SQL Server 2016 Always Encrypted

Josip Šaban

EDU IT Pro, 17.02.2017

Microsoft Community

Sadržaj

- O predavaču
- Koncepti i povijesni razvoj enkripcije u SQL Serveru
- Osnove Always Encrypted arhitekture
- Ograničenja
- Demo

Microsoft Community

O predavaču

- FER, 2004-e diplomirao na smjeru računarstvo
- Kasnije završio MBA na Cotrugli-u, ako bude sreće do kraja 2018 trebao bi i doktorirati na FOI-u
- Bavi(o) se .NET programiranjem, poslovnom inteligencijom, bazama podataka, …
- Ostalo na https://www.linkedin.com/in/jsaban
- Kontakt na josipsaban@gmail.com

Microsoft Community

Koncepti i povijesni razvoj enkripcije u SQL Serveru

- SQL Server do verzije 2005 – nema ugrađenih alata za enkripciju (SSL/Protocol encryption)

Koncepti i povijesni razvoj enkripcije u SQL Serveru

- Simetrična/asimetrična enkripcija dostupna od SQL Server-a 2005, ali…morali ste ju programirati
  - OPEN SYMMETRIC KEY [MySSLCertificate]
    DECRYPTION BY CERTIFICATE [MySSLCertificate]
    SELECT FirstName, LastName,
    CONVERT(NVARCHAR(100), decryptbykey(SSN)) as 'SSN'
    FROM [dbo].[People]
- Enkripciju je odrađivao SQL Server Engine, ali ne sprečava DBA-a da vidi sve podatke

Microsoft Community

Koncepti i povijesni razvoj enkripcije u SQL Serveru

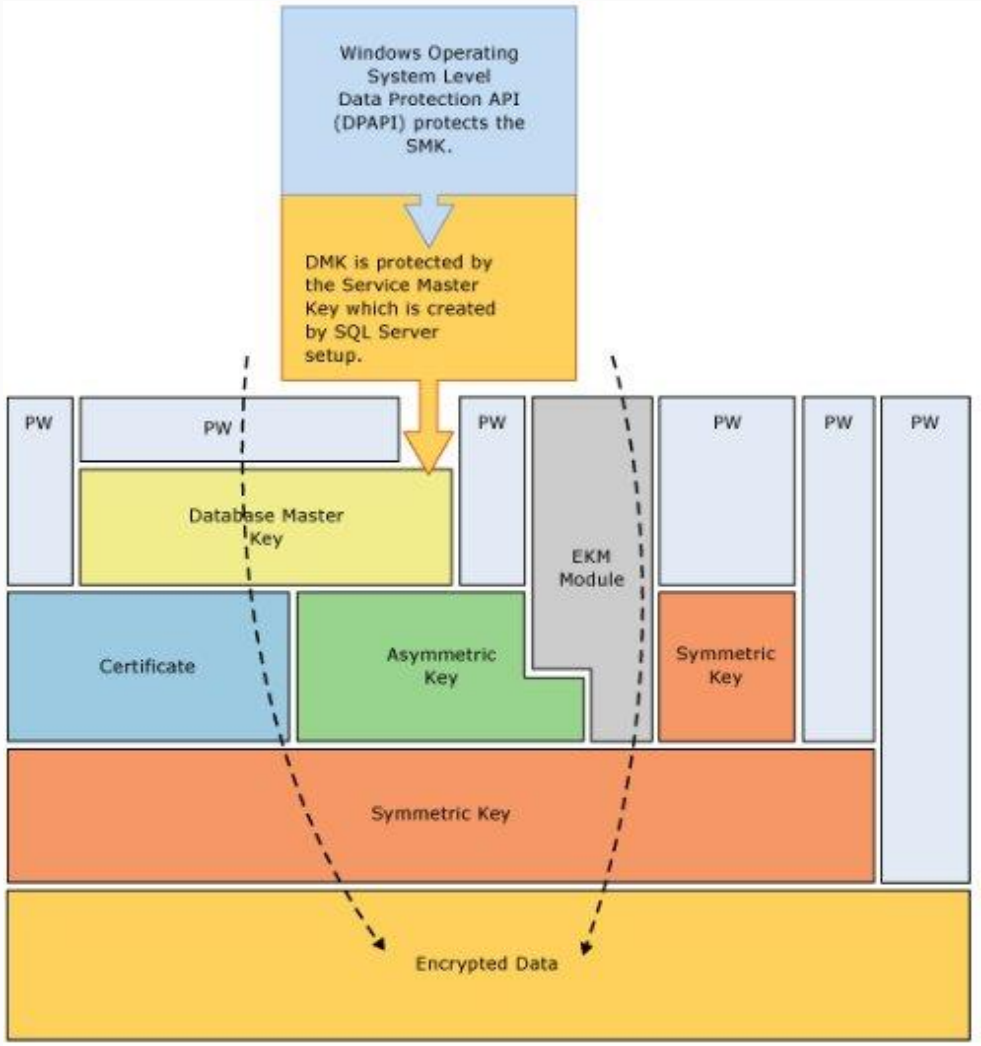
- SQL Server 2008 – TDE (Transparent data encryption) na nivou MDF/LDF datoteka, sigurnosne kopije
- Još se koriste third-party rješenja ili .NET biblioteke (System.Security.Cryptography)
  - Ukoliko se koriste vlastite biblioteke za enkripciju potrebno je znati detalje algoritma, inače se otvaraju „nevidljive” sigurnosne rupe

