Microsoft

Security - Encryption

Module 5

Learning Units covered in this Module

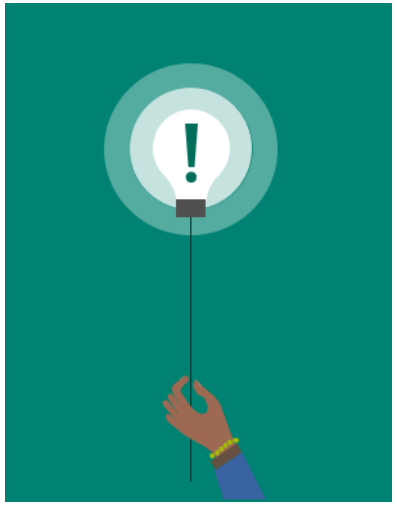
- Lesson 1: Backup Encryption
- Lesson 2: Transparent Data Encryption (TDE)
- Lesson 3: Encrypted Connections
- Lesson 4: Column Level Encryption
- Lesson 5: Overview of Always Encrypted

Lesson 1: Backup Encryption

Objectives

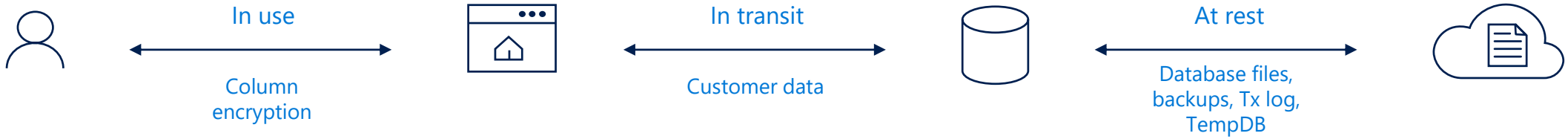
After completing this learning, you will be able to:

- Understand the backup encryption feature.



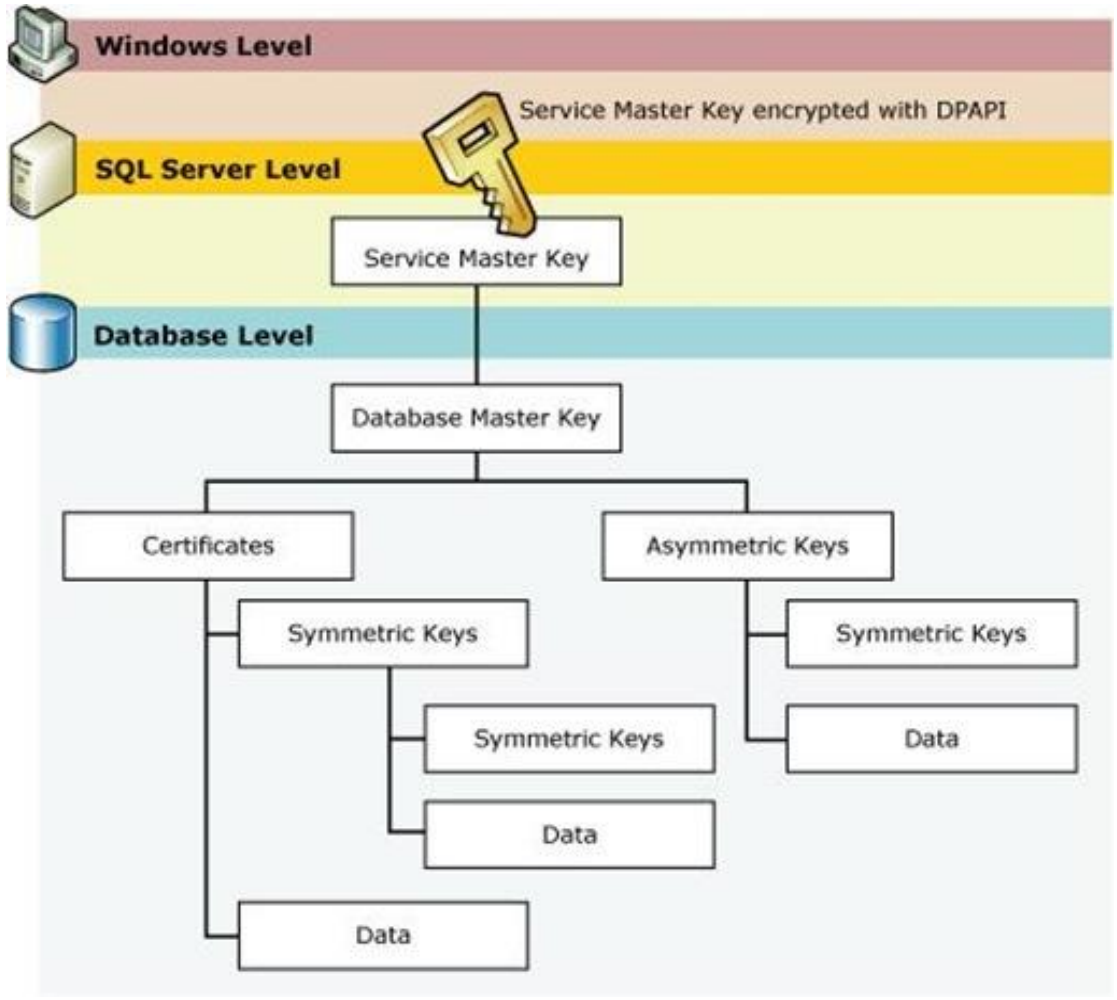
Types of data encryption

Data encryption	Encryption technology	Customer value
In transit	Transport Layer Security (TLS) from the client to the server.	Protects data between client and server against snooping and man-in-the-middle attacks.
At rest	Transparent Data Encryption (TDE) for Azure SQL Database.	Protects data on the disk. Key management is done by Azure, which makes it easier to obtain compliance.
In use (end-to-end)	Always Encrypted for client-side column encryption.	Data is protected end-to-end, but the application is aware of encrypted columns. This is used in the absence of data masking and TDE for compliance-related scenarios.



Encryption Hierarchy

- Service Master Key
- Database Master Key
- Certificates
- Database Encryption Key (DEK)



Backup Encryption

Available in Standard or Enterprise Edition

Encrypts native backup files

Based on certificate stored securely within SQL Server engine

Supported algorithms are AES (128, 192, 256) and Triple DES

Backup Encryption Prerequisites

Create a Database Master Key for the master database

Create a certificate or asymmetric key to use for backup encryption

It is very important to back up the certificate or asymmetric key

```
BACKUP DATABASE AdventureWorks2016 TO DISK = N'D:\DATA\ADWorkSecure.bak'
    WITH ENCRYPTION(ALGORITHM = AES_256, SERVER CERTIFICATE = BackupCert)
```

Backup Encryption Restrictions

If using asymmetric keys, must store on an EKM provider

Express and Web editions do not support backup encryption.

Restoring encrypted backups to Express and Web editions is allowed

Appending to existing backup sets in not supported

Demonstration

Backup Encryption

Backup Encryption with SQL Server Management Studio (SSMS)

Questions?

Knowledge Check

Which SQL Server Editions support backup encryption?

When would you use backup encryption?

