

A Developer's Guide to Protecting Your Data Inside of SQL Server

David Berry

@DavidCBerry13

<https://github.com/DavidCBerry13/sql-server-security/>



Security is a **Team** Sport

Plan for a **defense in depth**

You **cannot** rely on perimeter defenses

In 2018, the **perimeter is everywhere**

What We Will Talk About

Users, Schemas and Roles

Connection Strings

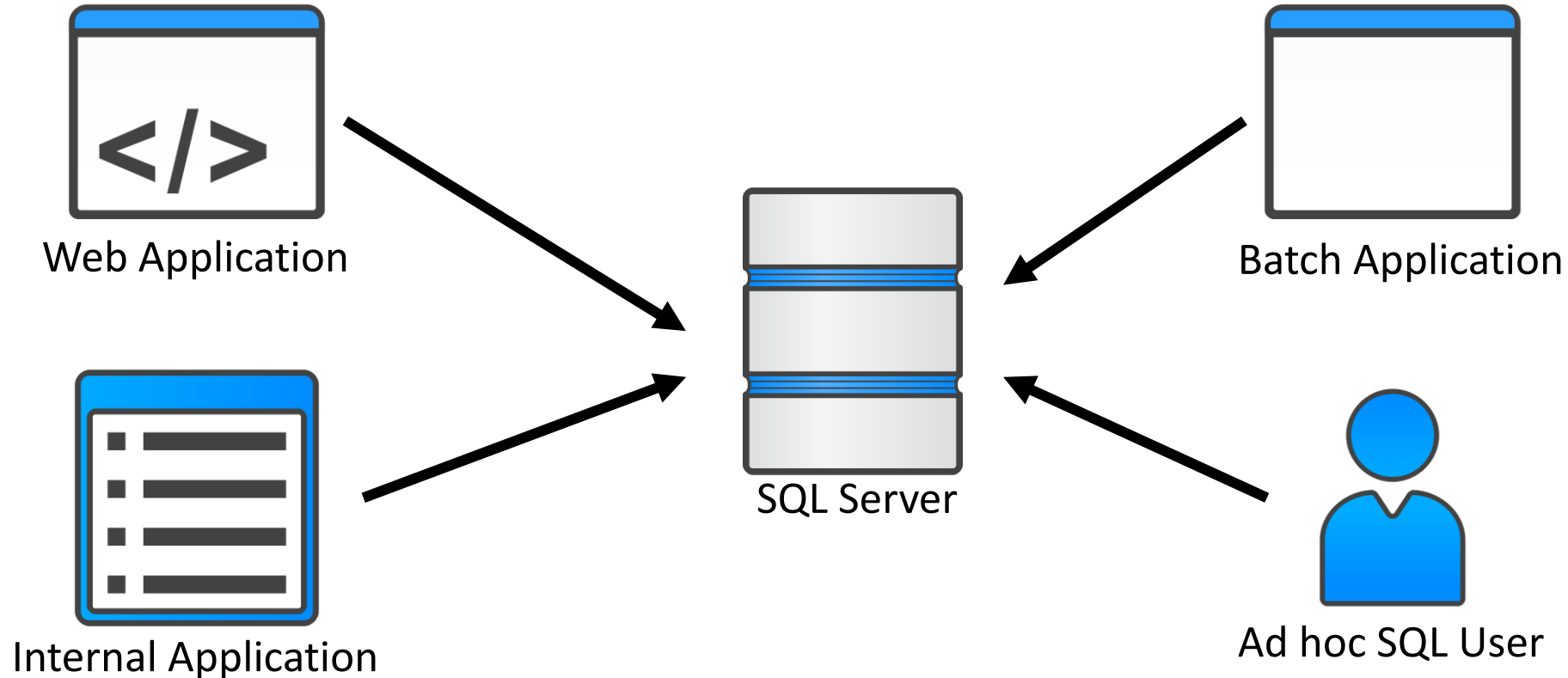
Transport Layer Security

Encrypting Data

SQL Injection

Users, Schemas and Roles

Overused and Overprivileged Users



Problem is when all these users use the same account

Assigning proper users, roles and privileges is your first line of defense

Secure Database User Practices

Use Windows
Authentication

Have a separate
service account
for each
application

Use AD groups to
For user access

Have a separate
dedicated account
to push schema
changes

Do not assign
db_owner to any
application or user
account

Use database roles
to assign
permissions

Why Use Windows Authentication



Eliminates an additional password



Take advantage of existing processes

- Especially around employee transfers and termination

db_owner Role

Fixed-Database Roles

The following table shows the fixed-database roles and their capabilities. These roles exist in all databases. Except for the **public** database role, the permissions assigned to the fixed-database roles cannot be changed.