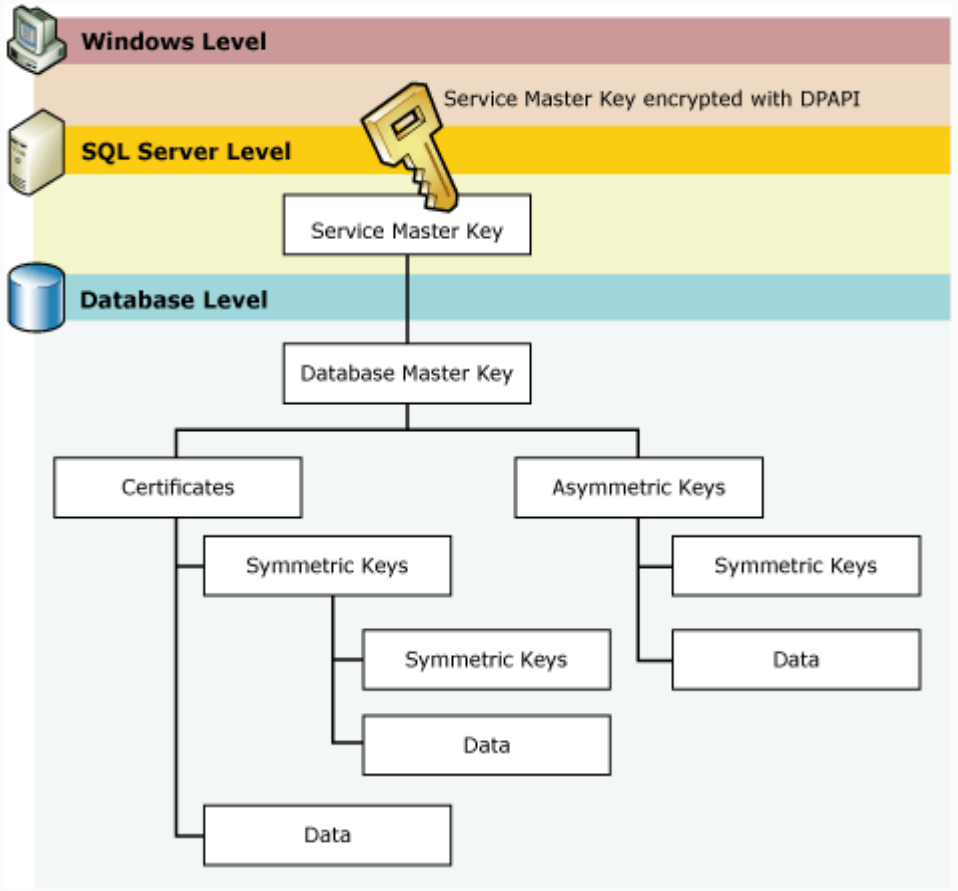
Koncepti i povijesni razvoj enkripcije u SQL Serveru

- SQL Server 2008 R2 / 2012 - enkripcija temeljena na certifikatima
- SQL Server 2016
  - Always Encrypted (tema ovog predavanja)
  - Dynamic Data Masking
    - Korisno za razvoj aplikacija
    - 123-45-6789 = XXX-XX-6789
    - Prihvatljivo za prikaz podataka...ali i dalje nesigurno - DBA ili neki drugi korisnik sa dovoljnim pravima može vidjeti podatke u bazi

Microsoft Community

Koncepti i povijesni razvoj enkripcije u SQL Serveru





Microsoft Community

Always Encrypted arhitektura

- Zašto kriptirati:
  - Sigurnost
  - Regulatorna podrška
  - PII standardi („Personally Identifiable Information”)
    - NIST Special Publication 800-122
    - Credit Card Numbers
    - Social Security Numbers
    - Names, Address, Biometrics, etc.
    - “People with the proper authority can view my data, but how do I keep the DBA from viewing it?”

Always Encrypted arhitektura

- Performanse
  - Enkripcija i dekripcija se rade na klijentu/srednjem sloju
- Sigurnost
  - Podaci su jedino vidljivi uz postojanje certifikata

Microsoft Community

Always Encrypted arhitektura

- https://msdn.microsoft.com/en-us/library/mt163865.aspx
- Podaci se nikad ne nalaze u nekriptiranom stanju, bilo da su neaktivni (nisu u transakcijama ili dohvatima) ili dok su aktivni (uključujući i dok su u memoriji)
- Samo klijenti sa odgovarajućim master ključem imaju pristup nekriptiranim podacima

Always Encrypted arhitektura

- Ovaj pristup se može kombinirati sa TDE-om za kompletno kriptografsko rješenje
- Always Encrypted zatvara „rupu” u kriptografskim rješenjima na SQL Server platformi, osobito u slučaju regulatornih zahtjeva – administratori baze podataka nemaju pristup podacima

Always Encrypted arhitektura

- Zahtjevi
  - SQL Server 2016 (Enterprise/Developer Editions only, Azure)
  - .NET 4.6 ili viši
    - Mora korisiti ADO.NET
  - Certificate store
    - Koristi se za spremanje Master ključa
  - String podaci moraju koristiti binary2 collation
    - Latin1_General_BIN21