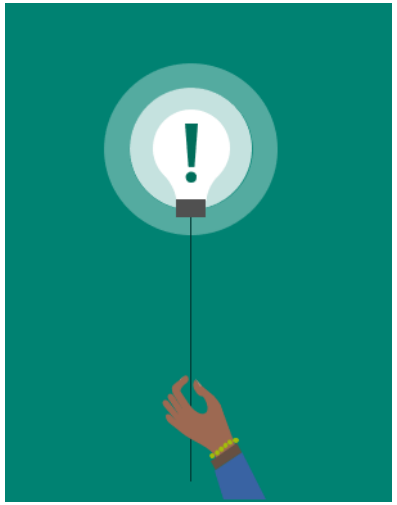
What are some key considerations when using backup encryption?

Lesson 2: Transparent Data Encryption (TDE)

Objectives

After completing this learning, you will be able to:

- Understand what protection Transparent Data Encryption (TDE) offers and how it works.



Transparent Data Encryption Benefits

Performs all the cryptographic operations at the database level

Removes any need for application developers to create custom code to encrypt and decrypt data

Data is encrypted as it is written to disk and decrypted as it is read from disk

Because SQL Server manages encryption and decryption transparently, there is no need for application changes

Protects data, files, and backups at rest

How Transparent Data Encryption works

Entire database is encrypted

Protects data, files, and backups at rest

Tempdb encrypted by default when any database has TDE enabled

Backups are also encrypted for TDE databases

How Transparent Data Encryption works

Works at storage I/O level (encryption at rest)

Check the status of encryption using **sys.dm_database_encryption_keys**

Encryption happens before writing to disk and performed by background threads

- Page protection (checksum/torn page) is applied after encryption
- Page protection (e.g. checksums) is checked before decryption
- Database pages are decrypted when read into memory

Why Transparent Data Encryption

Securing data at rest

No changes in the application layer

Performance should not be affected

Scalability

Space should not be increased or affected

Supports AES and 3DES encryption algorithms

Impact of Transparent Data Encryption

Performance Impact

- Encryption or decryption scan
- Query impact

Backup/Restore and Detach/Attach

- Certificate should have two files
- Backup both files

Key Management

- If unwanted access to the key should happen, consider changing the certificate

High Availability

- Create a Master Key on the mirror (secondary replica or stand by server)
- Backup the certificate on the principal and restore it on the mirror.

Questions?

Knowledge Check

What does TDE encrypt?

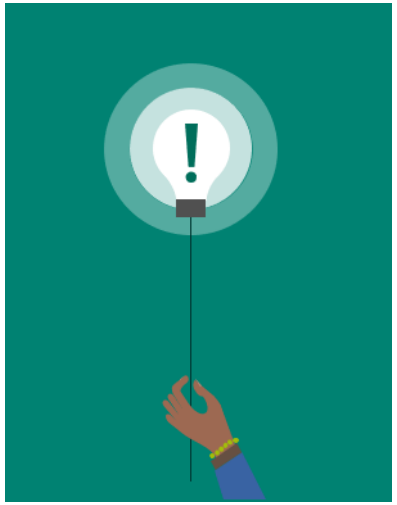
What are some important considerations when deploying TDE?

Lesson 3: Encrypted Connections

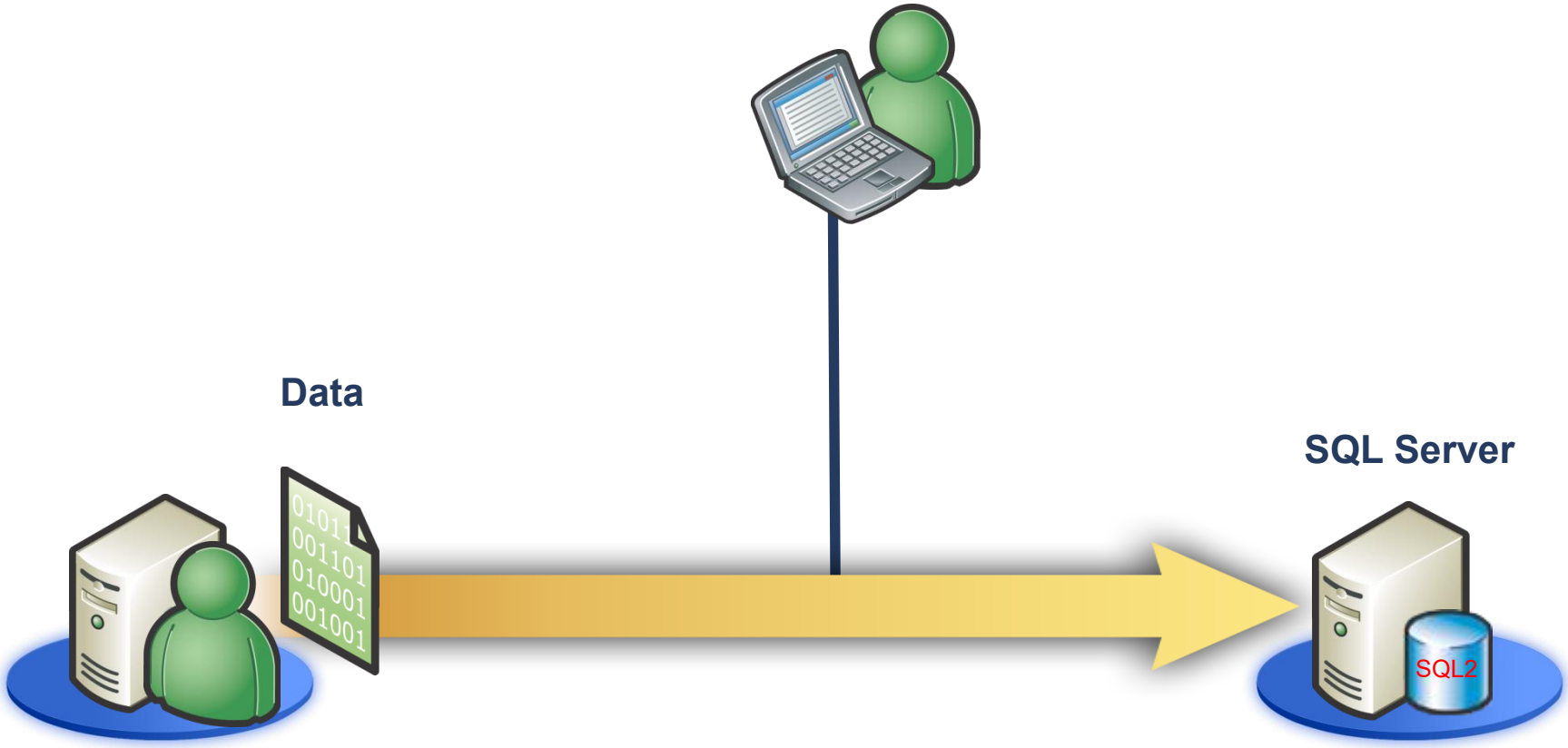
Objectives

After completing this learning, you will be able to:

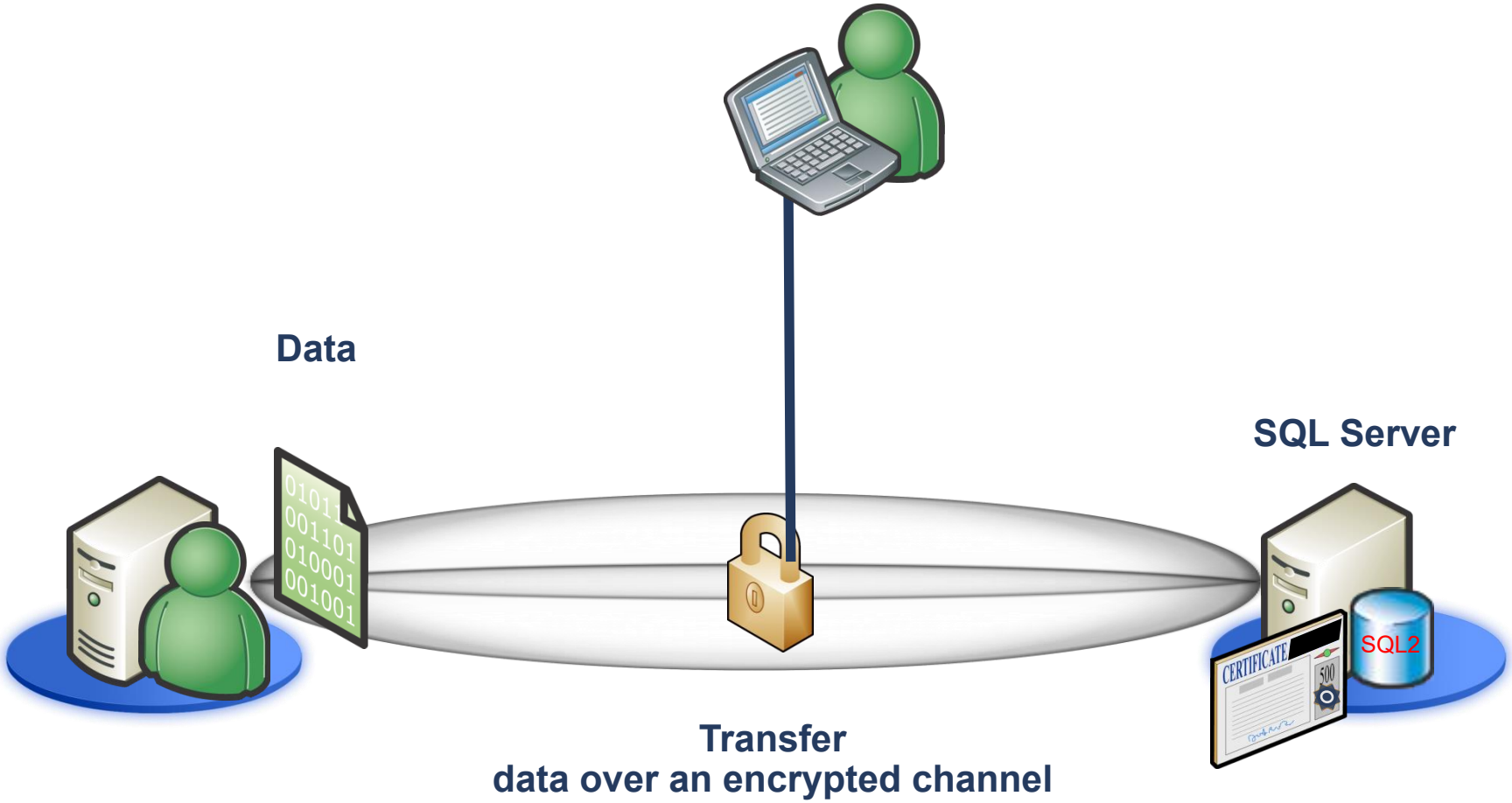
- Understand what Secure Socket Layer (SSL) Encryption offers and how it works at a high level.



Why do we need to encrypt data in transit?



Protected Communication Channel



SQL Server Encrypted Connections

SQL Server uses Transport Layer Security (TLS) for secure connections.

Configured using the SQL Server Configuration Manager

Server computer must have a certificate provisioned.

Client machine must have a trust with the certificate’s root authority.

Certificate Requirements

The certificate must be in either the local computer certificate store or the current user certificate store.

The SQL Server Service Account has the permissions to access the TLS certificate.

The current system time must be after the **Valid from** property of the certificate and before the Valid to property of the certificate.

The certificate must be meant for server authentication. This requires the **Enhanced Key Usage** property of the certificate to specify **Server Authentication (1.3.6.1.5.5.7.3.1)**.

Install a Certificate on a Server

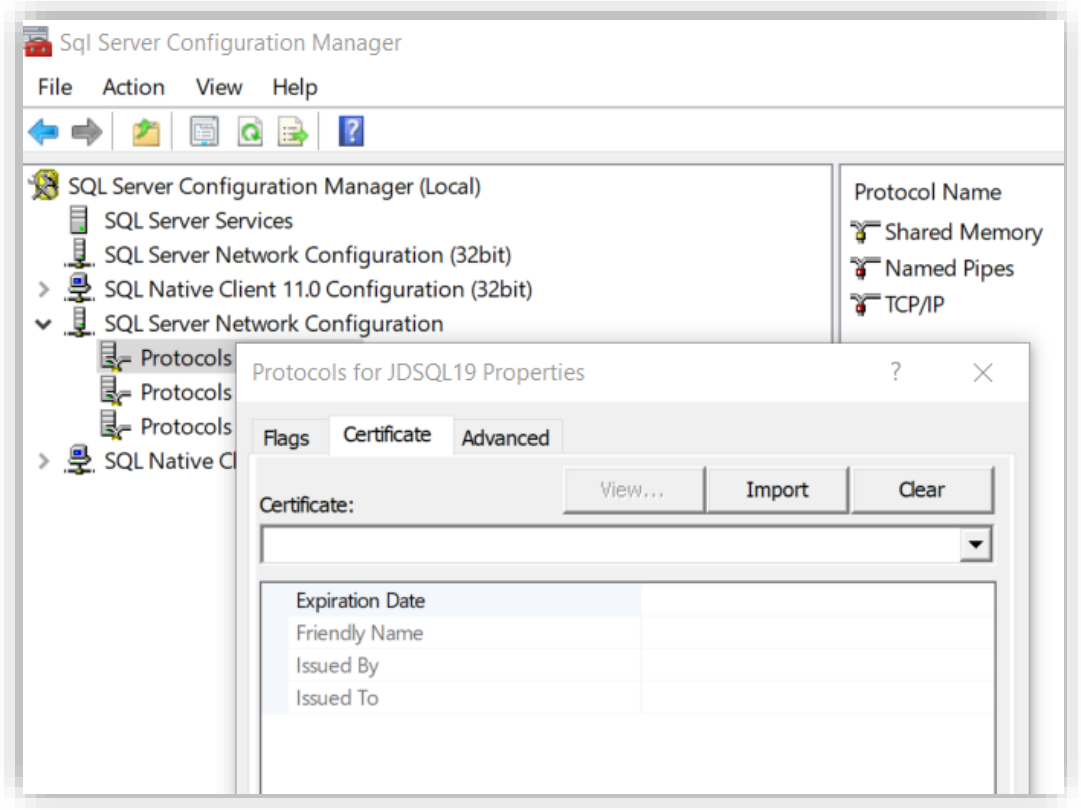
Expand SQL Server Network Configuration.

Right-click Protocols for <instance Name> and then select Properties.

Choose the Certificate tab, and then select Import.

Select Browse and then select the certificate file.

Select Next to validate the certificate.

