDURE use_profiler
AS
declare @passward varchar(30) = 'D%'
select encryptedData,
convert(varchar,decryptByPassPhrase(@passward,encryptedData)) as
decrypted
from MyPassTable
go
```

- `use_profiler` - שם הפרוצדורה.

- `@password` - פרמטר שיגיע מגורם חיצוני וישמש כ-PassPhrase, לדוגמא - הסיסמה של המשתמש (זה המידע הרגיש).

אם נסתכל על ה-Profiler לאחר הרצת הפרוצדורה, נראה:

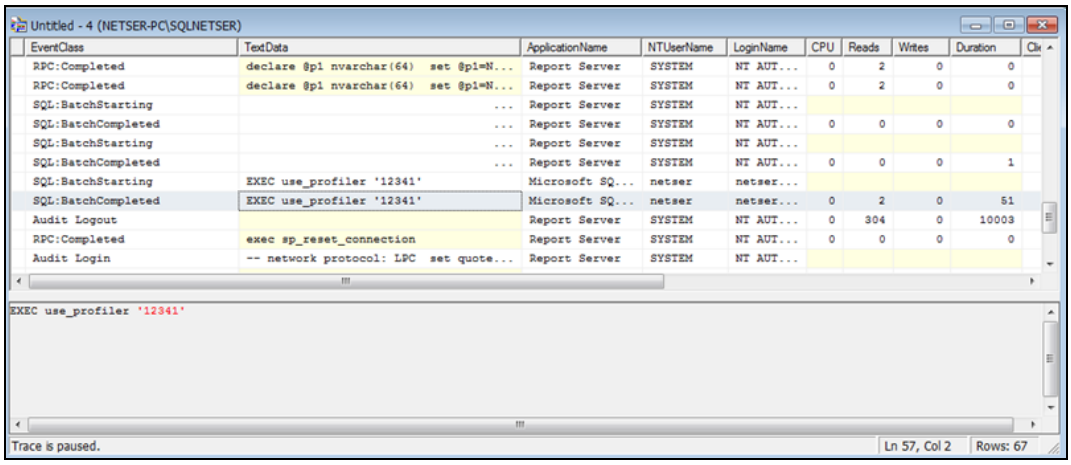


ניתן לראות את תחילת יצירת הפרוצדורה אך כל השאר, הדבר היחידי שנוכל לראות הוא שהפרוצדורה אמורה לקבל את סיסמת המשתמש כפרמטר ולהציב אותו ב-@password.

אם נפעיל את הפרוצדורה, לדוגמא כך:

```
EXEC use_profiler '12341'
```

ונסתכל ב-Profiler:



וזהו, הר הבית בידינו! ☺

- **Brute Force**:

התקפה לא כל כך חכמה, אך די יעילה כאשר ישנו שימוש ב-PassPhrase הצפנה פשוט, נדגים אותה על הטבלה שיצרנו. ניצור פונקציה שתבצע את ההתקפה, לא אסביר אותה שורה שורה, אבל היא פועלת בצורה דומה לכל מימוש אחר של BruteForce למעט מגבלה של 6 אותיות למידע המוצפן.

ניתן כמובן להרחיב את הפונקציה ליותר אותיות. שם הפונקציה היא - encryption_data:

```
CREATE function [dbo].[encryption_data] (@encryptedtext varbinary(max))
returns varchar(8000)
with execute as caller
as
begin
declare @data varchar(8000)
declare @password varchar(6)
declare @i int
declare @j int
declare @k int
declare @l int
declare @m int
declare @n int


set @i=-1
set @j=-1
set @k=-1
set @l=-1
set @m=-1
set @n=-1
set @password =''

while @i<255
begin
    while @j<255
    begin
        while @k<255
        begin
            while @l<255
            begin
                while @m<255
                begin
                    while @n<=255
                    begin
                    set @password=isnull(char(@i),'') +
isnull(char(@j),'')+isnull(char(@k),'')
                              + isnull(char(@l),'')+isnull(char(@m),'')
+ isnull(char(@n),'')
                    if
convert(varchar(100),DecryptByPassPhrase(ltrim(rtrim(@password)),
                              @encryptedtext)) is not null
                    begin
                    --print 'This is the Encrypted text:' +@password
```

```
                    set @i=256;set @j=256;set @k=256;set @l=256;set
@m=256;set @n=256;
                    set @data = convert(varchar(100),

