Transparent Data Masking with DbDefence

What is Data Masking?

Data masking is a special way of encrypting and displaying sensitive data. Unauthorized users or applications see data in masked form. For example '****-****-****-9363' or '0000 0000 0000 0000' for credit cards. Authorized users or applications see the data transparently in its original form.

Introduction

In this document we will demonstrate how to use DbDefence and selectively protect sensitive data without changing anything in the application. It requires no source code changes or special knowledge. The method described here can be applied to all kinds of applications: Desktop, Windows services and Web Applications.

Software developing companies may purchase a redistribution license and install DbDefence as a part of their own software.

Requirements

Supported SQL Server versions on Windows OS:

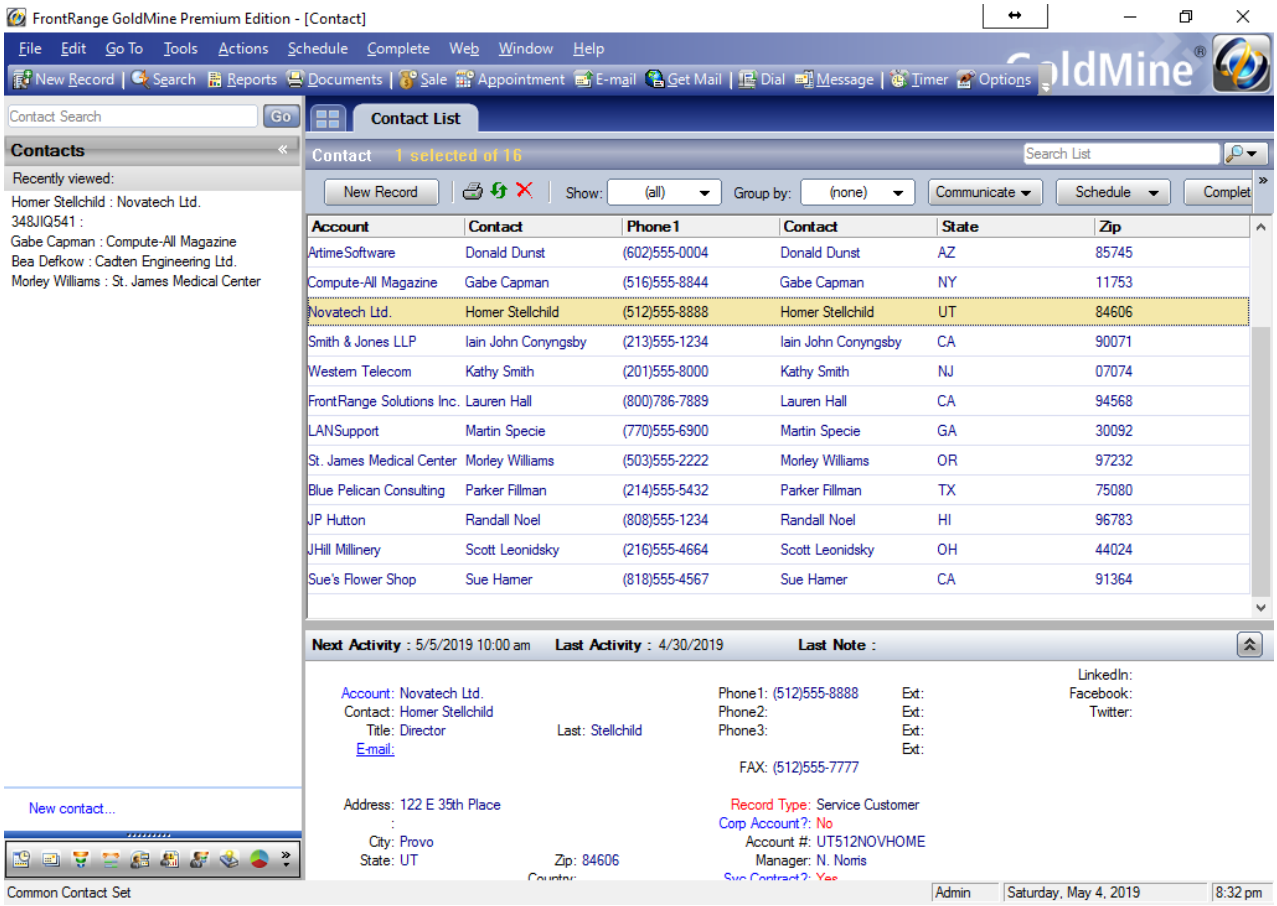
- SQL Server 2008 R2 X64 (All Editions)
- SQL Server 2012 X64 (All Editions)
- SQL Server 2014 X64 (All Editions)
- SQL Server 2016 X64 (All Editions)
- SQL Server 2017 X64 (All Editions)

32-bit versions are not yet supported, but this is planned for the future.
Please contact us if you're interested.

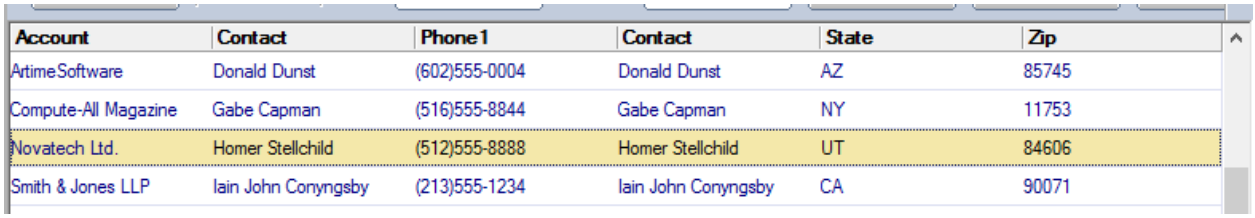
Data Masking functionality isn't available in the Free Version. Please request the time-limited evaluation version from info@activecrypt.com.

Demo Application

As a demo application we will take GoldMine CRM. It is a popular CRM and stores its data in SQL Server. Below you will see GoldMine's main window with some demo data.