Fixed-Database role name	Description
db_owner	Members of the db_owner fixed database role can perform all configuration and maintenance activities on the database, and can also drop the database in SQL Server. (In SQL Database and SQL Data Warehouse, some maintenance activities require server-level permissions and cannot be performed by db_owners .)

<https://docs.microsoft.com/en-us/sql/relational-databases/security/authentication-access/database-level-roles?view=sql-server-2017>

What's Wrong with the db_owner Role?

Do you really want an application user to be able to create or drop tables?

How about modify your stored procedures?

Or grant and revoke permissions from other users?

What Is a Database Role

A collection of privileges. Roles help you manage permissions in a database so that you can assign a group or privileges together

Creating Database Roles

```
CREATE ROLE Operations;
```

```
GRANT SELECT ON Customers TO Operations;
```

```
GRANT SELECT ON Addresses TO Operations;
```

```
GRANT SELECT ON CustomerAddresses TO Operations;
```

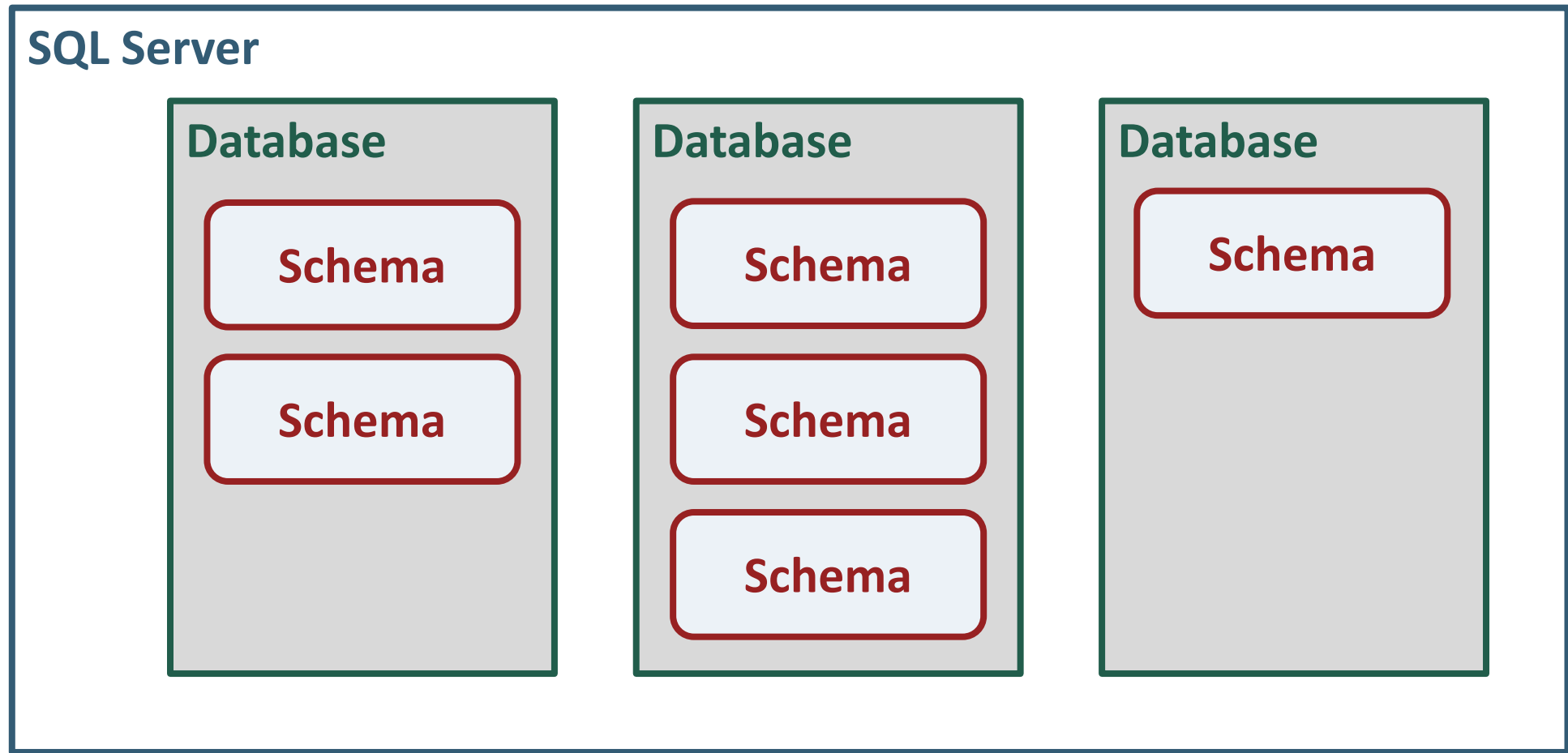
```
GRANT SELECT ON Orders TO Operations;
```

```
GRANT SELECT ON OrderItems TO Operations;
```

```
ALTER ROLE Operations
```

```
    ADD MEMBER [MyDomain\OperationsTeam];
```