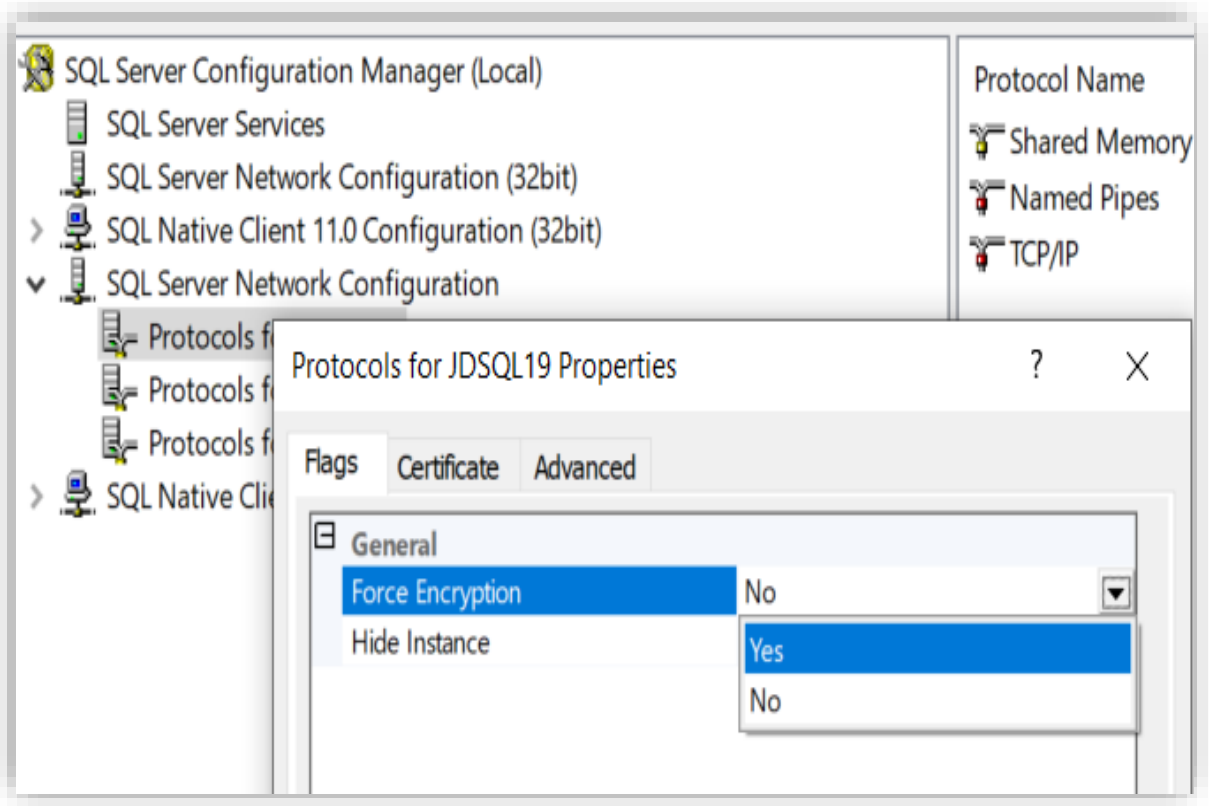


Force Encryption on the SQL Server

Expand **SQL Server Network Configuration**.

Right-click **Protocols for** <*instance Name*> and then select **Properties**.

Change **Force Encryption** to **Yes**

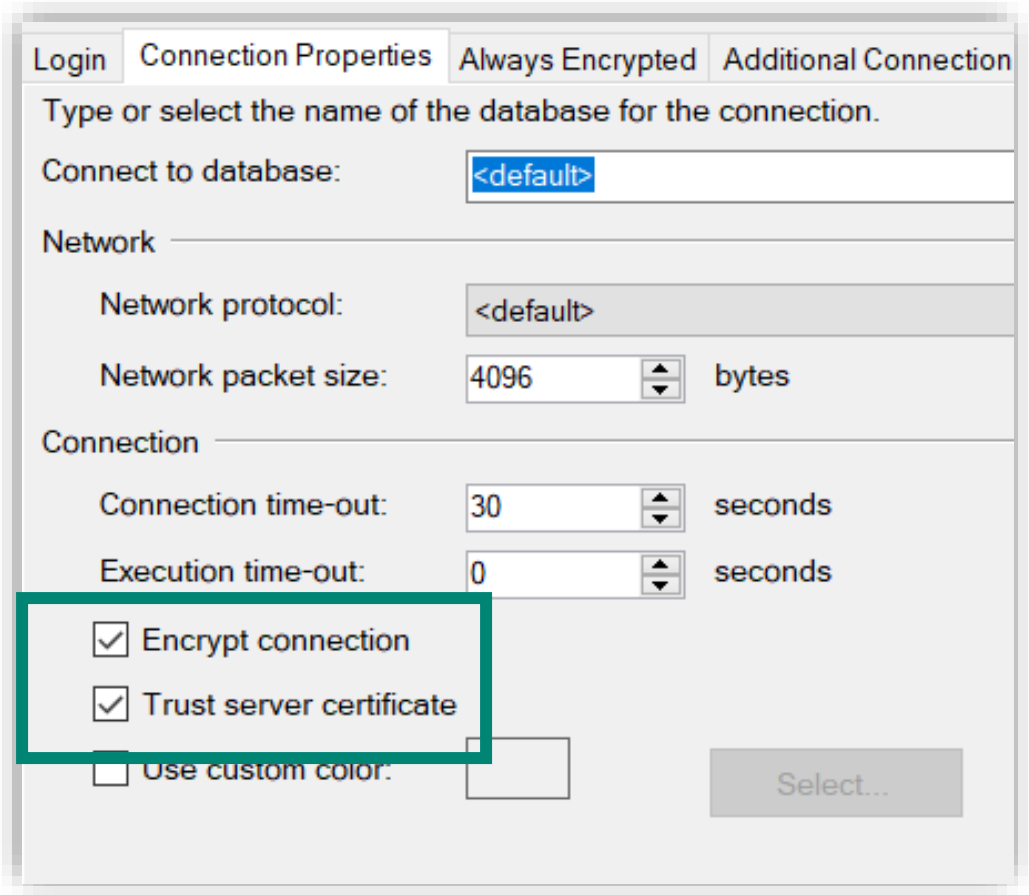


Enable Encryption from SSMS

Connect to **SQL Server Management Studio** and click the **Options** button.