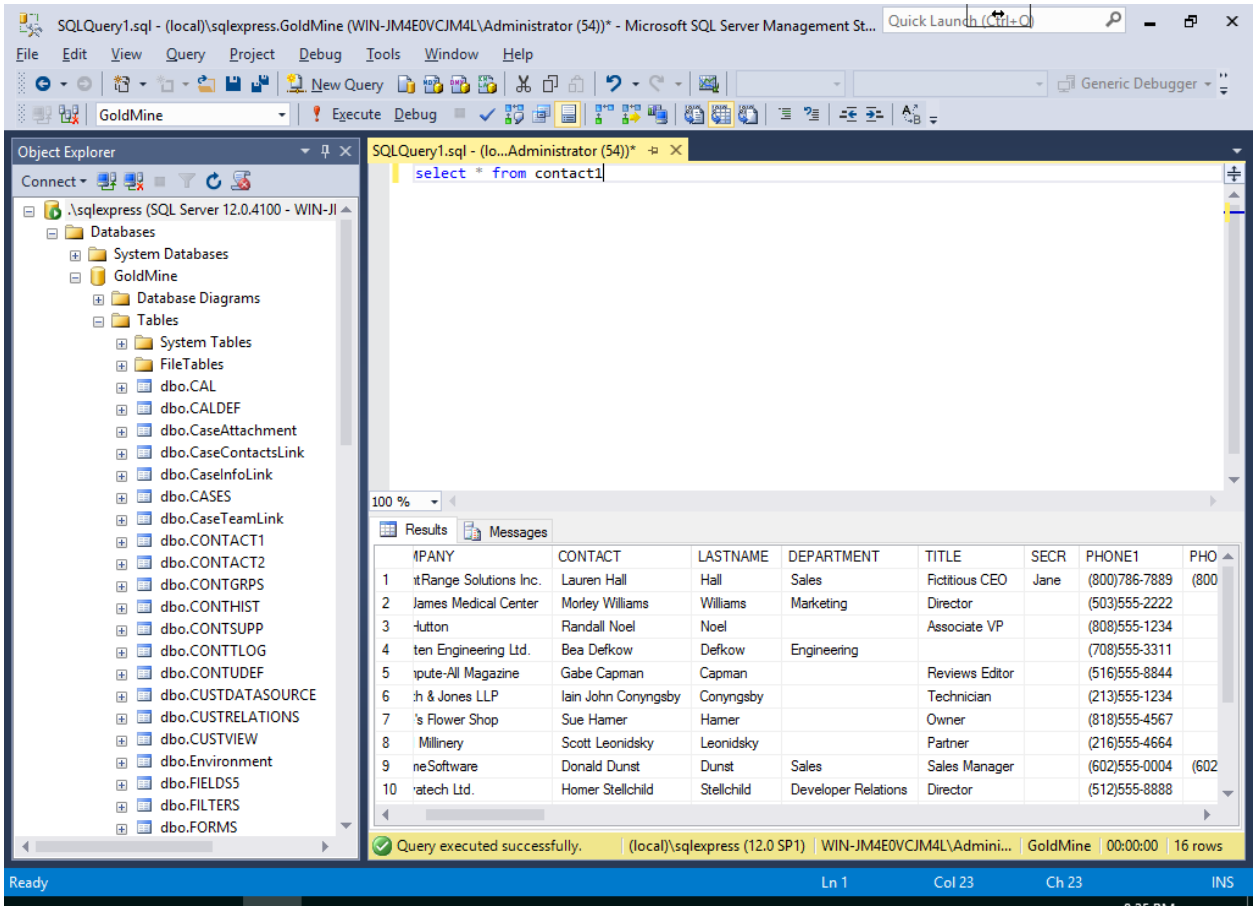


Pic. 1. Main Window



Pic. 2. Record details

Let’s see where GoldMine stores the data. Navigate to the GoldMine database with SSMS and select from the table *CONTACT1*.



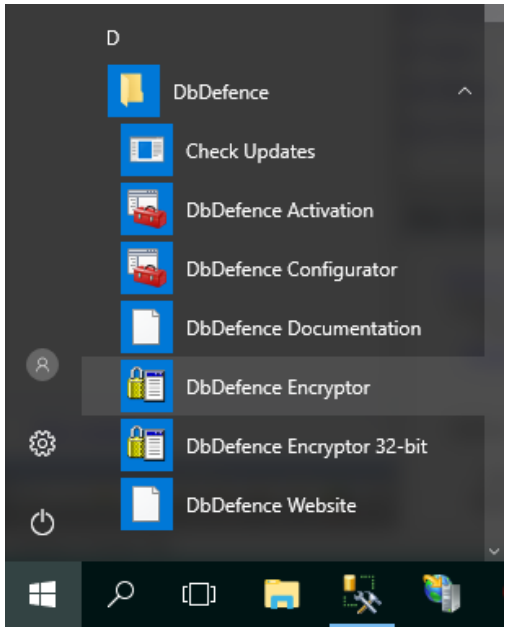
Pic. 3. Source data

As you can see, the data is stored in clear text form. Let’s assume we need to hide the fields *CONTACT*, *LASTNAME* and *PHONE1*. For the *PHONE1* field, we plan to show only the area code and mask the rest. Let’s also assume we need GoldMine CRM to see the masked data transparently, without any encryption or masking.

Encryption

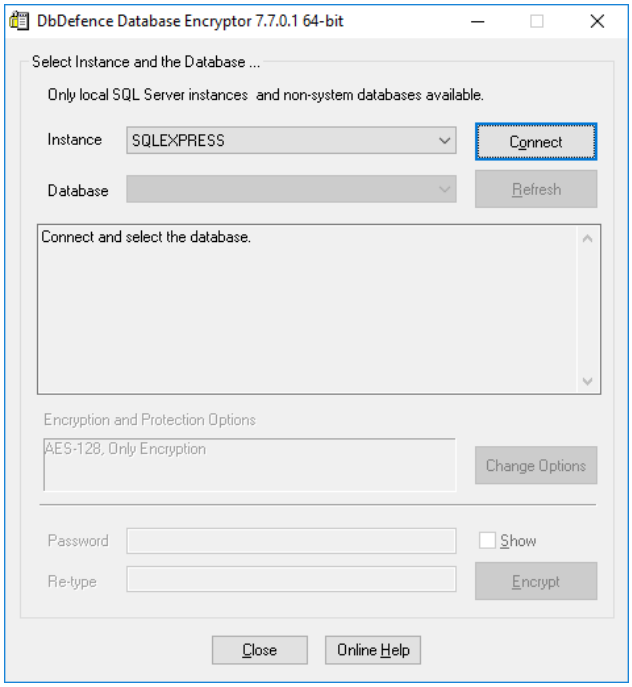
Get DbDefence from https://www.database-encryption.com, and request an evaluation license from info@activecrypt.com.

At this step we assume you already have DbDefence installed.
To start database encryption run DbDefence Encryptor.