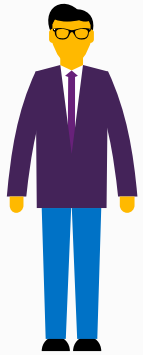
Always Encrypted arhitektura

- Always Encrypted podržava dvije vrste enkripcije:
  - Randomiziranu enkripciju
    - Same data values will have different encrypted values
    - (225) 555-1234 = 0x0003456
    - (225) 555-1234 = 0x00078910
    - More secure
    - Can't allow comparisons (no JOINS)
    - No Indexes
  - Deterističku enkripciju
    - Same data values will have same encrypted values
    - (225) 555-1234 = 0x0003456
    - (225) 555-1234 = 0x0003456
    - Less secure
    - Allows comparisons (JOINS)

Always Encrypted arhitektura

- Always Encrypted koristi dvije vrste ključeva:
  - Column Encryption Keys (CEK)
  - Column Master Keys (CMK)
- CEK ključevi se koriste za enkripciju CMK-ova
- CMK-ovi se koriste za enkripciju jedne kolone u tablici
  - Na svaki CMK se mogu vezati dva CEK-a što omogućava rotaciju CEK-ova (npr. u slučaju isticanja validnosti ključa)
  - CMK se mora instalirati na svaki klijentski stoj koji treba pristup nekriptiranim podacima – na SQL Server ti ključevi NISU instalirani

Always Encrypted arhitektura



1. Generate CEKs and Master Key

Column Encryption Key (CEK)

Column Master Key (CMK)

2. Encrypt CEK

Encrypted CEK

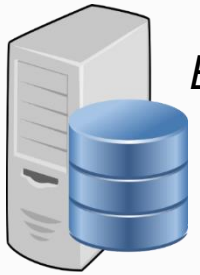
3. Store Master Key Securely

CMK Store:

- *Certificate Store*
- *HSM*
- *Azure Key Vault*
- *…*

CMK

4. Upload Encrypted CEK to DB

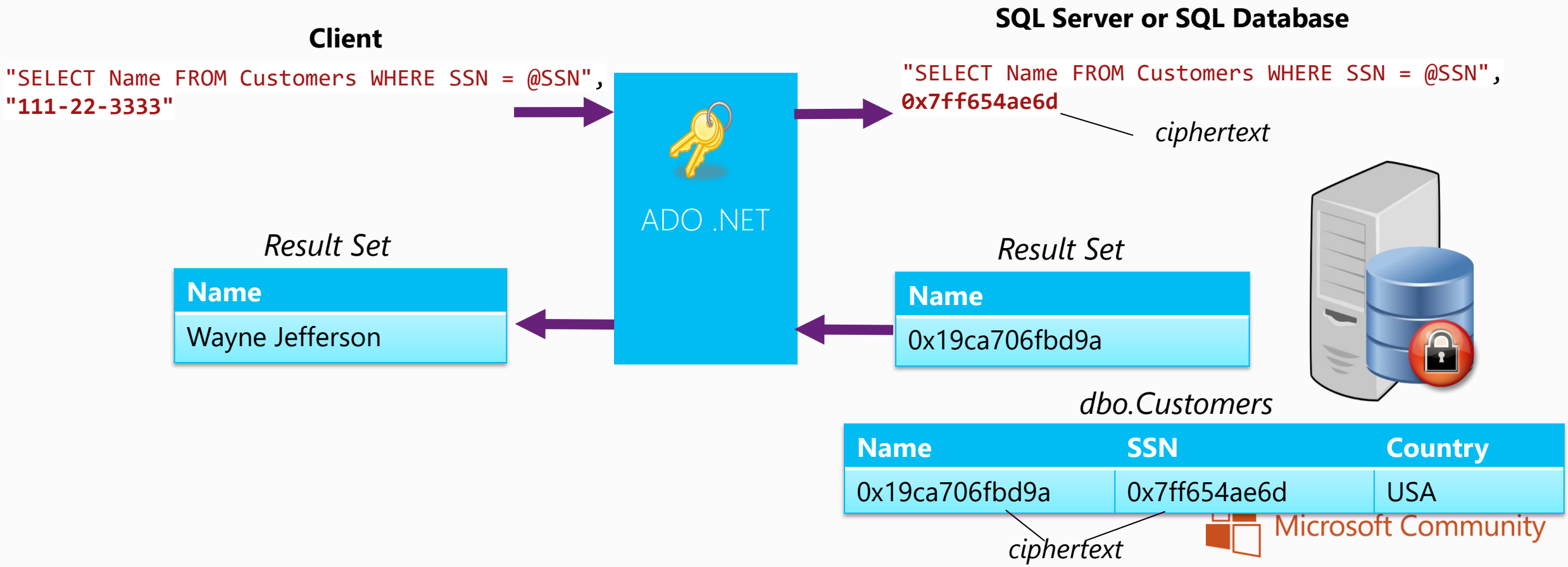


Encrypted CEK

Database

Always Encrypted arhitektura - MSDN

Encrypted sensitive data and corresponding keys are never seen in plaintext in SQL Server