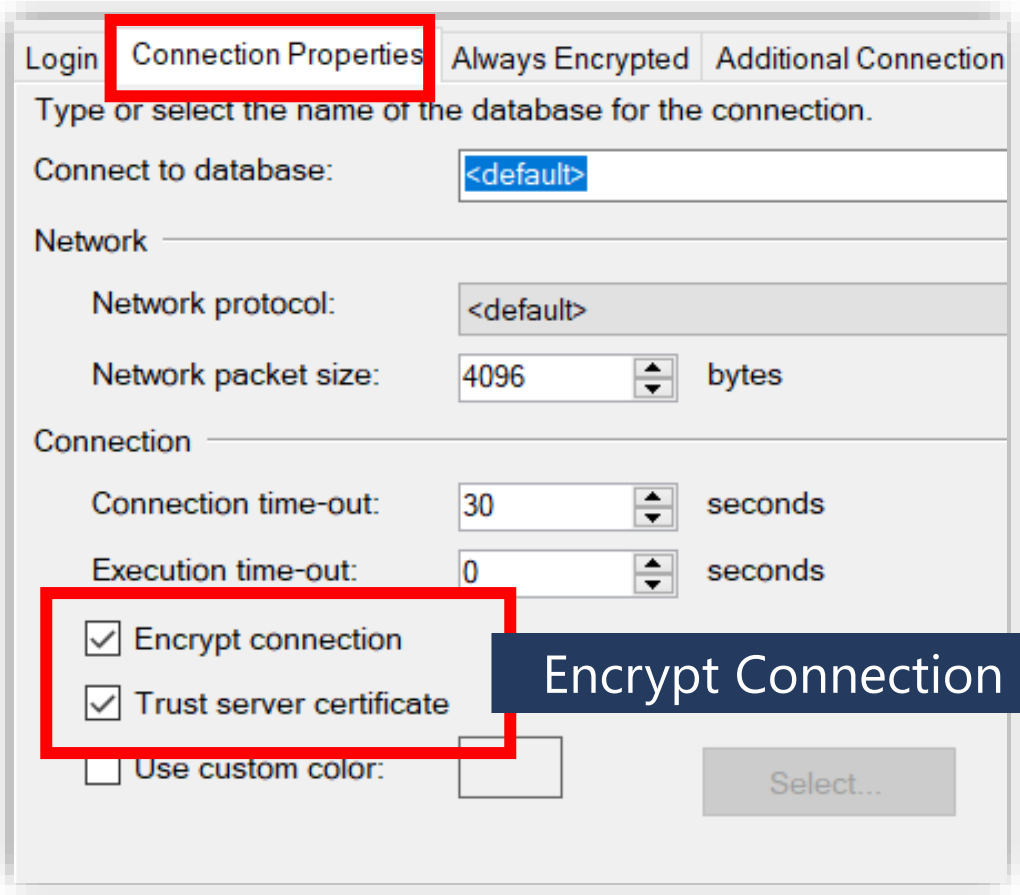
Select the **Connection Properties** tab.

Check the **Encrypt connection** and **Trust server certificate** checkboxes



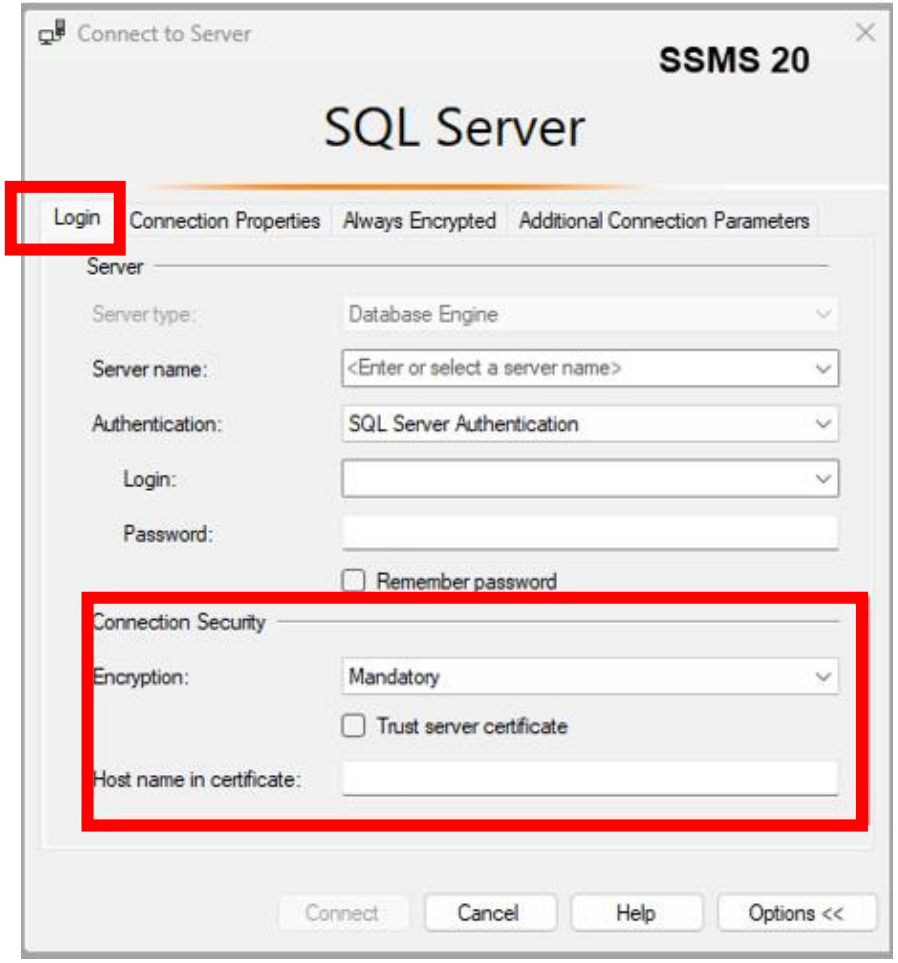
SSMS 19.3

SQL Server Management Studio 20



SSMS 19.3

Encrypt Connection Moved



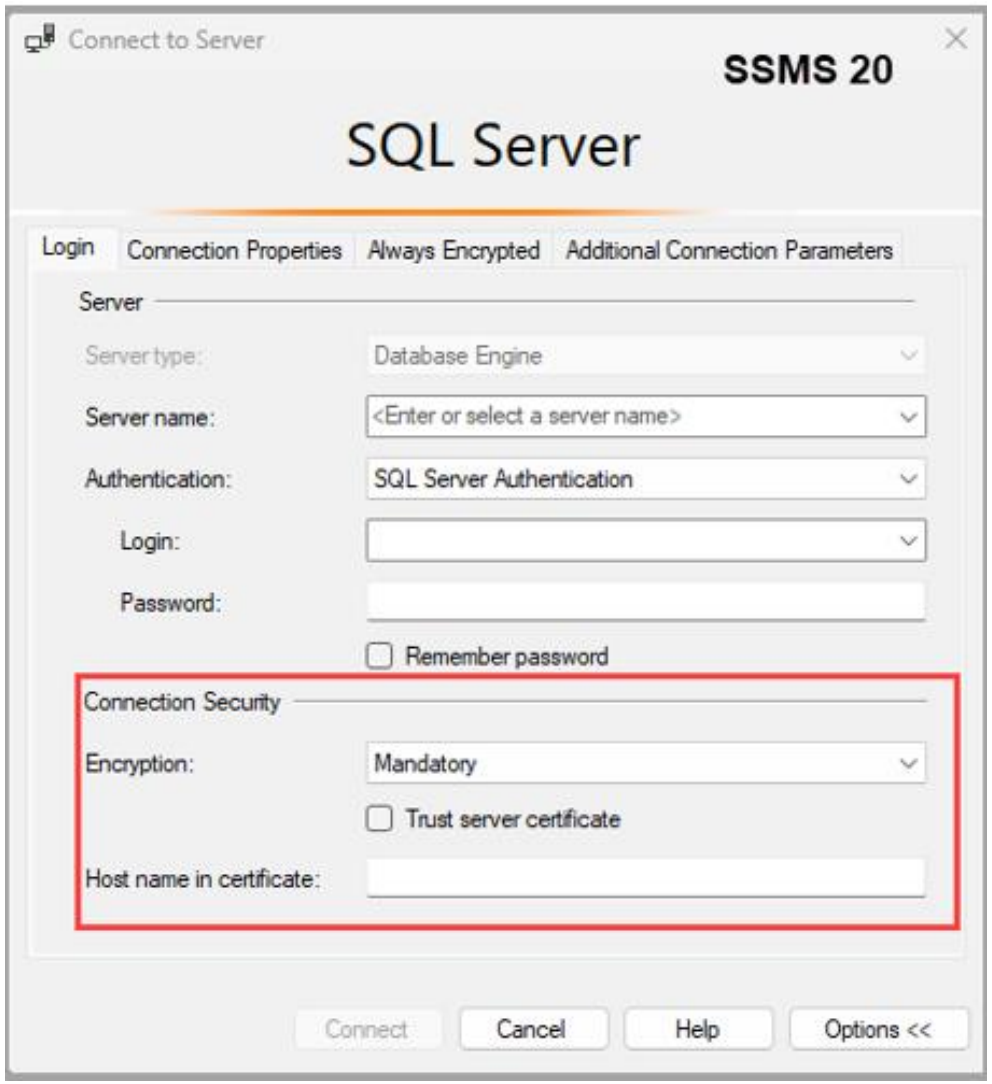
TLS 1.3 and Strict Encryption

SSMS 20 is the first major version of SSMS that supports Strict encryption and TLS 1.3, .