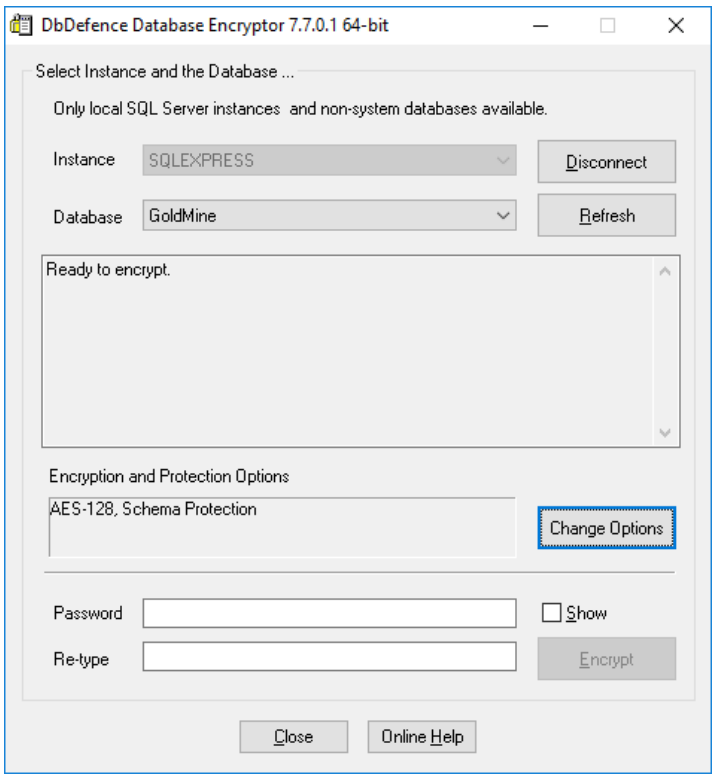
Pic. 4. Starting Encryptor tool

Encryptor starts and lists local instances.



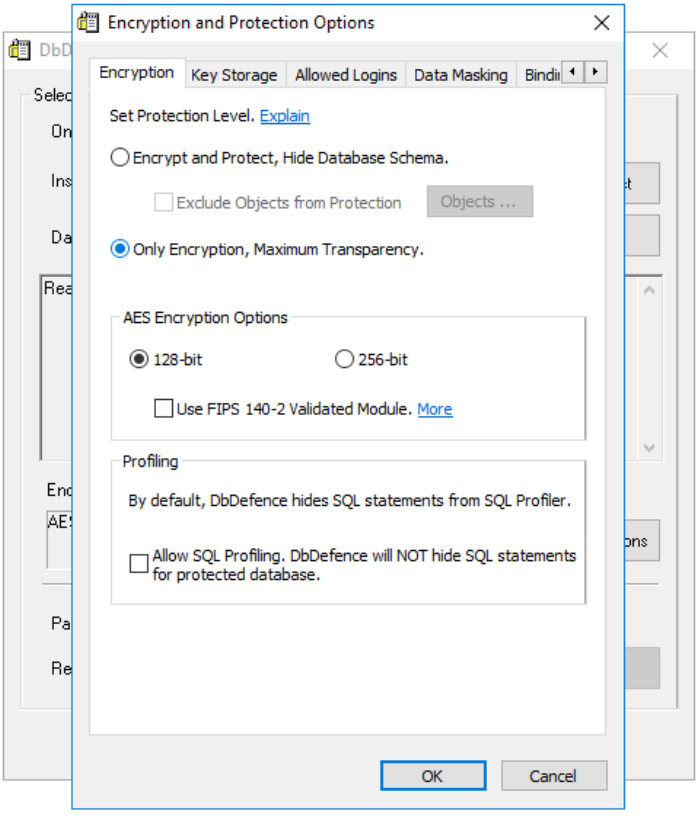
Pic. 5 Encryptor's main window

We connect to the SQLEXPRESS instance and select GoldMine database. At this step we need to configure Data Masking. Click "Change Options" to start configuration.



Pic. 6 Encryptor connected

On the first tab of this dialog, select “Only Encryption, Maximum Transparency”.

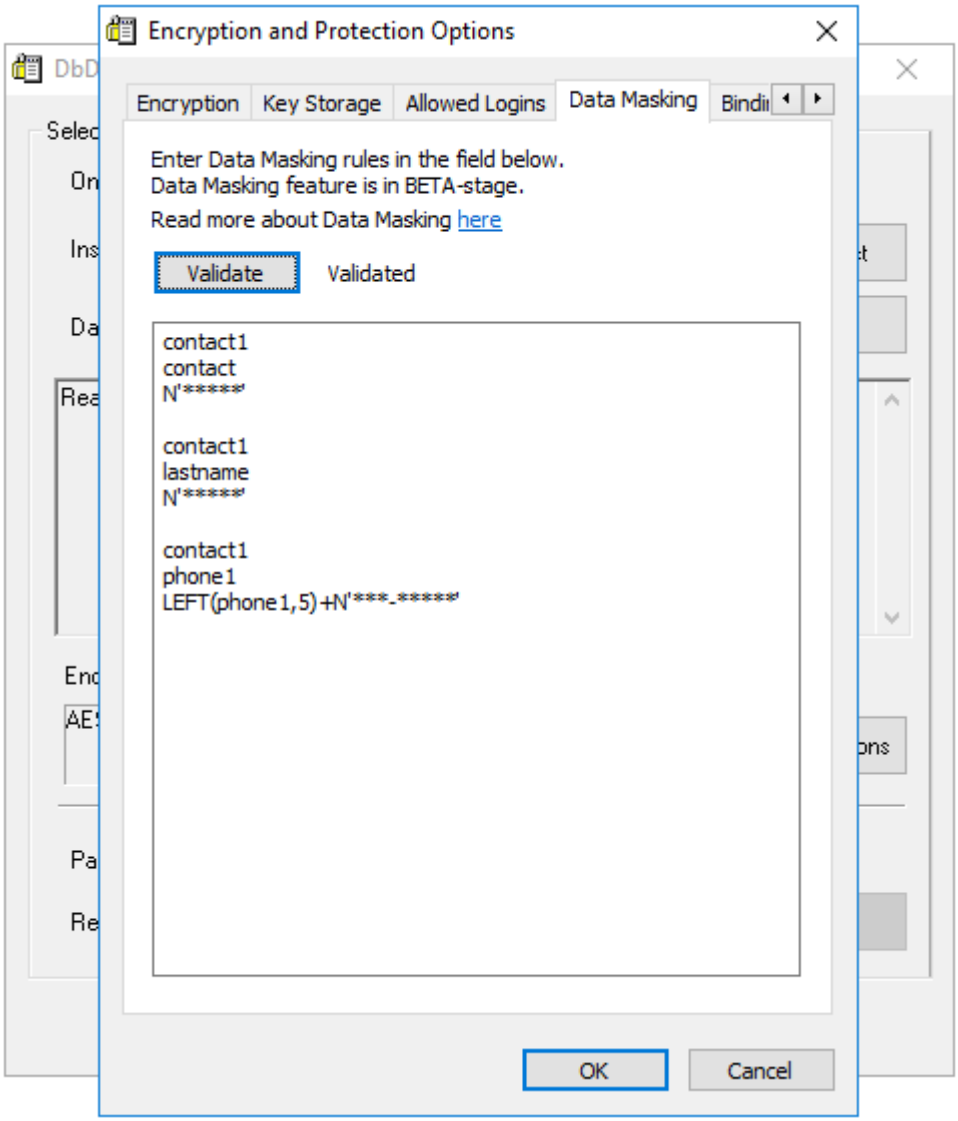


Pic. 7 Setting Only Encryption, Maximum Transparency.