Ograničenja

- Replikacija ne radi
  - Transactional ili Merge
- Distributed Queries (Linked Servers) nisu dozvoljeni
- Tipovi podataka koji nisu dozvoljeni
  - XML, timestamp/rowversion, image, ntext, text, sql_variant, hierachyid, geography, geometry, alias, user-defined
- FOR XML, FOR JSON, Check Constraints nisu dozvoljeni
- Change Data Capture ne radi
- SQL Server Data Tools (SSIS) ne radi
- Dozvoljen je samo operator jednakosti (=) a upiti moraju slati tipizirane parametre (SQLCommand, SQL Parameter)
- I još neki...https://msdn.microsoft.com/en-us/library/mt163865.aspx

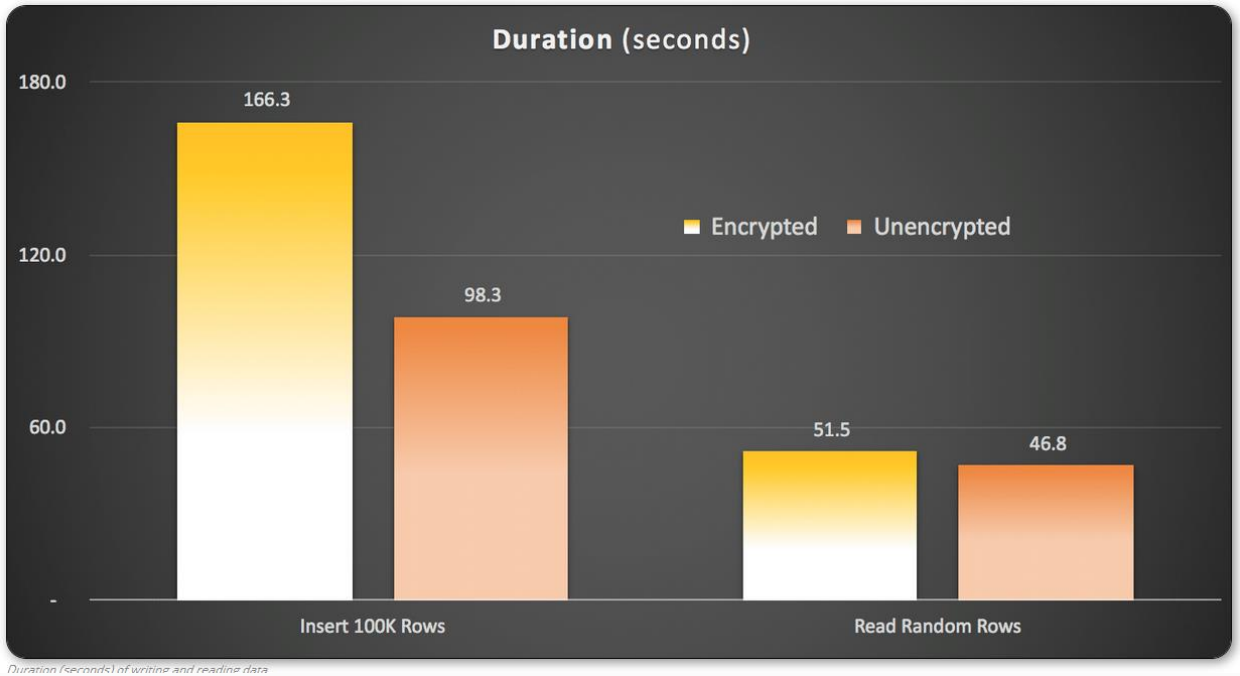
Ograničenja

MSDN - when using the current version of the wizard, you need to make sure no other application inserts or updates rows in the tables, containing encrypted columns, while the encryption workflow is running. During the encryption workflow, the wizard creates a temporary table, downloads the data from your original table, encrypts the data and uploads it to the temporary table. Finally, the wizard deletes the original table and renames the temporary table to the original table. If another app is inserting or modifying data in the original table, the new or updated data may be lost. Make sure, you only run the encryption workflow in a planned maintenance window. This issue will be addressed in a later version of SSMS.

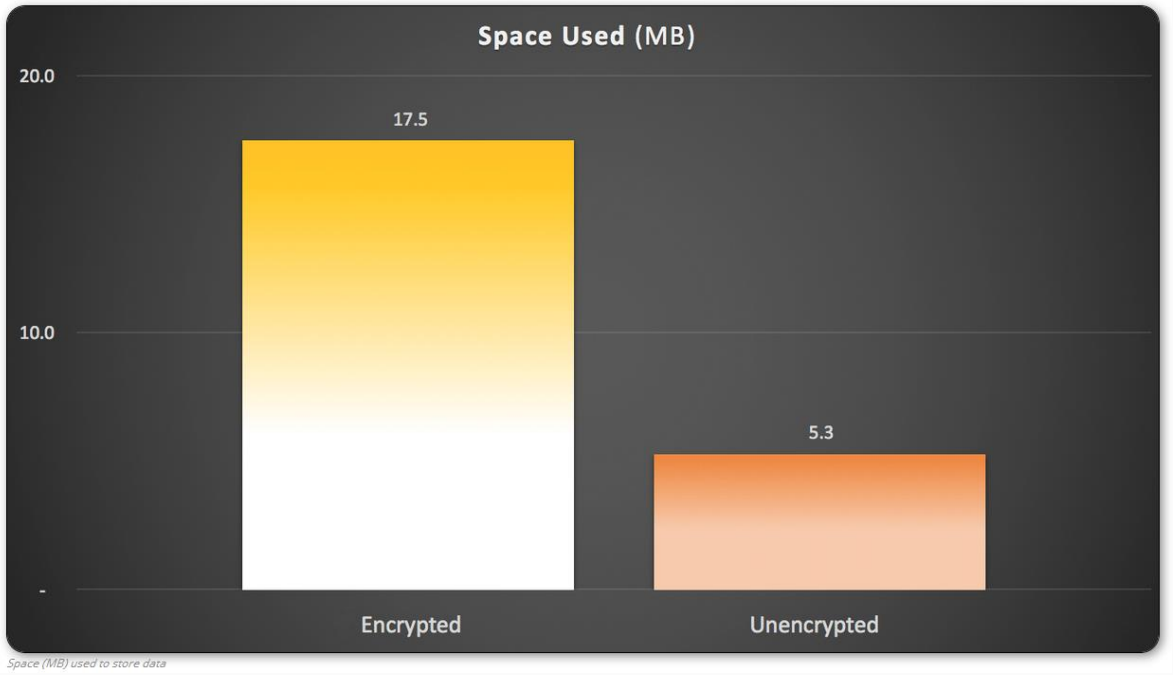
https://blogs.msdn.microsoft.com/sqlsecurity/2015/10/31/ssms-encryption-wizard-enabling-always-encrypted-in-a-few-easy-steps/