Strict encryption requires a trusted server certificate for encrypted connections

To use TLS 1.3, the client and the server must **both** support TLS 1.3 and have a trusted certificate installed.

Strict encryption is the most secure option for encryption in transit and is recommended for connections to Azure SQL Database and Azure SQL Managed Instance.



Demonstration

Encrypting Connections

- Install and configure the certificate for the SQL server instance.
- Test connection using Encrypt Connection option
- View the DMV to confirm the connections are encrypted.

Questions?

Knowledge Check

Why would you encrypt data in transit?

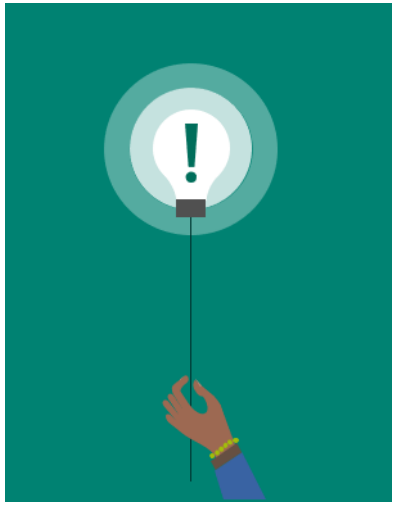
Are there any other encryption technologies that you might combine with Channel Encryption?

Lesson 4: Column Level Encryption

Objectives

After completing this learning, you will be able to:

- Understand how column level encryption works and when it could be a viable solution.



Column Level Encryption Overview

- Data is encryption both on the page and in memory
- Data needs to be encrypted and decrypted whenever it is called by an application
- Requires specific design considerations
- Also called Cell-Level Encryption

Why Use Column Level Encryption?

- Often used for compliance and regulatory reasons
- Provides additional layer of protection for sensitive data
- Helps protect against unauthorized access

Impact of Cell-Level Encryption

Application code change/database design considerations

Manual process

Index considerations

Potentially high-performance impact

Sample Code for Column Level Encryption

```
--Step 1
CREATE SYMMETRIC KEY key1 WITH ALGORITHM = AES_256
   ENCRYPTION BY PASSWORD = 'My Password!';

--Step 2
DECLARE @x varbinary(8000), @y varbinary(8000);
OPEN SYMMETRIC KEY key1
DECRYPTION BY PASSWORD = 'My Password!';
SET @x = EncryptByKey( key_guid( 'key1'), 'Test' );
SET @y = EncryptByKey( key_guid( 'key1'), 'Test' );
IF ( @x = @y )
    PRINT 'ERROR: EncryptByKey returned the same output twice!!!!';
ELSE
    PRINT 'EncryptByKey returns different results every time it is called';
CLOSE SYMMETRIC KEY key1;

--Step 3
DROP SYMMETRIC KEY key1;
```

Questions?

Knowledge Check

What is the difference between Column Level Encryption and Transparent Data Encryption (TDE)?

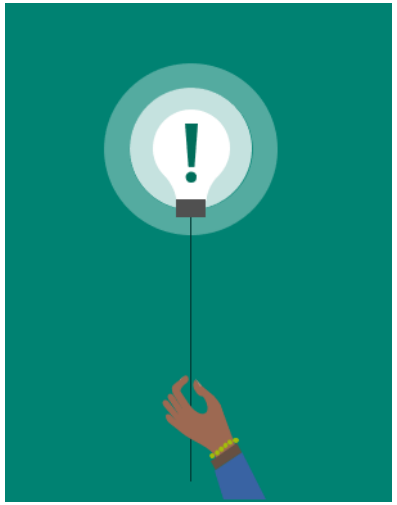
Is there any potential performance impact when you use Column Level Encryption?

Lesson 5: Overview of Always Encrypted

Objectives

After completing this learning, you will be able to:

- Understand the basic flow of data in Always Encrypted.
- Describe when data is encrypted and decrypted.



Always Encrypted - Typical Scenarios

Client and Data on-premises

Customer has client application and SQL Server, both running on-premises at business location

Client on-premises with data in Azure

Automatic encryption and decryption of sensitive data

Client and Data in Azure

Customer has client application hosted in Azure, which operates on sensitive data also stored in Azure

Always Encrypted - Usage

Medical professionals

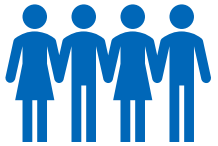
Hospitals

Private practices

Financial institutions

Banks

Credit unions



Social services

Employment

Family Info

Always Encrypted - Benefits

Allows customers to securely store sensitive data outside of their trust boundary while protecting data from highly privileged users.

Prevention of data disclosure

- Client-side encryption of sensitive data using keys that are never given to database system

Queries on encrypted data

- Support for equality comparison, including join, group by, and distinct operators