Click on the Data Masking tab to configure the masking rules. The screenshot below shows 3 rules. Each rule consists of 3 lines:

<table name>
<field name>
<mask>

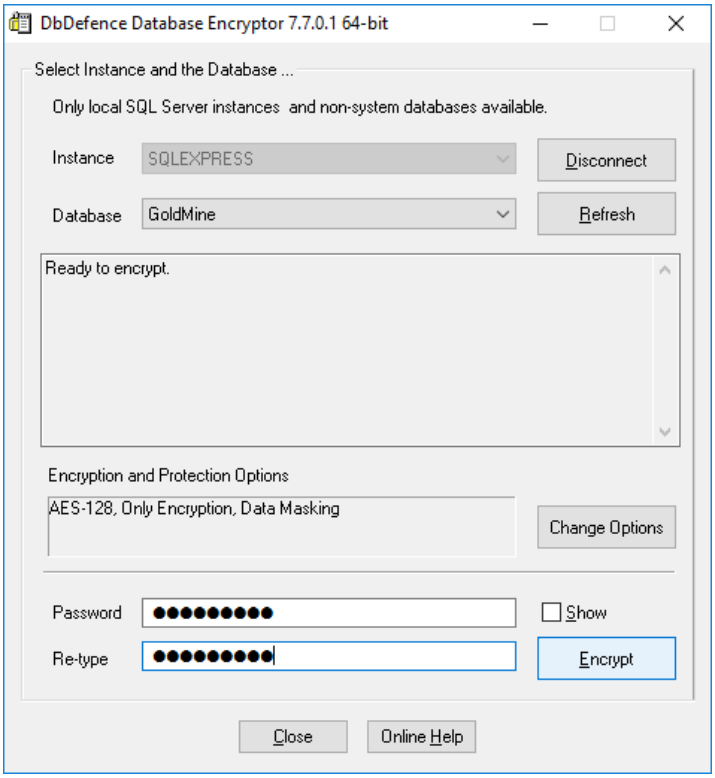
As you can see, the last rule contains the SQL function LEFT. This is used to cut the area code from the phone number. All rules use * as a masking character. You may use any character. The number of characters need not to be equal to the length of the original data. It can be any string, for example, N'not for you' or N'' for an empty string.



Pic. 8 Masking rules

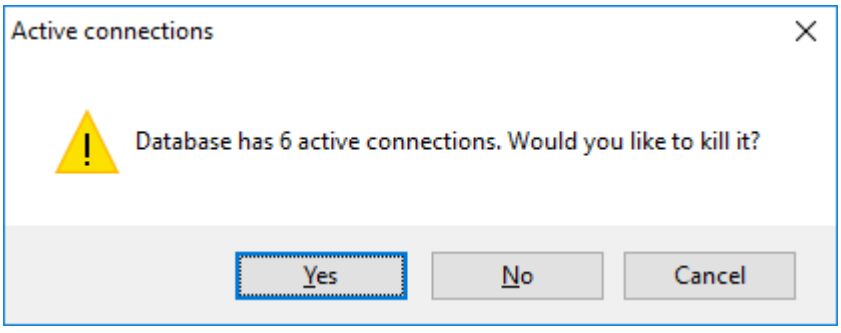
Click "Validate" and Encryptor will check if the rules are correctly defined.
Click "OK" when you're done.

For the final step of the configuration, enter the password. The complexity of the password depends on the SQL Server password policy. Usually Windows Workstation has less strict requirements, while Windows Server requires stronger passwords.

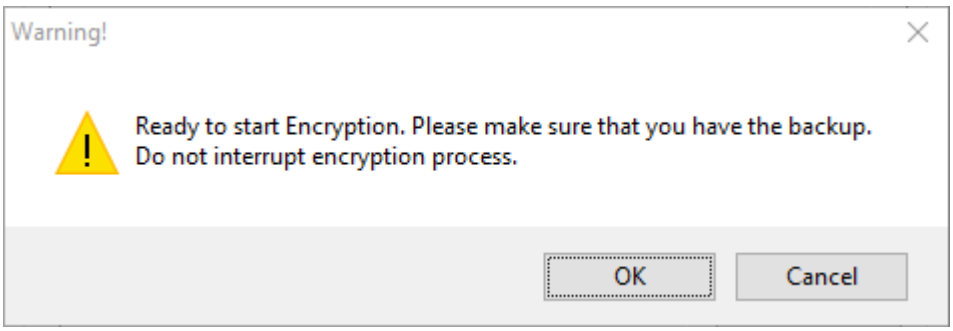


Pic. 9 Ready to start encryption

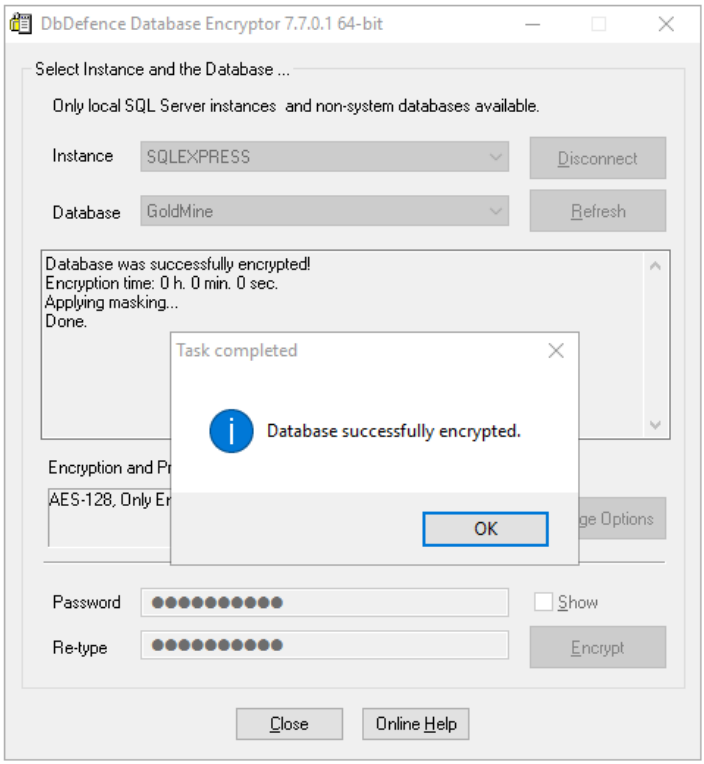
Before starting the encryption process, Encryptor disconnects users from the database (if there are any)



and warns you about the backup.