      DecryptByPassPhrase(ltrim(rtrim(@password)),@encryptedtext))
                    end
                    --print 'A'+ltrim(rtrim(@password))+'B'
                    --print convert(varchar(100),
                              --
DecryptByPassPhrase(ltrim(rtrim(@password)),@encryptedtext))
                    set @n=@n+1
                    end
                set @n=0
                set @m=@m+1
                end
            set @m=0
            set @l=@l+1
            end
        set @l=0
        set @k=@k+1
        end
    set @k=0
    set @j=@j+1
    end
set @j=0
set @i=@i+1
end

return @data
END
```

[הקוד במקור: http://www.databasejournal.com/features/mssql/article.php/3717826/SQL-Server-2005---Hacking-password-Encryption.htm]

הפעלה של הפונקציה:

```
select  data, [dbo].[encryption_data]
(encryptedData) data_hacked from MyPassTable
```

וכמובן שככל שהסיסמה תהיה יותר ארוכה ויותר מסובכת יקח יותר זמן לביצוע הפעולה. לעוד דוגמאות אני ממליץ לעבור על הקישור הבא:

http://www.databasejournal.com/features/mssql/article.php/3717826/SQL-Server-2005---Hacking-password-Encryption.htm

הצפנה סימטרית עם חתימה דיגיטלית:

ההצפנה השניה שנראה היא הצפנה סימטרית בעזרת חתימה דיגיטלית. הרחבה על חתימה דיגיטלית ועל צופן סימטריניתן למצוא במאמר של הלל חיימוביץ' ובמאמר של עמיחי פרץ קלפשטוק בנושא, שפורסמנו בגליונות הקודמים של המגזין.

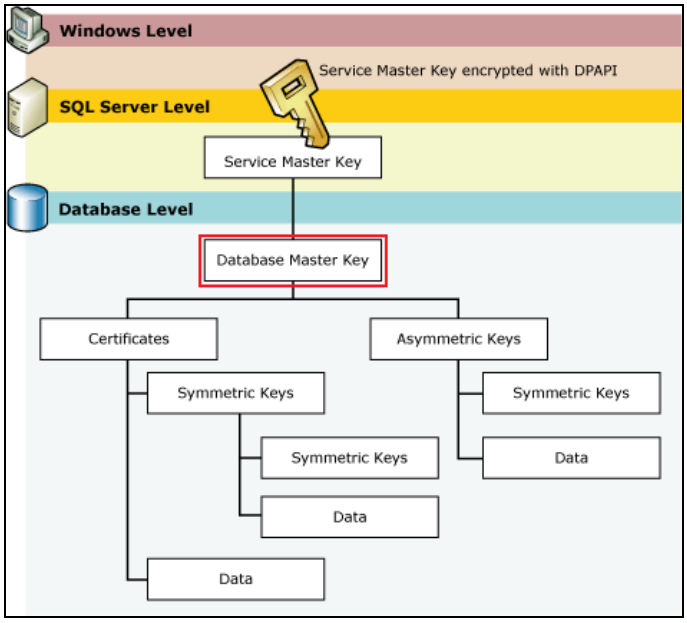
ראשית ניצור טבלה שמכילה מספר כרטיסי אשראי וכן את שם בעל הכרטיס:

```
create table creditCardTable(
      names varchar(22),
      creditCard NVARCHAR(100)
)
```

נזין את הטבלה ידנית. הטבלה נראית כך:

names	creditCard
david saas	1241 1412
משה בטחה	5673462
ori moss	87944488...
עודד ברטום	897446 3...

כעת ניצור את ה-Master key. ה-Masket Key הוא מפתח ייחודי לכל מסד נתונים, ובעזרתו יוצפנו החתימות הדיגיטליות. בתרשים הבא ניתן לראות את ההיררכיה של הגורמים בהצפנה:



[במקור: http://www.4guysfromrolla.com/articles/022107-1.aspx]

בכדי ליצור Master Key, נשתמש ב:

```
create Master Key encryption by password = 'SectertPassword1';
```

במידה והסיסמה מתאימה למדיניות מורכבות הסיסמה, אנו נקבל את ההודעה הבאה:

```
Command(s) completed successfully
```

ואכן, ה-Master Key נוצר בהצלחה.

כעת ניצור בעזרתו את החתימה הדיגיטלית:

```
create certificate MyCert with subject = 'MyCertSubj'
```

- Mycert - השם של החתימה.
- MyCertSubj - הסיסמה של החתימה.

לאחר יצירת החתימה נוכל ליצור את המפתח הסימטרי:

```
create symmetric key MyKey with algorithm=AES_256 encryption
 by certificate MyCert;
```

- MyKey - השם של המפתח הסימטרי.
- algorithm=AES_256 - סוג האלגוריתם שבו משתמשים להצפנה.
- by certificate MyCert - מוצפן בעזרת החתימה הדיגיטלית שיצרנו בשורה הפקודה הקודמת.

ב-SQL Server יש הרבה מידע שמאופסן בתוך טבלאות של המערכת עצמה (sys) וממנה אפשר לדעת הרבה מאוד פרטים טכניים, כמו במקרה שלנו שנרצה לדעת כמה מפתחות נוצרו לנו. לכן נכתוב:

```
select * from sys.symmetric_keys
```

והתוצאה:

	name	principal_id	symmetric_key_id	key_length	key_algorithm	algorithm_desc	create_date
1	##MS_DatabaseMasterKey##	1	101	128	D3	TRIPLE_DES	2011-12-01
2	MyKey	1	256	256	A3	AES_256	2011-12-01
3	creditCardKey	1	257	256	A3	AES_256	2011-12-01
4	SecureSymmetricKey	1	260	192	DX	DESX	2011-12-04
5	SecureSymKey	1	262	256	A3	AES_256	2011-12-04

בין שלל המפתחות שקיימים כבר, ניתן לראות את המפתחות שאנחנו יצרנו MyKey ואת ה-Master Key שיצרנו. כמו כן ניתן לראות נתונים שונים על כל מפתח וביניהם את סוג ההצפנה ותאריך היצירה שלו.

כעת נחזור לטבלה creditCardTable שיצרנו ונוסיף לה עוד עמודה שתשמש למידע המוצפן:

```
alter table creditCardTable
add creditCardEncrypted VARBINARY(MAX)
```

שימו לב שסוג השדה הוא מסוג varbinary והגודל שהוא יכול להכיל הוא המקסימלי. כעת ניתן להתחיל עם ההצפנה עצמה, נפתח את המפתח הסימטרי שיצרנו:

```
open symmetric key MyKey decryption by certificate MyCert
```

לצורך הפתיחה השתמשנו ב-cert שיצרנו בשלבים הקודמים, אחרת לא היה ניתן לפתוח את המפתח. לאחר מכן, נשתמש בפקודה update לצורך עידכון של השורות הקיימות (העמודה creditCardEncrypted היא השדה names לאחר הצפנה):

```
update creditCardTable
SET creditCardEncrypted =
EncryptByKey(Key_GUID('MyKey'),creditCard);
```

"EncryptByKey" מקבלת בפרמטר הראשון את שם המפתח, ובפרמטר השני את השדה אותו מעוניינים להצפין.

והתוצאה:

	names	creditCard	creditCardEncrypted
1	david saas	1241 1412	0x0027D99294A0A24F8FEA00F7E08F32FB01000000B6A039EC9DBC202450BE5F24C8A3E...
2	משה בטחה	5673462	0x0027D99294A0A24F8FEA00F7E08F32FB010000007A4EADFFFED81A383E458FFE223CD...

נוסיף עוד עמודה שבה נציג את השורה המוצפנת:

```
alter table creditCardTable
add creditCardDecrypted VARBINARY(MAX)
```

ונציב בעמודה החדשה את הפיענוח:

```
update creditCardTable
SET creditCardDecrypted =
DecryptByKey(creditCardEncrypted);
```

והתוצאה:

	names	creditCard	creditCardEncrypted	creditCardDecrypted
1	david saas	1241 1412	0x0027D99294A0A24F8FEA00F7E08F32FB010...	0x310032003400310020003100340031003200 2
2	משה בטחה	5673462	0x0027D99294A0A24F8FEA00F7E08F32FB010...	0x350036003700330034003600320 0

בעמודה של creditCardDecrypted מוצגת התוצאה בצורה בינארית לא מוצפנת, נרצה להמיר אותה לתצוגה מקובלת וניתנת לקריאה של varchar, ולכן נבצע:

```
SELECT CONVERT(nVARCHAR(100),creditCardDecrypted) as decryptedRow
 from creditCardTable
```

שימו לב ל-n בתוך ה-nvarchar, לפעמים משתמשים איתה ולפעמים לא, זה רק עניין של המרה, במקרה שלנו אם לא היינו משתמשים היינו מקבלים המרה של ההצגה הבינארית לסינית.

והתוצאה הסופית:

	creditCard	decryptedRow
1	1241 1412	1241 1412
2	5673462	5673462

decryptedRow זה השם שנתתי לתצוגה, וזה לא חלק מהטבלה המקורית. ניתן לראות שזה מתאים למספרי ה-creditCard, כלומר העמודה הראשונה היא לפני הצפנה, והעמודה השניה היא אחרי הצפנה ואחרי פיענוח.

הצפנה סימטרית ללא תעודה דיגיטלית:

קיימת אפשרות פשוטה יותר להצפנה בעזרת מפתח סימטרי, והיא הצפנה סימטרית מבלי להשתמש בחתימה דיגטלית, ואז אין צורך לא ליצור Master Key ולא ליצור חתימה דיגיטלית, אלא רק לשמור על המפתח בעזרת סיסמה. כלומר, לבצע:

```
CREATE SYMMETRIC KEY SecureSymmetricKey
      WITH ALGORITHM = DESX
      ENCRYPTION BY PASSWORD = N'StrongPassword1'
```

וכאשר נרצה לפתוח אותו, נבצע:

```
OPEN SYMMETRIC KEY SecureSymmetricKey
      DECRYPTION BY PASSWORD = N'StrongPassword1';
```

כל שאר הפעולות הן כמו שתיארנו בפעולת ההצפנה הקודמת של המפתח הסימטרי. החסרון בשימוש בהגנה על המפתח רק עם סיסמה הוא כמובן שיהיה יותר קל לפצח את ההצפנה.

הצפנה א-סימטרית:

תהליך ההצפנה א-סימטרית מתבצע באופן כמעט זהה לתהליך ההצפנה הסימטרי. ההבדל הוא כמובן בסוג ההצפנה - ובתוצאה. ניצור את המפתח כמו בדוגמא הבאה:

```
CREATE ASYMMETRIC KEY SecureAsymmetricKey
      WITH ALGORITHM = RSA_2048
      ENCRYPTION BY PASSWORD = N'AnotherStrongPassword1';
```

נשתמש בטבלאות המערכת של SQL Server על מנת לראות את המפתח שיצרנו. הטבלה הזאת מייצגת רק את המפתחות הא-סימטריים ולכן המפתחות שיצרנו בעבר לא מופיעים:

```
SELECT * FROM sys.asymmetric_keys;
```

	name	principal_id	asymmetric_key_id	pvt_key_encryption_type	pvt_key_encryption_type_desc	thumbprint
1	SecureAsymmetricKey	1	256	PW	ENCRYPTED_BY_PASSWORD	0x335328

ניתן לראות כי סוג הסיסמה הוא encrypted by passward, ואכן כך הדבר - בשיטה זו אנו מצפינים בעזרת סיסמה ולא בעזרת תעודה דיגיטלית.

נעבור להצפנה עצמה, נעשה את זה בצורה קצת שונה מהתהליך הקודם, ונשתמש בפרמטרים:

```
DECLARE @str NVARCHAR(100)
      SET @str ='avram zimer';

INSERT INTO AsymmetricTempTable (PlainText, CipherText)
VALUES (
      @str,
      EncryptByAsymKey(AsymKey_ID('SecureAsymmetricKey'), @str)
);
```

- שורה 1 - הגדרנו פרמטר str (@ מופיע לפני השימוש בפרמטר) מסוג nvarchar בגודל 100.
- שורה 2 - הגדרנו את הפרמטר שיהיה שווה למחרוזת "avram zimer".
- שורה 3-5 הכנסנו לתוך הטבלה AsymmetricTempTable את הפרמטר str.
- שורה 6 - הכנסנו לתוך העמודה chiperText את הפרמטר מוצפן בעזרת המפתח שיצרנו בתחילת התהליך.

נחזור על הפעולה הזאת 4 פעמים וכל פעם נשנה את המחרוזת של הפרמטר, כלומר כאן:

```
SET @str ='Type here what you want";
```

נריץ:

```
SELECT * FROM AsymmetricTempTable;
```

והתוצאה:

	Id	PlainText	CipherText
1	1	Hello RSA 2048	0x6E71D839FAA910EB8120418C415DD82F0171CEB07B39EDF98E269FC530034F5AA6F630AC813...
2	2	יצחק אברהם	0x594E2A68E5F944F66F315E810DF4C948BF1FBFA2A5D8253FF61DE2BADFBE0DB3C071FE4C9...
3	3	חיים משה	0x0B9682F76B369466D6C36D6AAE7EEDE6E1FAF6681BF4044902CBFE0029F73042E2C83162F59...
4	4	david saas	0x1B1940E6DBF42B6B06B0A046B17B3E5FA20CBE6AB765F597B2FD3268AD5FDBA86957EC88C...

ובכדי לפענח, נבצע (לשם הבדיקה - נכניס סיסמה לא נכונה):

```
SELECT CipherText,CONVERT(NVARCHAR(100),
    DecryptByAsymKey(AsymKey_ID('SecureAsymmetricKey'),
      CipherText, N'AnotherStrongPassword1')) AS PlainText
FROM AsymmetricTempTable;
```

- CipherText - השדה שהוצפן.
- Convert - המרה מהסוג varbinary שמשמש להצגת ההצפנה לבין varchar.
- DecryptByAsymKey - מקבל שלוש פרמטרים,
  - הראשון - שם המפתח שישמש להצפנה.
  - שני- שם השדה אותו אנחנו רוצים לפענח.
  - שלישי - הסיסמה של המפתח הסימטרי. שימו לב שהשתמשתי בסיסמה שגויה (חסר 1).
- AS PlainText- השם שנתנו לעמודה שתוצג.

התוצאה:

```
Msg 15466, Level 16, State 1, Line 1
An error occurred during decryption.
```

וכמובן עכשיו נכניס את הסיסמה הנכונה:

```
SELECT CipherText,CONVERT(NVARCHAR(100),
    DecryptByAsymKey(AsymKey_ID('SecureAsymmetricKey'),
      CipherText, N'AnotherStrongPassword1')) AS ourPlainText
FROM AsymmetricTempTable;
```

והתוצאה:

	CipherText	ourPlainText
1	0x6E71D839FAA910EB8120418C415DD82F0171CEB07B...	Hello RSA 2048
2	0x594E2A68E5F944F66F315E810DF4C948BF1FBFA2A5...	יצחק אברהם
3	0x0B9682F76B369466D6C36D6AAE7EEDE6E1FAF6681B...	חיים משה
4	0x1B1940E6DBF42B6B06B0A046B17B3E5FA20CBE6AB7...	david saas

קיימת האפשרות להשתמש בהצפנה בעזרת תעודה דיגיטלית, וליצור תאריך "תפוגה":

```
CREATE CERTIFICATE SelfSignedCertificate
   ENCRYPTION BY PASSWORD = 'CertificateStrongPassword1'
   WITH SUBJECT = 'Self Signed Certificate',
EXPIRY_DATE = '12/01/2030'
```

כאשר נרצה לפענח את ההצפנה, נפענח קודם לכן את התעודה בעזרת הסיסמה שקבענו ללא כל צורך בשימוש במפתחות:

```
DecryptByCert(Cert_Id('SelfSignedCertificate'),
    CipherText, N'CertificateStrongPassword1'))
```

כמו כן קיימות עוד אפשרות של הצפנה של המפתחות הקיימים בעזרת מפתחות אחרים וכן מנגנון לבדיקת אימות החתימה דיגיטלית בעזרת שימוש בפונקציות הבאות:

Function	Description
SignByAsymKey	Signs plaintext with an asymmetric key.
VerifySignedByAsmKey	Tests whether digitally signed data has been changed since it was signed.
SignByCert	Signs text with a certificate and returns the signature.
VerifySignedByCert	Tests whether digitally signed data has been changed since it was signed.

למעוניינים להרחיב מומלץ להיכנס:

http://www.4guysfromrolla.com/articles/022807-1.aspx

הצפנת כלל מסד הנתונים

אחרי שסקרנו השיטות להצפנת נתונים חלקיים בלבד, נעברו להצפנה של כלל מסד הנתונים, המנגנון נקרא TDE, קיצור של: "Transparent Data Encryption" והוא קיים רק מגירסה 2008 Enterprise.

כדי להדגיש את ההבדל בין המנגנון הנ"ל להצפנות שסקרנו עד כה, נציין כי כל ההצפנות עד כההצפינו נתונים רק ברמת השורה או ברמת הטבלה.

היתרונות של ה-TDE הם:

- הצפנה של כלל מסד הנתונים על כל הטבלאתיו ושורותיו. אין צורך לשנות את הארכיטקטורה של מסדי הנתונים, לא צריך להוסיף עמודות או שורות בשביל שיכילו את המידע המוצפן. פשוט מצפינים את הכל ואחר כך מפענחים את הכל.
- מינימום פעולות של פיענוח/הצפנה.
- הצפנה מובנית של כל ה-log-ים וכל הפעולות הנלוות לטבלאות.
- מאחר שהמופע הפיזי של המידע מוצפן, ולא המידע עצמו, אין השפעה על האינדקסים ועל המפתחות (הראשי או הזר בתוך הטבלאות ולא קשור למפתחות ההצפנה) אין בעיה להרצת שאילתות שמסתמכות על האינדקסים/מפתחות.
- הבדלים בין הביצועים (לפי מיקרוסופט...): ההבדלים בין הביצועים הם 3-5% עכשיו לעומת 20-28% במנגנוני ההצפנה הישנים.

והחסרונות הם:

- כאמור, זמין רק מהגירסא ה-Enterprise 2008.
- פחות שליטה על מה יוצפן ומה לא.
- כמעט ולא ניתן יהיה לכווץ או לדחוס את הקבצים ובמקרים שכן נצליח- ההשפעה תהיה שולית.

נתחיל עם ההצפנה עצמה:

מבחר הפעולות שנעשה ישתנו כל פעם בין רמת Master, שהוא מצב הניהול של כל ה-DB לבין רמת ה-DB הפרטני (במקרה שלנו TDE) שאותו ניצור. כדי לעבור בין התחומים לוחצים על ה-Drop Down ואפשר לבחור את מיקום הפעולה:

נבדוק כי לא קיים כבר מסד נתונים בשם TDE ואם כן אז נמחוק אותו:

```
if exists (select * from master.sys.databases where name = 'TDE')
drop database TDE;
```

שימו לב שמדובר במחיקה של כלל ה-DB כלומר כל הטבלאות שקיימות בתוכו גם ימחקו. כעת ניצור את ה-DB מחדש:

```
CREATE DATABASE TDE
```

אחרי שיצרנו בהצלחה את ה-DB "TDE", חשוב לעבור ב-Drop Down ל-TDE:

כעת ניצור את הטבלה Customers כאשר המידע הרגיש יהיה בעצם רק `CreditCardNumber`:

```
Create Table Customers
(
  CustomerId int,
  FirstName nvarchar(50),
  LastName nvarchar(50),
  CreditCardNumber varchar(50)
)
```

לאחר שיצרנו את הטבלה בהצלחה, נמלא אותה בעזרת השאילתא בשלוש שורות חדשות:

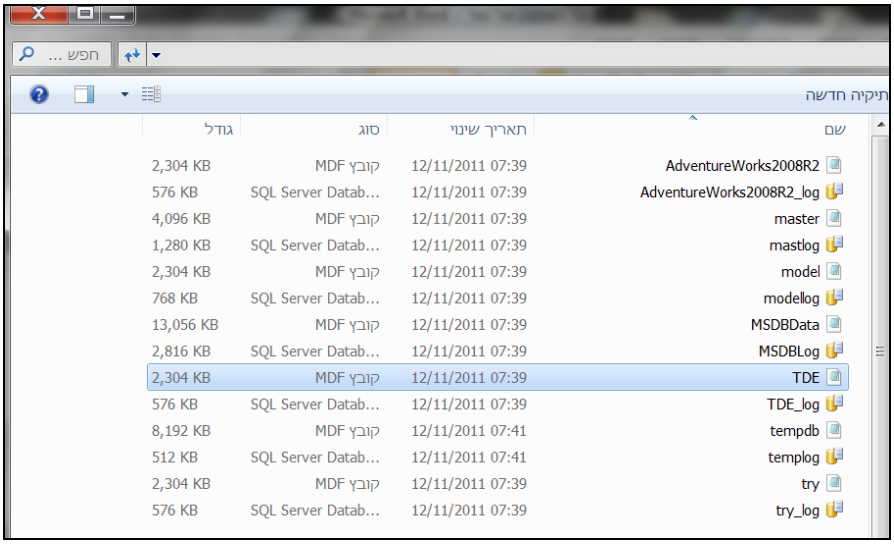
```
Insert Into Customers
Select 1, N'Anakin', N'Skywalker', '1234567812345678'
go
Insert Into Customers
Select 2, N'David', N'Gabay', '9348'
go
```

```
Insert Into Customers
Select 2, N'יצחק', N'פיליפס', '231-415-6617-71631'
```

נביט על הטבלה שנוצרה, ונראה שאכן היא מלאה במידע שרצינו:

	CustomerId	FirstName	LastName	CreditCardNumber
1	1	Anakin	Skywalker	1234567812345678
2	2	David	Gabay	9348
3	2	יצחק	פיליפס	231-415-6617-71631

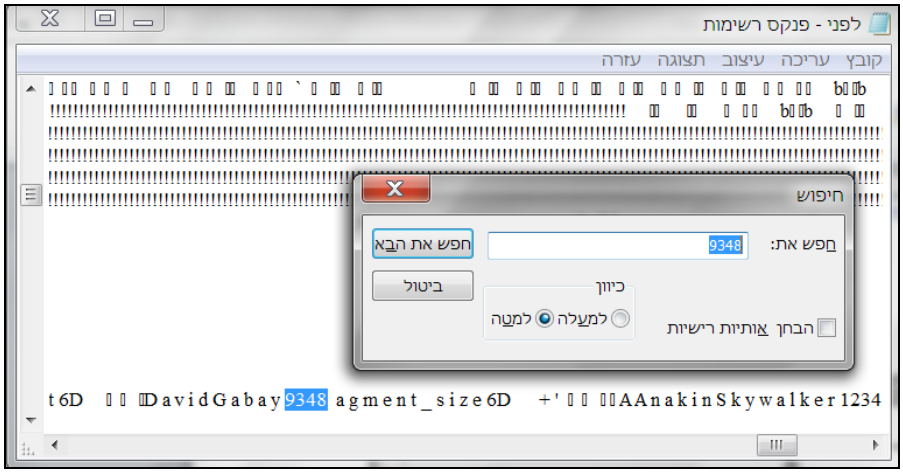
לפני שנמשיך חשוב שנבין דבר נוסף: ה-SQL Server שומר את כל המידע שבמסד הנתונים בצורה קשיחה על המחשב שבו הוא מותקן במבנה קבצים יחודיים לו. אפשר בקלות יחסית לפתוח את הקבצים האלה בעזרת notepad או כל עורך טקסט אחר, המידע אומנם לא יהיה מסודר, רק עדיין יהיה ניתן לקריאה. בכדי להמחיש זאת, נפתח את הספריה שבה כל הקבצים של ה-SQL Server נשמרים, הוא כמובן ישתנה בהתאם לאיפה שהתקנתם את SQL Server:



מבין כלל הקבצים בתמונה, ניתן לראות את הקובץ ששומר את ה-DB שלנו- TDE. על מנת שלא נפגע במידע הקיים, העתקתי את הקובץ למקום אחר וגם שיניתי את שמו מ-"TDE" ל-"לפני". פתחתי אותו ולחצתי על ctrl+f לצורך חיפוש בתוך הקובץ. אני מחפש מידע מתוך הטבלה הלא מוצפנת:

	CustomerId	FirstName	LastName	CreditCardNumber
1	1	Anakin	Skywalker	1234567812345678
2	2	David	Gabay	9348
3	2	יצחק	פיליפס	231-415-6617-71631

נחפש את 9348:



בקלות ניתן לראות כי בעזרת גישה פיזית לקובץ אנו יכולים לראות את הנתונים הקיימים בו – הם אינם נשמרים באופן מוצפן.

כעת, נתחיל בתהליך ההצפנה. נעבוד ברמת ה-Master: נמחק את הMaster-Key הקיים:

```
drop master key
```

ניצור Master Key מוגן בעזרת סיסמה:

```
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'masterkeypassword';
```

כעת ניצור את התעודה הדיגיטלית:

```
CREATE CERTIFICATE TDECertificate WITH SUBJECT = 'TDECertificate';
```

ניצור את המפתח ברמת ה-DB, מוגן בעזרת AES באורך 128 ביט ובעזרת התעודה הדיגיטלית שיצרנו בפעולה הקודמת:

```
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_128
ENCRYPTION BY SERVER CERTIFICATE TDECertificate;
```

לאחר מכן, עלינו להגדיר שימוש בהצפנה:

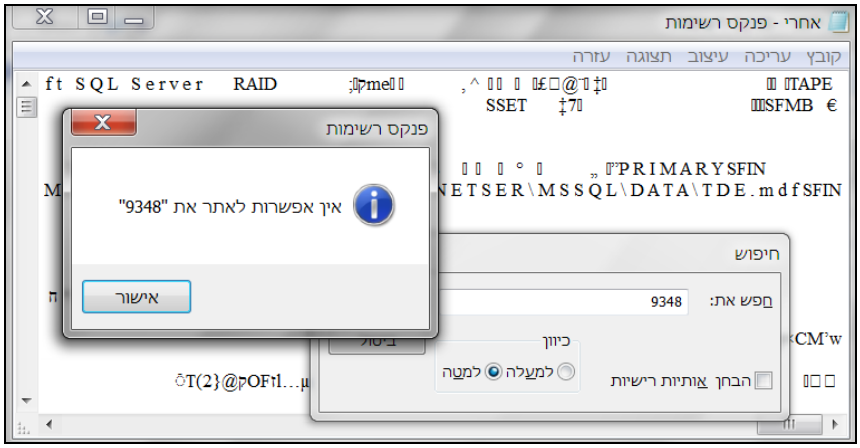
```
ALTER DATABASE TDE SET ENCRYPTION ON;
```

מעכשיו - כל פעולה שתיעשה ותשנה את מסד הנתונים TDE תהיה רשומה מוצפנת. נוכל לבדוק שתהליך ההצפנה אכן התחיל לעבוד:

```
SELECT is_encrypted FROM master.sys.databases WHERE name = DB_NAME()
```

אם קיבלנו "1" - תהליך ההצפנה עובד. "0" - לא עובד.

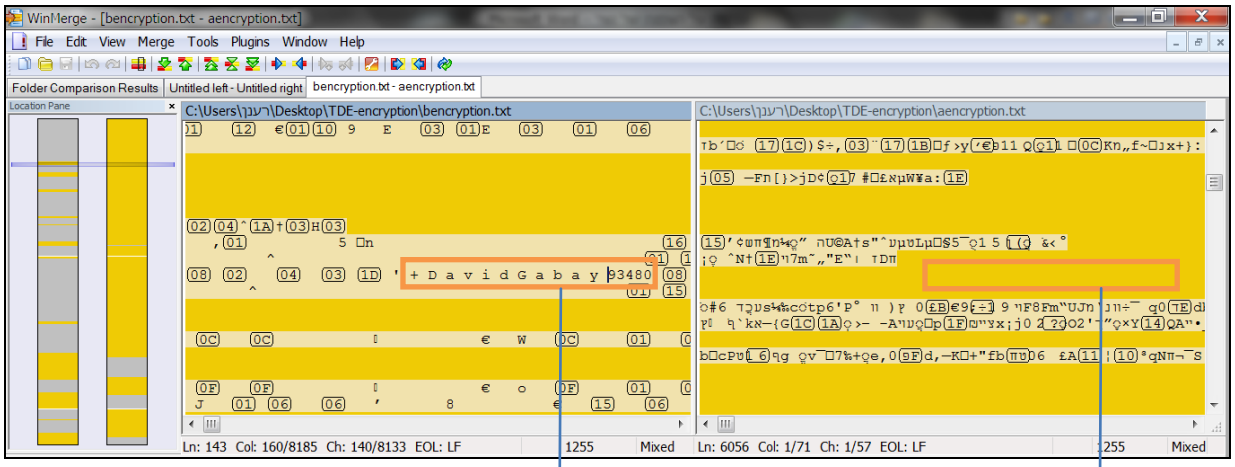
כעת נעשה את אותה פעולה שעשינו לפני ההצפנה וניגש לעותק הקשיח של ה-DB. נבצע את אותו החיפוש שביצענו אחר המחרוזת 9348, והתוצאה:



לצורך ההדגמה נשתמש בתוכנה מאוד טובה להשוואה בין קטעי קוד בשם WinMerge. יש הרבה מאוד דברים שאפשר לעשות איתה, אבל אנחנו נסתפק בהצגת קובץ הDB לפני השינויים ואחרי השינויים. ניתן להוריד אותה מכאן:

http://sourceforge.net/projects/winmerge/files/stable/2.12.4/WinMerge-2.12.4-Setup.exe/download

נעתיק את הקובץ לפני ההצפנה ולאחר ההצפנה, את שניהם נפתח בתוכנה WinMerge ונראה את ההבדלים. הצד השמאלי הוא הקובץ הלא מוצפן והצד הימני מוצפן:



מוצפן - ולא מצאנו כלום. לא מוצפן - ומצאנו את המחרוזת.

בצד שמאל ניתן לראות את ההבדלים בין שני הקבצים. כתום אומר שיש מידע ואפור אומר שריק שם. ניתן לראות שבצד הימני, כל המסמך מלא במידע לעומת השמאלי שכמעט רובו ריק לחלוטין.

נחזור ל-SQL Server, נוכל לבצע כיבוי המנגנון של ההצפנה, או פיענוח של המידע הקיים ע"י:

```
ALTER DATABASE TDE SET ENCRYPTION off;
```

חשוב לציין שאם המשתמש מסתכל על הטבלאות בתוך המשתמש שלו ואם המפתחות פתוחים, הוא יראה את כל המידע בצורה גלויה. ולא יצטרך להתעסק עם ההצפנה כמעט לאורך כל העבודה על ה-DB הייעודי להצפנה.

סיכום

במאמר זה סקרנו את סוגי ההצפנה הקיימים ב-MSSQL, בתחילה נגענו בדרכים להצפין שדות בודדים ולאחר מכן ראינו כיצד ניתן להצפין את כלל מסד הנתונים. במאמר זה הוצגו אך ורק דרכי הצפנה המובנים במסד הנתונים MSSQL, קיימים מקרים בהם ישנו הצורך להגן על מסד הנתונים בעזרת כלים חיצוניים- כלים כגון EFS אשר מאפשרים הצפנת קבצים בודדים בעזרת כלים מובנים במערכת הקבצים ומערכת ההפעלה. כלים כגון BitLocker או TrueCrypt המאפשרים הצפנת מחיצות שלמות במערכת הקבצים. חשוב להכיר את הכלים הללו ולהתאים את השימוש בהם לפי הסיטואציה בהם אנו נתקלים.

תודות

תודה רבה למר מיכאל שובמן על ההדרכה וההכוונה.

מקורות

- http://www.databasejournal.com/features/mssql/article.php/3717826/SQL-Server-2005--Hacking-password-Encryption.htm
- www.msdn.com
- http://technet.microsoft.com
- http://www.4guysfromrolla.com/articles/022807-1.aspx
- http://programming4.us/security/1074.aspx
- http://www.simple-talk.com/sql/database-administration/transparent-data-encryption/
- http://www.mssqltips.com/sqlservertip/1312/managing-sql-server-2005-master-keys-for-encryption/