Application transparency

- Minimal application changes through server and client library enhancements

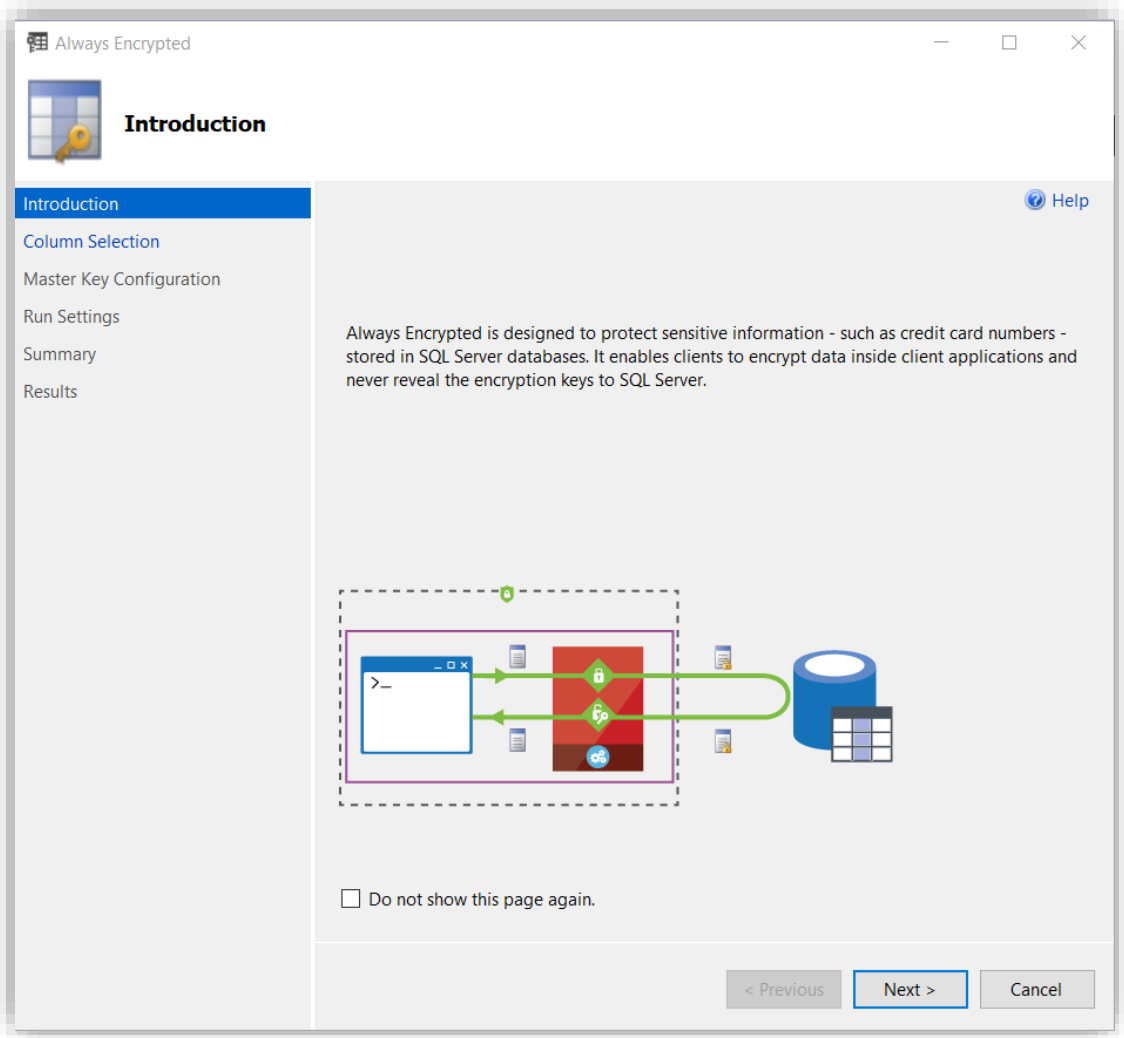
What is Always Encrypted?

Capability

- The ADO.NET client library provides transparent client-side encryption
- Microsoft SQL Server executes T-SQL queries on encrypted data

Benefits

- Sensitive data remains encrypted and can always be queried , on-premises and in the cloud
- Unauthorized users never have access to data or keys
- No application changes



Always Encrypted - Capabilities and Functions

Migration of sensitive data in application

- SQL Server only handles encrypted data—not plain text values

Automatic encryption and decryption of sensitive data

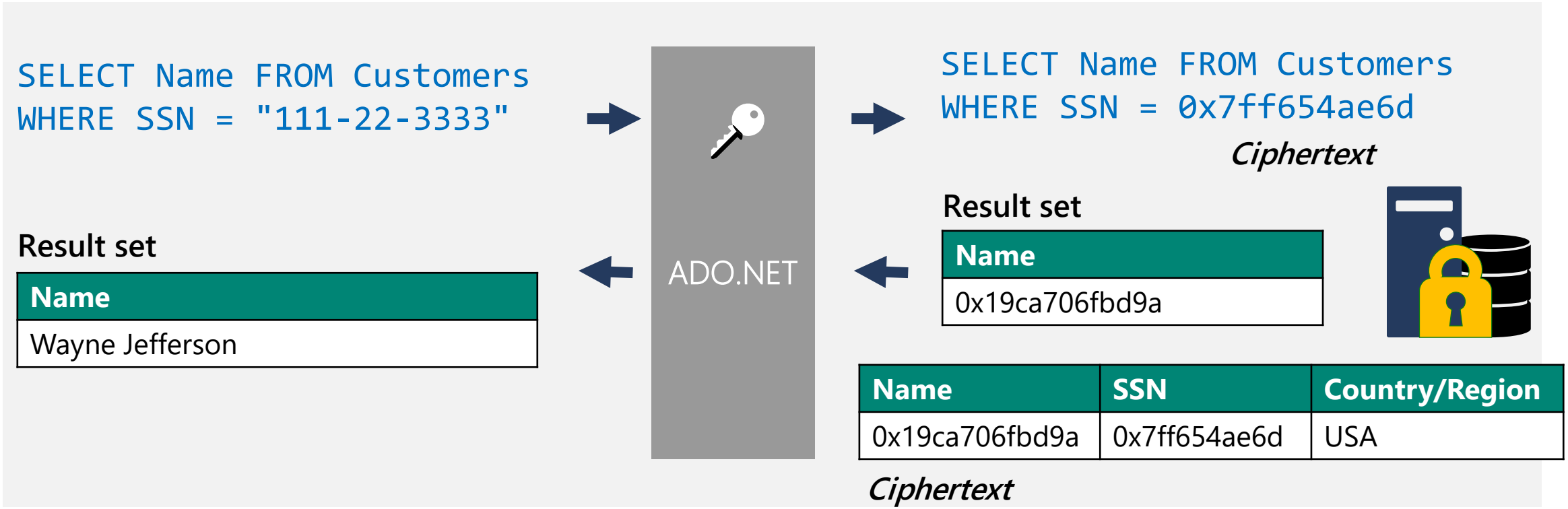
- Automatically rewrites queries to preserve semantics to application
- Driver transparently decrypts data

Bulk loading of encrypted data

- Use ALLOW_ENCRYPTED_VALUE_MODIFICATIONS option for bulk loading

How does Always Encrypted work?

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



Column Keys

CMK – Column Master Key is used to encrypt other keys, always in client's control, and in an external key store

- Azure Key Vault
- Windows Certificate Store
- Hardware Security Sections

CEK – Column Encryption Key is a content encryption key

Key provisioning

Encryption Types

Randomized

- Unpredictable results, more secure
- No support for equality searches, joins, grouping, or indexing
- Use for data that is returned, but not queried

Deterministic

- Predictable results, less secure
- Use for data that must be queried (equality support only)
- Easier to guess by examining encryption results
  - Increased risk for small value sets (True/False)

Encryption Methodologies

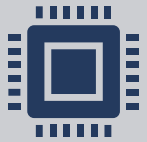
Randomized encryption

- Encrypt ('123-45-6789') = 0x0123A99C
- Repeat: Encrypt ('123-45-6789') = 0x01EB449B

Deterministic encryption

- Encrypt ('123-45-6789') = 0x17cfd50a
- Repeat: Encrypt ('123-45-6789') = 0x17cfd50a