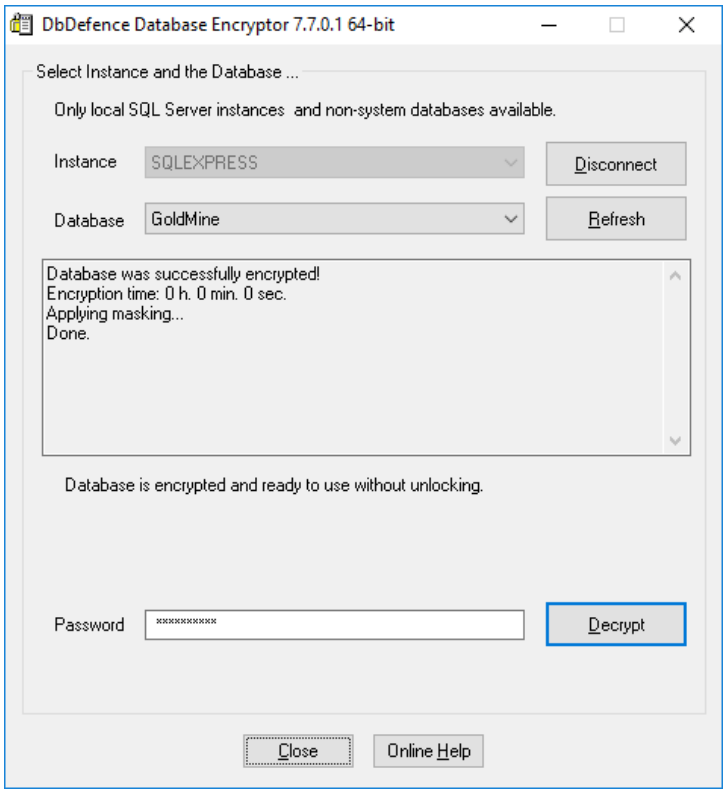


It takes just seconds to encrypt small databases.



Pic. 10 Encryption completed

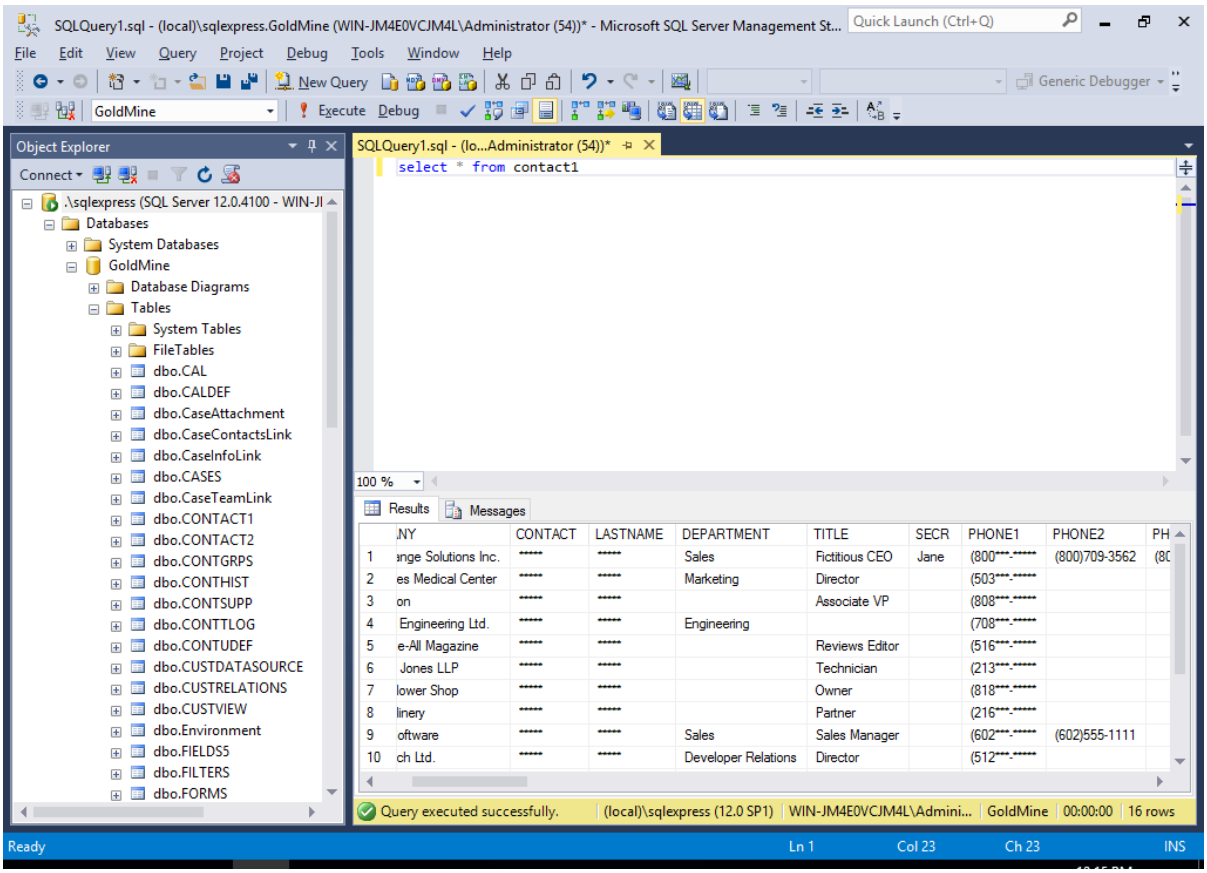
After encryption, Encryptor offers decryption. If something went wrong or you want to make changes you may decrypt the database and start again.



Pic. 11 Ready for decryption

Verification

Let’s see what masked fields look like after encryption.



Pic. 12 Masked data in SSMS

The selected data shows masks instead rather than actual data. The *PHONE1* field contains only area codes and the rest is masked. Everything as expected!

Yet, as well as SSMS, GoldMine has no access to real data and displays masks too. No matter whether you are logged as DBA or not.

American Bank	*****	(310)***-*****	Santa Monica	CA	90403
Cadten Engineering Ltd.	*****	(708)***-*****	Arlington Heights	IL	600005
ArtimeSoftware	*****	(602)***-*****	Tucson	AZ	85745
Compute-All Magazine	*****	(516)***-*****	Jericho	NY	11753
Novatech Ltd.	*****	(512)***-*****	Provo	UT	84606
Smith & Jones LLP	*****	(213)***-*****	Los Angeles	CA	90071

Pic. 13 Masked data in GoldMine

To read the data in decrypted form you need to unlock access. There are 3 ways to do this…

Setting up access for GoldMine

There are **two automatic ways** that GoldMine can use to unlock the database to see the original data:

- Unlock access by SQL login.