Ograničenja - performanse

https://sqlperformance.com/2015/08/sql-server-2016/perf-impact-always-encrypted



Duration (seconds) of writing and reading data



Space (MB) used to store data

Microsoft Community

DEMO

Demo

- Aplikacija radi kriptiranje – šalje čist tekst ADO.NET driveru, nakon čega se kriptirani podaci šalju bazi
- Jedina promjena u .NET aplikaciji je promjena connection stringa, koja ukazuje da je uključena enkripcija
  - Nakon toga se ADO.NET brine o kriptiranju i de-kriptiranju podataka
- Cilj – onemogućiti DBA/sysadmin korisnicima pristup dekriptiranim podacima

Demo

- Dva ključa su osnova cijelog procesa – Column Master Key (na klijentskom stroju, u key store-u) koji služi zaštiti Colum Encryption Key-a
  - Ovakva arhitektura onemogućava dekriptiranje podataka SQL Server-u
- Drugi ključ (Column Encryption key) se sprema na SQL Server, te služi za kriptiranje i dekriptiranje Always Encrypted kolona u bazi

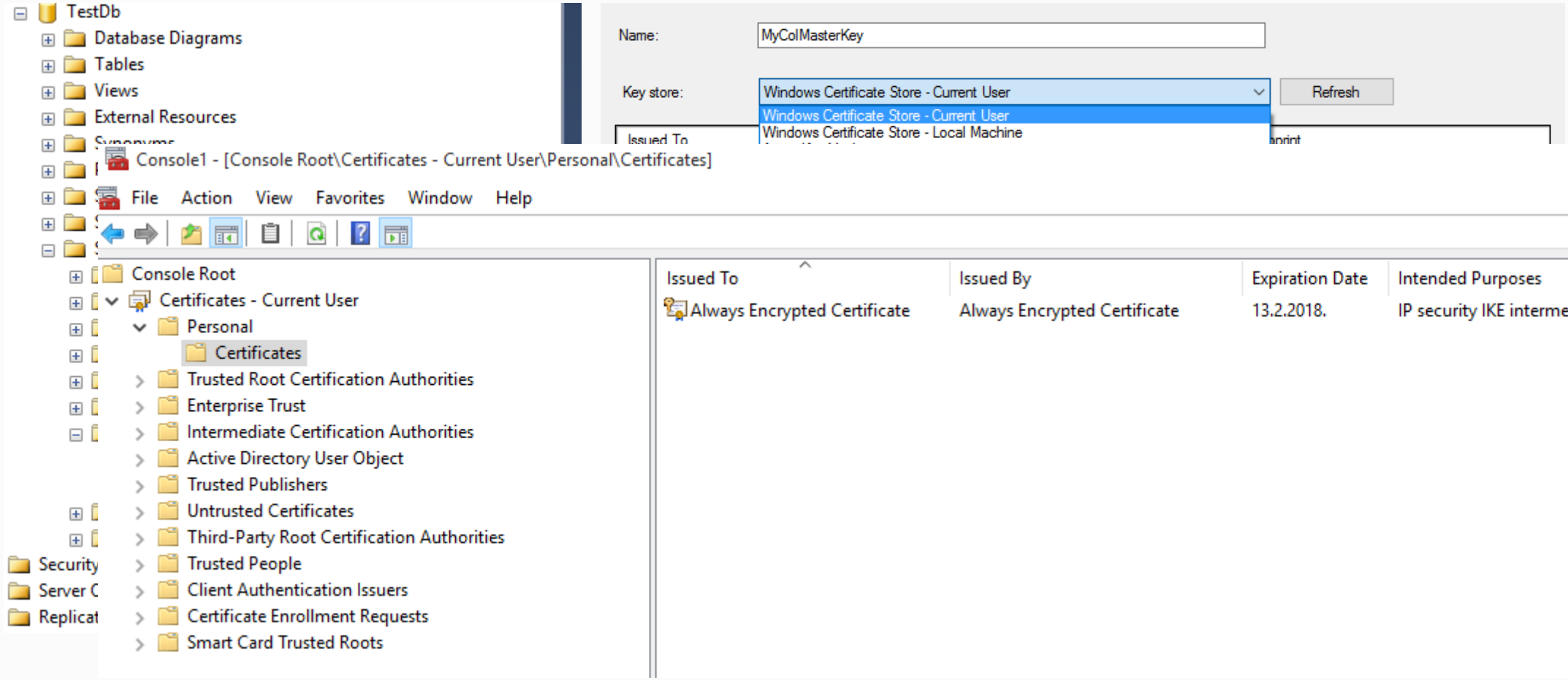
Demo