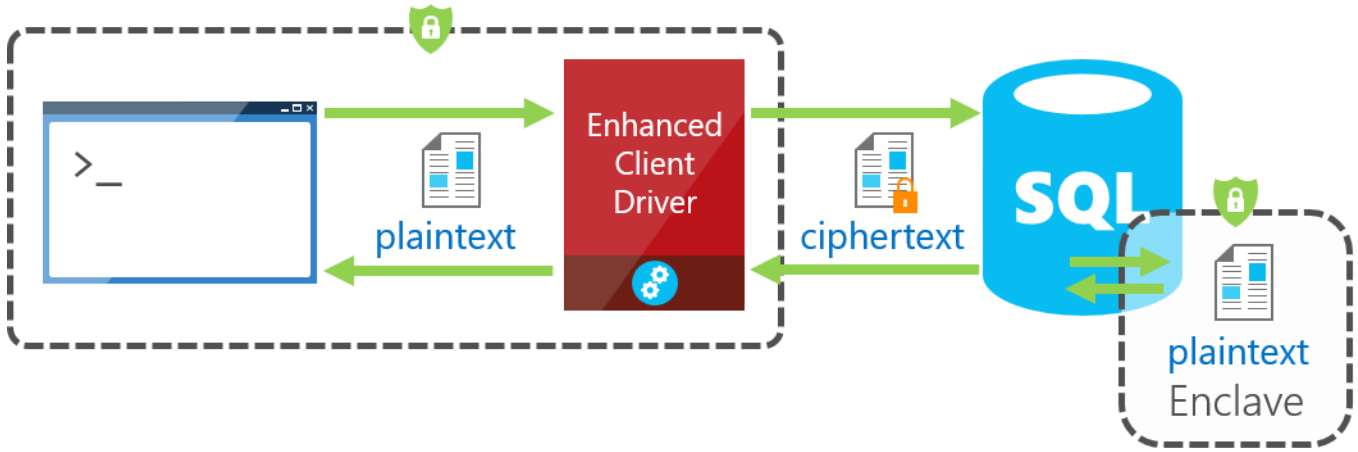
Data Encryption Algorithm



Always Encrypted uses the AEAD_AES_256_CBC_HMAC_SHA_256 algorithm to encrypt data in the database

Ciphertext length varies depends on the data type

Always Encrypted with Secure Enclaves



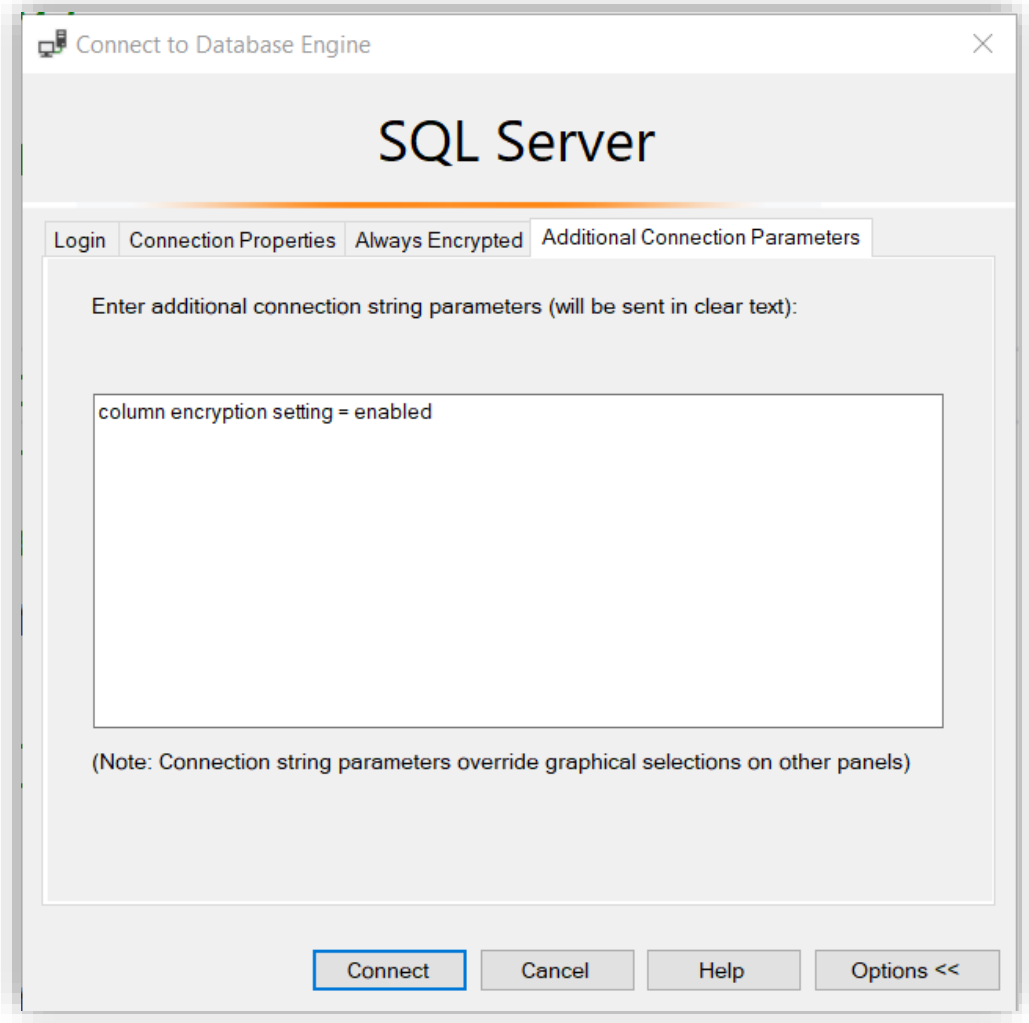
Always Encrypted Catalog Views

```
--Always Encrypted catalog views
SELECT * FROM sys.column_master_keys
SELECT * FROM sys.column_encryption_keys
SELECT * FROM sys.column_encryption_key_values
```

Find Always Encrypted Columns

```
--Find columns protected by Always Encrypted
SELECT c.name as E_Column, c.column_encryption_key_id,
   cek.name as E_Key, encryption_type_desc,
   encryption_algorithm_name
FROM sys.columns as c
JOIN sys.column_encryption_keys as cek
ON c.column_encryption_key_id =
   cek.column_encryption_key_id
WHERE c.column_encryption_key_id IS NOT NULL
```

Column Encryption Setting = Enabled



Permissions

ALTER ANY COLUMN MASTER KEY	• Required to create and delete a column master key
ALTER ANY COLUMN ENCRYPTION KEY	• Required to create and delete a column encryption key
VIEW ANY COLUMN MASTER KEY DEFINITION	• Required to access and read column master key metadata objects while managing keys or querying encrypting columns
VIEW ANY COLUMN ENCRYTPION KEY DEFINITION	• Required to access and read column encryption key metadata objects while managing keys or querying encrypting columns

Permissions Scenario

Creating, changing, or reviewing key metadata in the database

SCENARIO	ALTER ANY COLUMN MASTER KEY	ALTER ANY COLUMN ENCRYPTION KEY	VIEW ANY COLUMN MASTER KEY DEFINITION	VIEW ANY COLUMN ENCRYPTION KEY DEFINITION
Key management	X	X	X	X
Querying encrypted columns			X	X

Limitations

Not supported when columns use any of these datatypes

- xml, hierarchyid, rowversion, image, text, ntext, geography, geometry, user-defined types, or sql_variant

Clauses that cannot be used for encrypted columns

- FOR XML
- FOR JSON PATH

Features that do not work on encrypted columns

- Transactional or merge replication
- Distributed queries (linked servers)

Always Encrypted Best Practices

In Azure, complete isolation of data from cloud administrators is only provided when client tier is running on-premises.

If the client tier is running in the cloud, moving the encryption/decryption routine to the client tier still leaves data and keys exposed to cloud administrators.

Test application workload to ensure that restrictions and limitation do not affect your application

Carefully evaluate randomized vs deterministic encryption

Questions?

Knowledge Check

What are the two types of encryption that can be used with always encrypted?

What are some of the differences between the two?

Where does the encryption and decryption occur?

Always Encrypted

Configure Always Encrypted and query encrypted data

Questions?

Knowledge Check

Explain how the column master key and column encryption key work together

What are some important considerations when using always encrypted?