  This method is suitable if you want to grant access to all applications that use certain SQL logins. This is the best way if database performance is important.

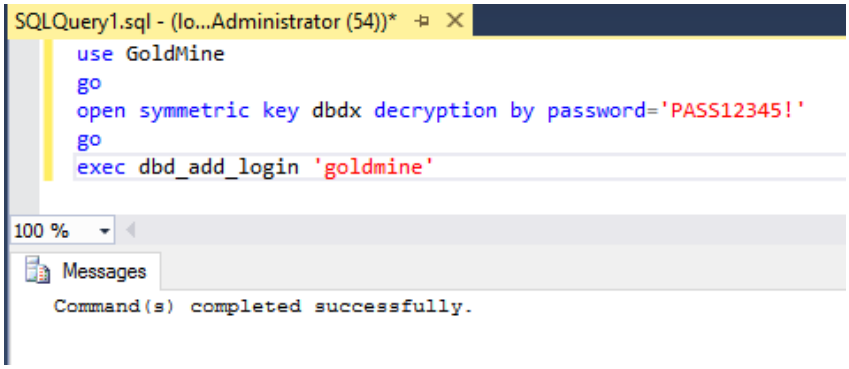
- Unlock access by application.

  This method allows you to permit access to the selected application only.

The third method is not automatic and is suitable for DBAs, programmers, or anyone who knows SQL.

Method 1. Unlock access by SQL login

If the database is already encrypted, you may grant access to SQL login without re-encryption. In our example, GoldMine CRM uses the login *goldmine*. This method works for all kinds of applications.
The SQL statement “OPEN SYMMETRIC KEY” with encryption password (used to encrypt the database) , unlocks the database, and lets you configure the access.



Pic. 14 Adding an SQL login

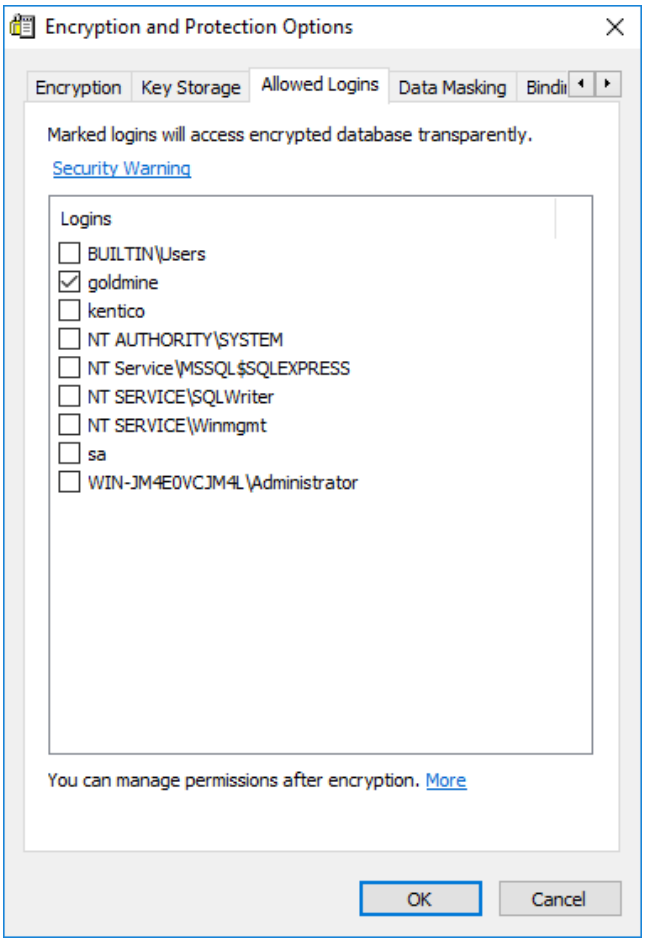
Now, every application logged as *goldmine* will read decrypted data automatically. Several logins can be added with the *dbd_add_login* function.
You need to re-start the application and login again. We can see that GoldMine shows the data correctly.

Account	Contact	Phone1	Contact	State	Zip
Artime Software	Donald Dunst	(602)555-0004	Donald Dunst	AZ	85745
Compute-All Magazine	Gabe Capman	(516)555-8844	Gabe Capman	NY	11753
Novatech Ltd.	Homer Stellchild	(512)555-8888	Homer Stellchild	UT	84606
Smith & Jones LLP	Iain John Conyngsby	(213)555-1234	Iain John Conyngsby	CA	90071

Pic. 15 Unmasked data in GoldMine

NOTE: If you're testing this method with SSMS, close and start SSMS again.

If the database is not yet encrypted, you can define allowed logins simply with mouse clicks. In the "Change Options" dialog, click on "Allowed Logins", then mark the logins.



Pic. 16 The login *goldmine* should access unmasked data

Method 2. Unlock access by application

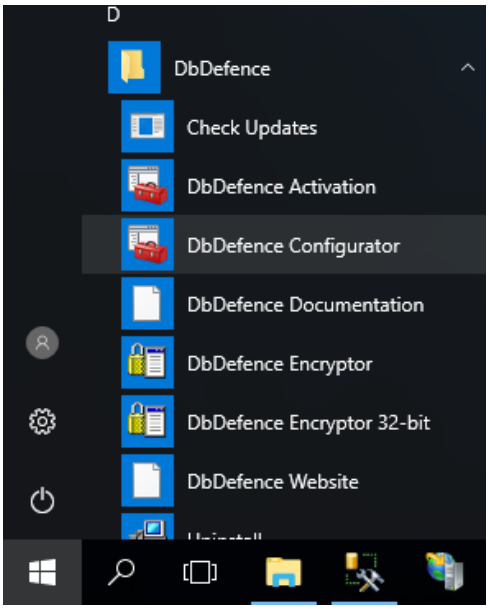
If you've just tried the previous method, decrypt the database and encrypt again, and then close all tested applications. This will help to avoid confusion.

Unlocking access by application allows you to fine-tune access and grant transparent access only to selected applications on selected computers. For this method the Client part of DbDefence must be installed on each client's computer. It is installed on the server by default.
Unlike the first method, the configuration for this method must be done on each client's computer.

If you want to test access to the database from a remote computer, install the Client part of DbDefence there.