- Jednom kad je ADO.NET dekriptirao Column Encryption key, korištenjem Column Master key-a, može ga iskoristiti za kriptiranje i dekriptiranje Always Encrypted kolona
- Stoga, da postavimo arhitekturu potrebno je:
  - Aplikacija koja koristi .NET 4.6 framework
  - Instanca SQL Server-a 2016 (samo Windows 10 ili Windows Server 2012 ☹)
  - Certificate store – u koji spremamo Column Master Key
  - Column Master Key
  - Column Encryption Key
  - Tablica u bazi sa Always Encrypted kolonama

Demo

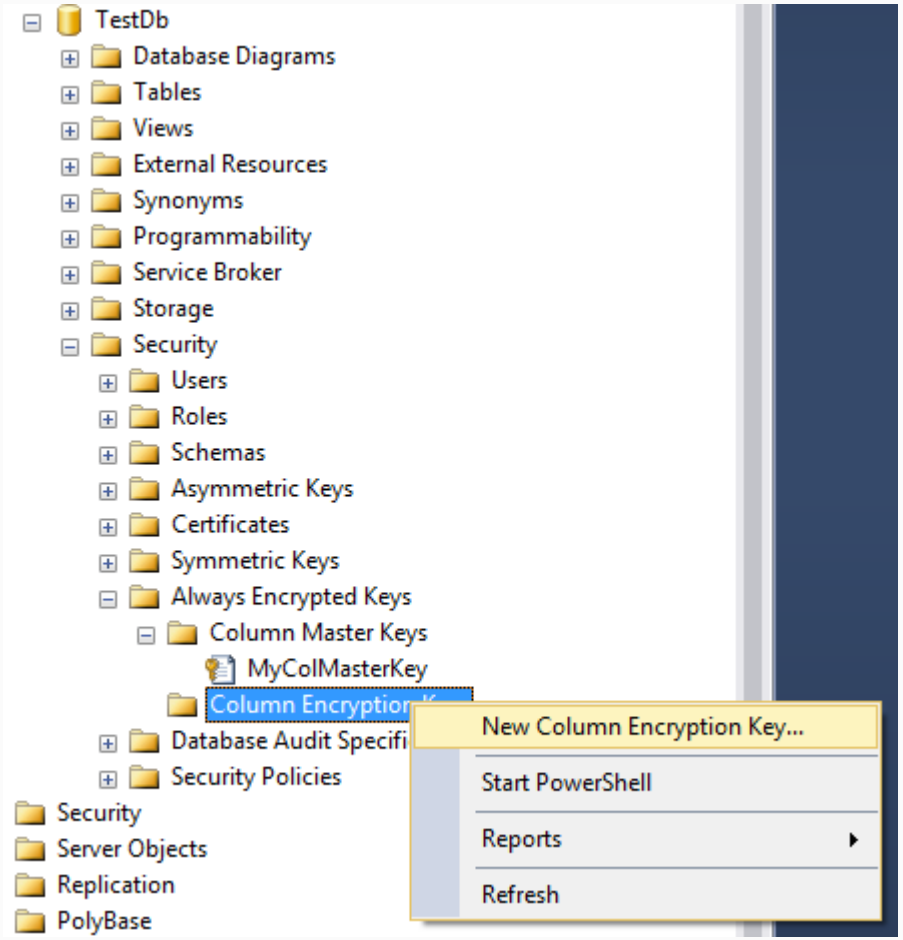
- Jednom kad je ADO.NET dekriptirao Column Encryption key, korištenjem Column Master key-a, može ga iskoristiti za kriptiranje i dekriptiranje Always Encrypted kolona
- Stoga, da postavimo arhitekturu potrebno je:
  - Aplikacija koja koristi .NET 4.6 framework
  - Instanca SQL Server-a 2016 (samo Windows 10 ili Windows Server 2012 ☹)
  - Certificate store – u koji spremamo Column Master Key
  - Column Master Key
  - Column Encryption Key
  - Tablica u bazi sa Always Encrypted kolonama