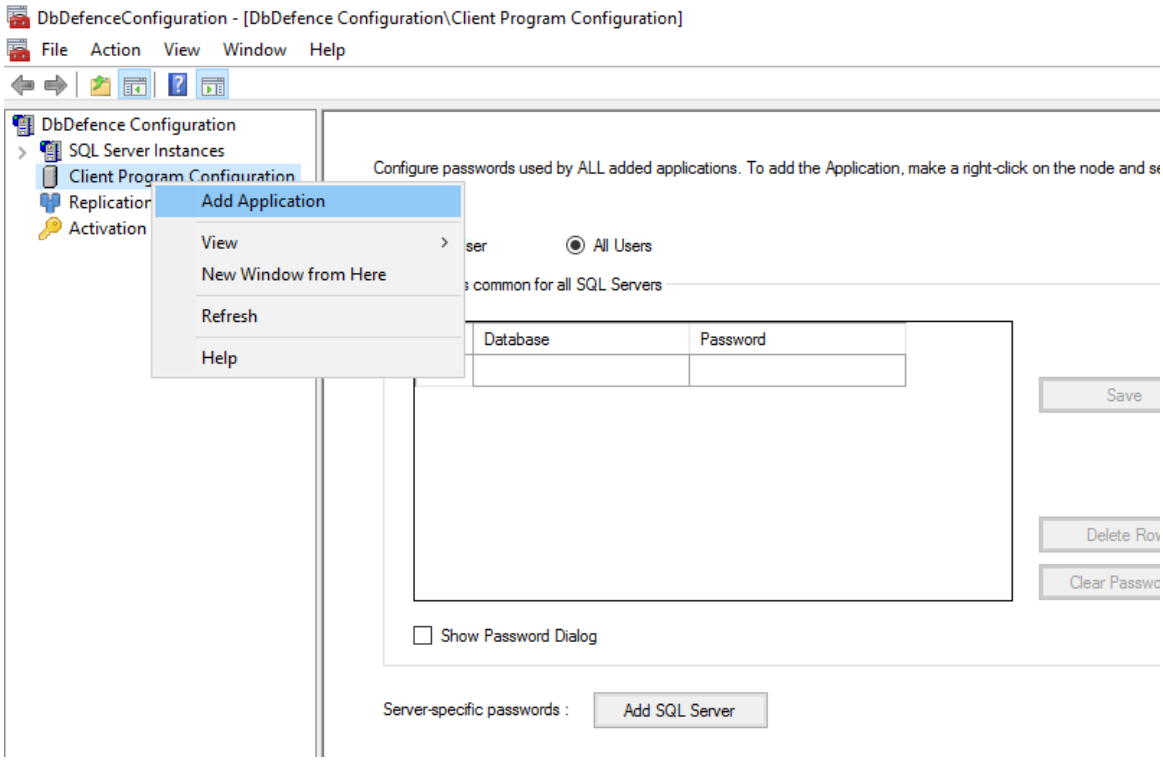
NOTE: If you're testing with Web Applications, please see the instructions at
https://www.database-encryption.com/support/dbdefence-documentation/dbdiis.html

Start DbDefence Configurator.



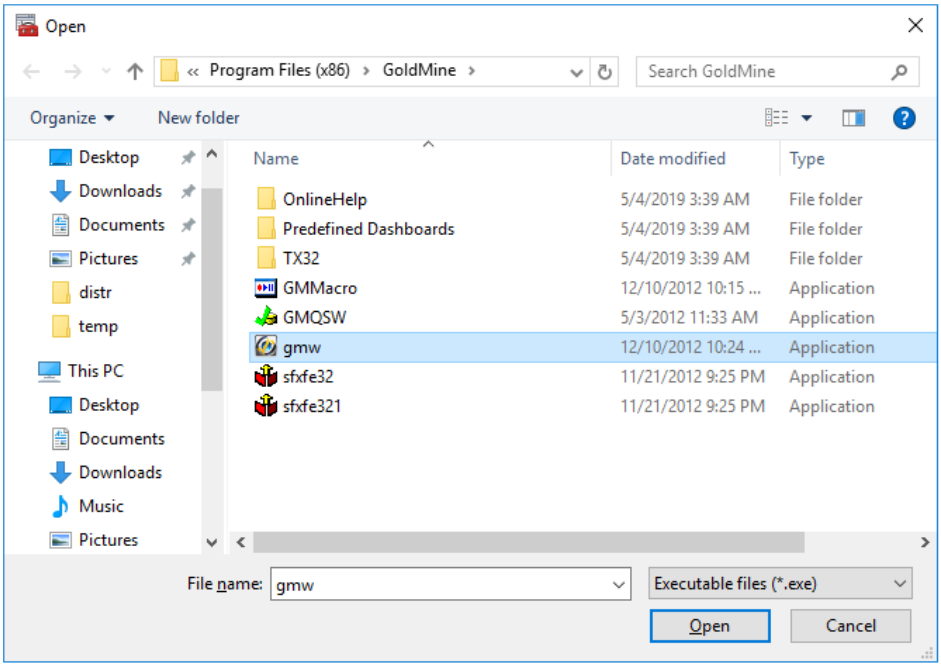
Pic. 17 Starting DbDefence Configurator

In Configurator's main window, click on "Client Program Configuration" and select "Add Application" from the popup menu.



Pic. 18 Adding Application

Navigate to GoldMine CRM executable. In our case, it is gmw.exe.