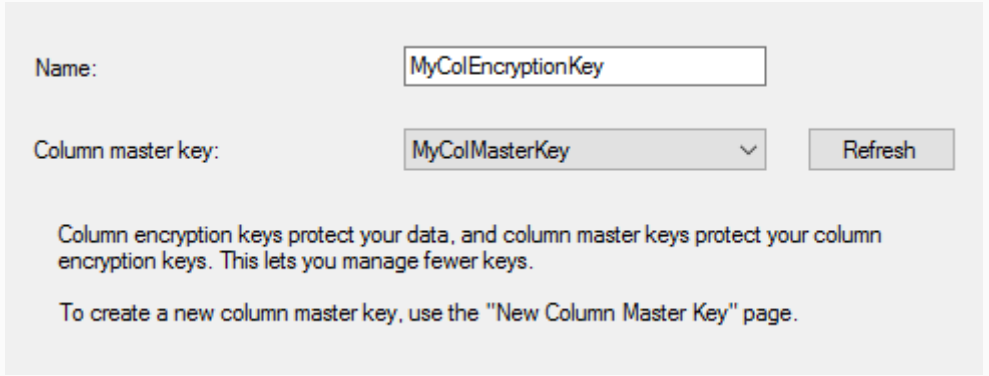
Column Master Key



Microsoft Community

Column Encryption Key





Microsoft Community

Stvaramo tablicu za demo

```
CREATE TABLE dbo.MojMaliDemo
(
  ID INT IDENTITY(1,1) PRIMARY KEY,

  StringOne NVARCHAR(255),
  StringTwo NVARCHAR(255),
  NekiBitanDatum DATE ENCRYPTED WITH (
     ENCRYPTION_TYPE = RANDOMIZED, ALGORITHM = 'AEAD_AES_256_CBC_HMAC_SHA_256',
      COLUMN_ENCRYPTION_KEY = MyColEncryptionKey ),
  NekiBitanString NVARCHAR(50) COLLATE Latin1_General_BIN2 ENCRYPTED WITH (
              ENCRYPTION_TYPE = DETERMINISTIC, ALGORITHM = 'AEAD_AES_256_CBC_HMAC_SHA_256',
              COLUMN_ENCRYPTION_KEY = MyColEncryptionKey )
);
```

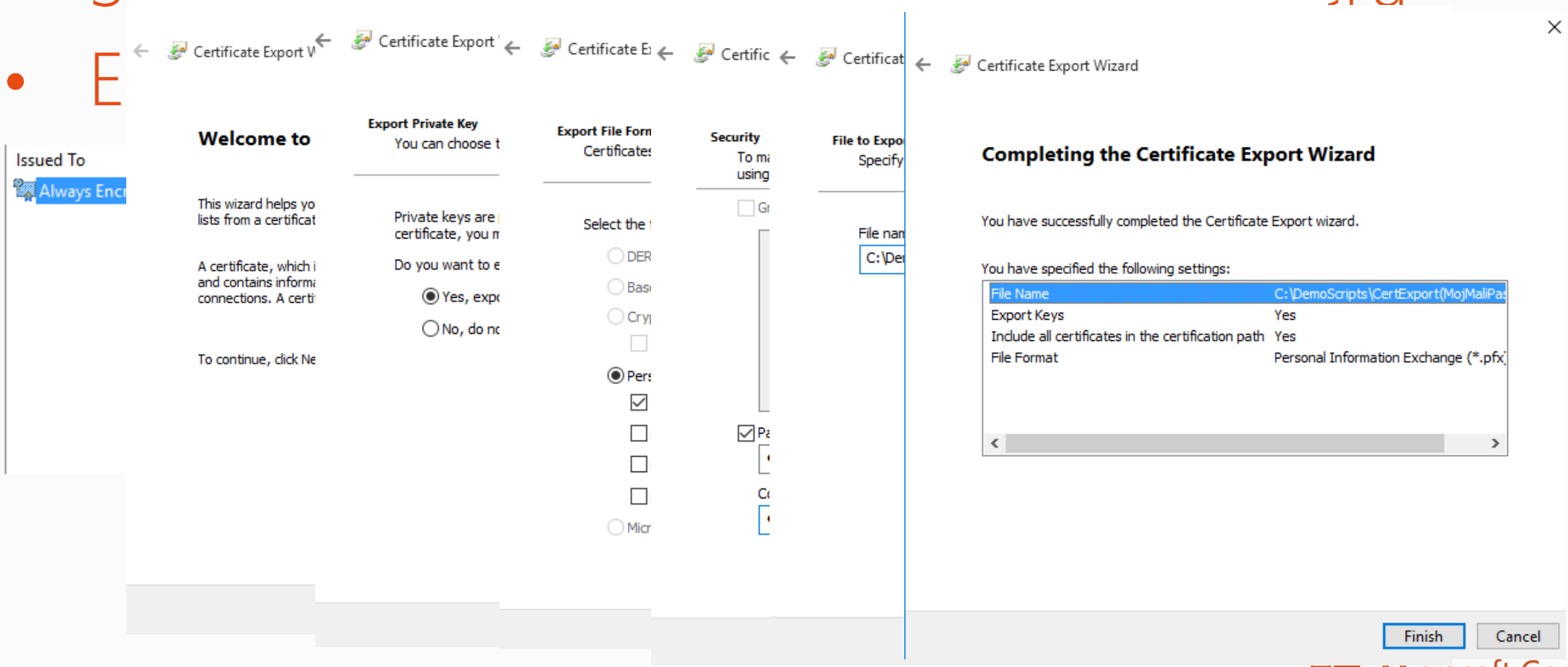
Procedura za unos

```
CREATE PROCEDURE AE_Insert
(
@StringOne NVARCHAR(255),
@StringTwo NVARCHAR(255),
@NekiBitanDatum DATE,
@NekiBitanString NVARCHAR(50)
)
AS
INSERT INTO dbo.MojMaliDemo (StringOne, StringTwo, NekiBitanDatum, NekiBitanString)
VALUES (@StringOne,@StringTwo,@NekiBitanDatum,@NekiBitanString);

-- Ne ide standardno :( - nije poslano ADO.NET enkriptirano
EXEC AE_Insert @StringOne = 'AAAAA',
               @StringTwo = 'BBBBB',
               @NekiBitanDatum = '2016-01-01',
               @NekiBitanString = 'MojBitanString';
/*
Msg 206, Level 16, State 2, Procedure AE_Insert, Line 0 [Batch Start Line 27] Operand type clash: varchar is incompatible with date encrypted with (encryption_type =
'RANDOMIZED',  encryption_algorithm_name = 'AEAD_AES_256_CBC_HMAC_SHA_256',
column_encryption_key_name = 'MyColEncryptionKey', column_encryption_key_database_name = 'TestDb')
*/
```