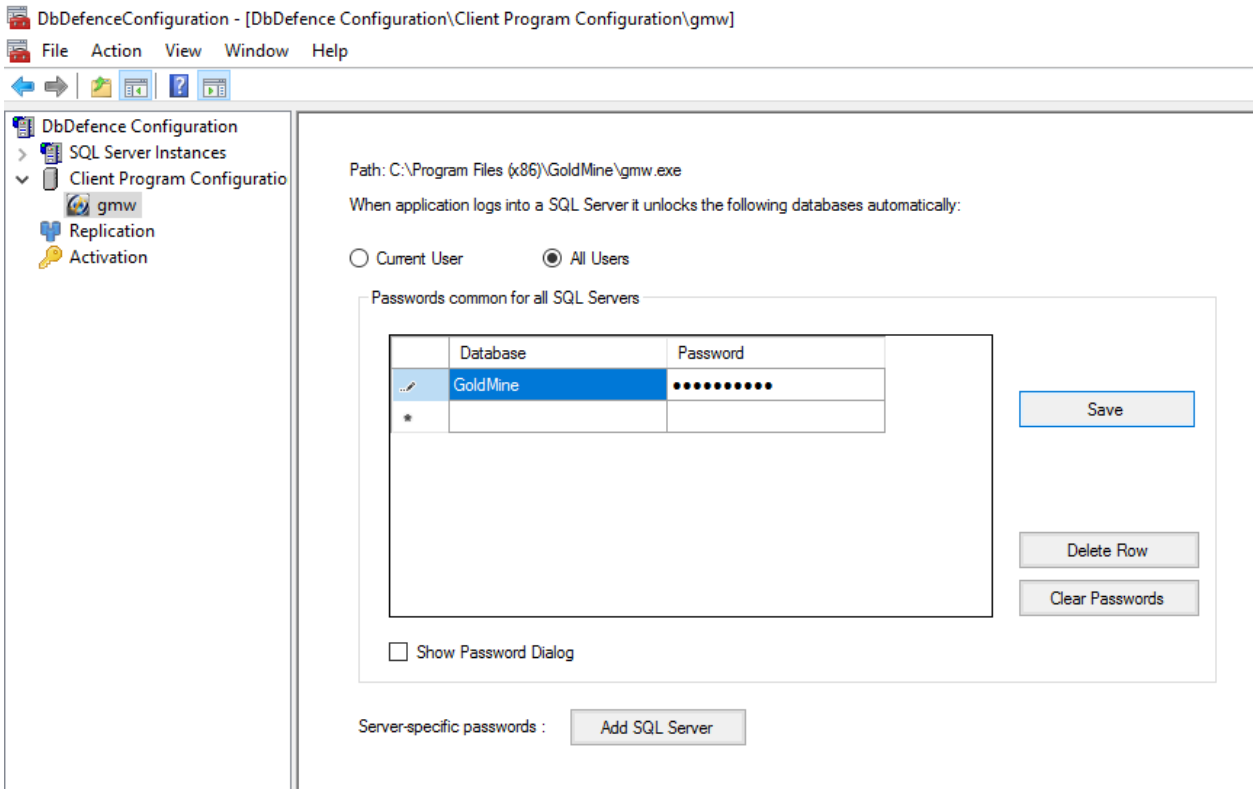


Pic. 19 Navigating to main executable

NOTE: If you're testing with Web Applications, please see the instructions at https://www.database-encryption.com/support/dbdefence-documentation/dbdiis.html

Enter the database name and encryption password as shown below. Don’t forget to click “Save”! It will securely store the password.

NOTE: Digitally sign applications to avoid substitution (when an authorized app is replaced with another one).



Pic. 20 Setting up the database name and encryption password for the executable

Then, restart GoldMine and connect to the database. The GoldMine app will see decrypted data. At the same time, all other applications with any login will see the masked data.

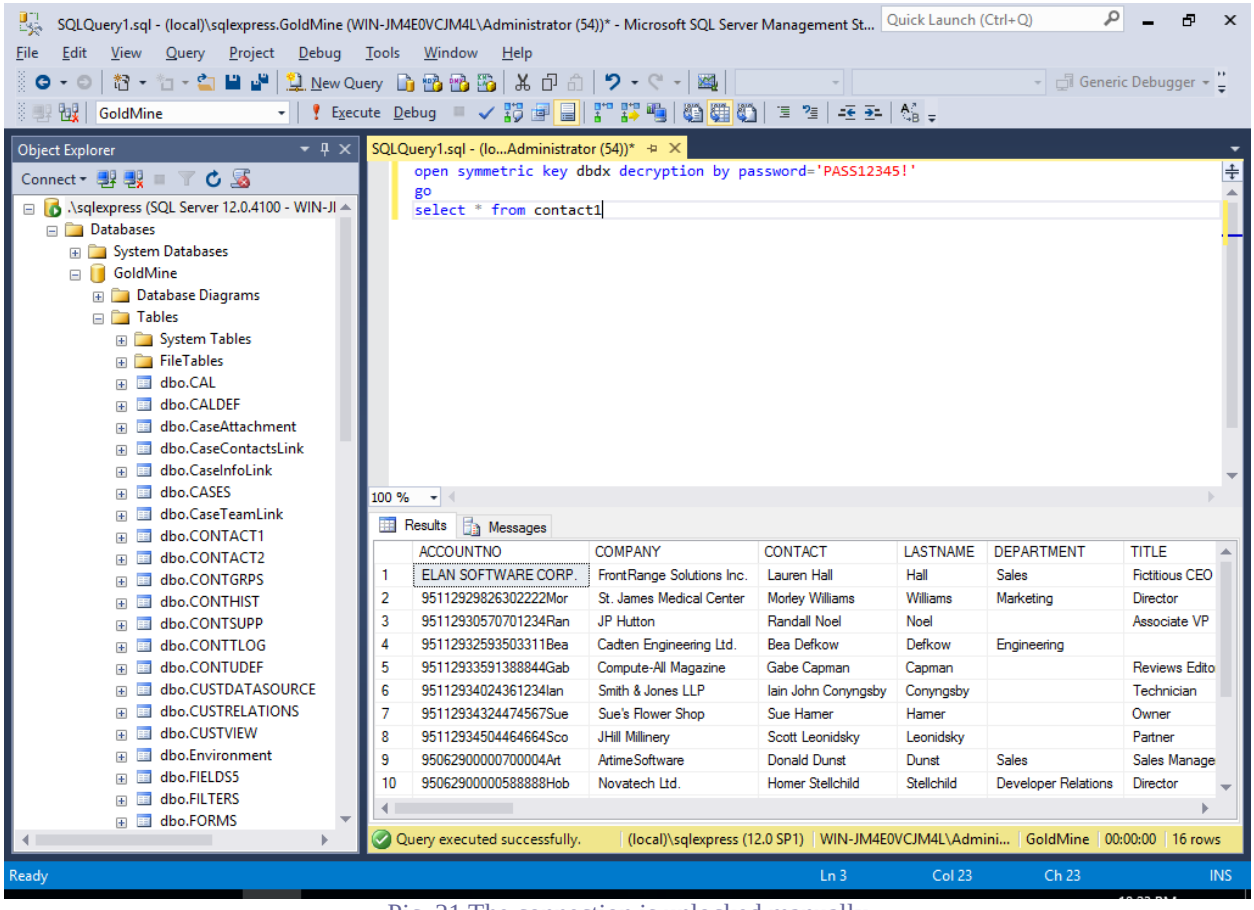
Method 3. Manual unlocking

This method is not automatic. It is suitable for DBAs, programmers, or anyone who knows SQL. Simply run:

OPEN SYMMETRIC KEY dbdx DECRYPTION BY PASSWORD='EncryptionPassword'

and connection will be unlocked.

NOTE: It is important to understand that this statement unlocks only the current connection. Other connections, even those opened from the same application, will not be unlocked.



Pic. 21 The connection is unlocked manually

If you open another Query Window and execute SELECT, it will display the masked data (unless access was granted with the previous methods).

To close access, use "CLOSE SYMMETRIC KEY dbdx".

Programmers may use OPEN and CLOSE SYMMETRIC KEY statements to explicitly unlock access from their applications.

Conclusion

In this document, we have shown how anyone can mask sensitive data without any special knowledge.

Software developing companies may purchase a redistribution license and install DbDefence as a part of their own software.

Please contact info@activecrypt.com for more information.