Key point – import/eksport certifikata

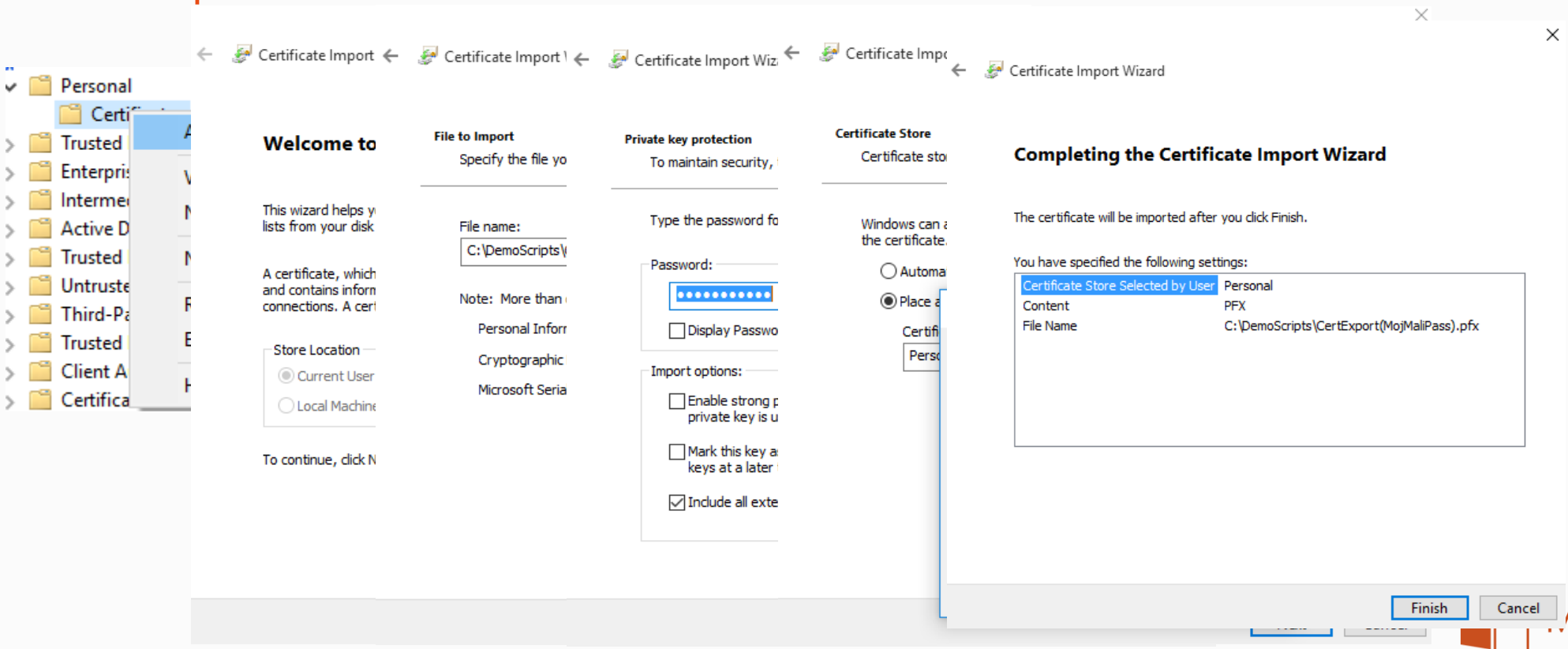
- Stvarni svijet – klijent nije na istom serveru
- E



Microsoft Community

Key point – import/eksport certifikata

- Stvarni svijet – klijent nije na istom serveru
- Import:



Microsoft Community

C# kod

- Kod je u dodatku ove prezentacije
- Nakon izvršenja koda treba provjeriti
  - Koje podatke vidi trenutno ulogirani korisnik
  - Koje podatke vidi neki drugi, administratorski, korisnik

```
SELECT *  FROM dbo.MojMaliDemo
```

- Pitanje za plus bodove – zašto „ovo” vide i svi lokalni korisnici?

Microsoft Community

Odgovor bitna iznimka– trenutna verzija

If you ran your SSMS from the same box as you used to run your C# app (that does have the CMK in its certificate store) this behavior should be by design - it just means the SSMS itself can use Ado.NET for encrypting/decrypting data. As far as I understand DBAs are NOT meant to use the same computers as DB users - only in this case Always Encrypted can work as expected.

Microsoft Community

Zaključak i pitanja

- Always Encrypted samo traži promjenu connection string-a
- Podaci su zaštićeni u trenutku napuštanja klijentovog računala
- Kriptirani podaci su zaštićeni od administratora servera i baze podataka
- Upravljanje certifikatima je ključno za zaštitu podataka

Microsoft Community

Hvala na pažnji!

Microsoft Community