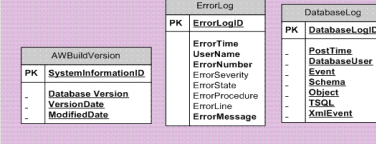
Schemas



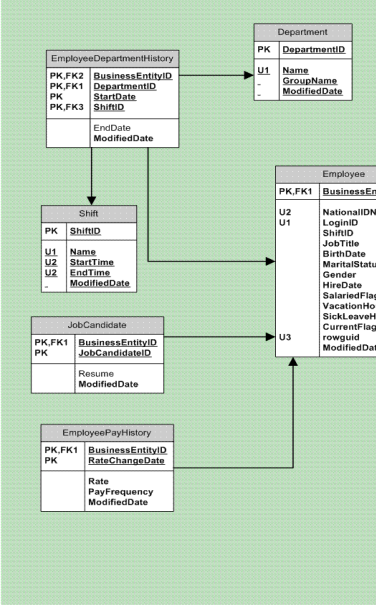
Best Print Results if:
11X17 paper
Landscape
Fit to 1 sheet

Samples and Sample Databases at
<http://CodePlex.com/SqlServerSamples>

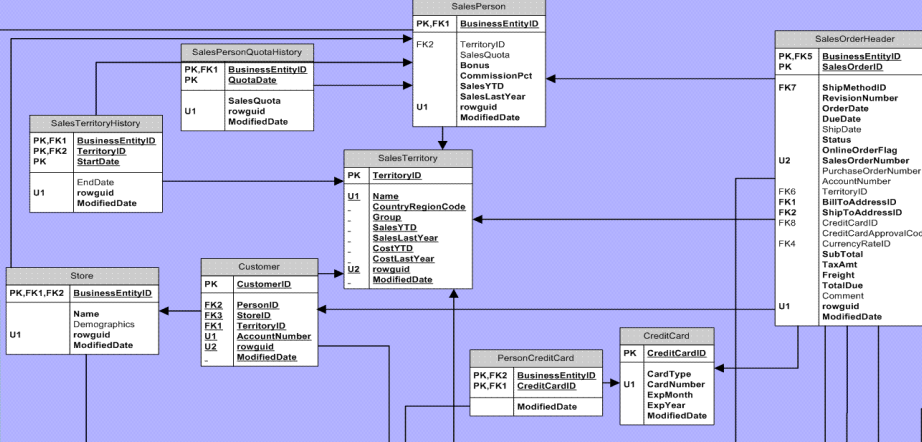
dbo



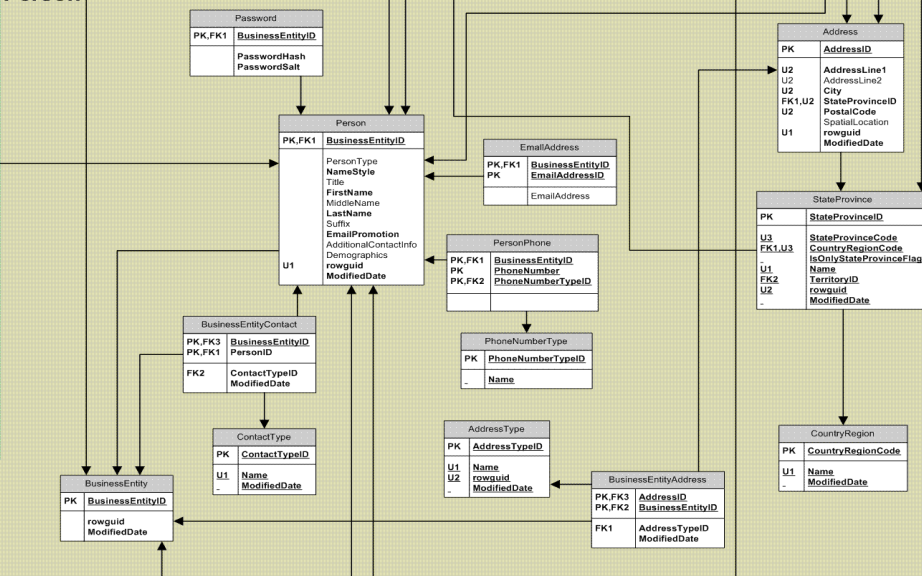
HumanResources



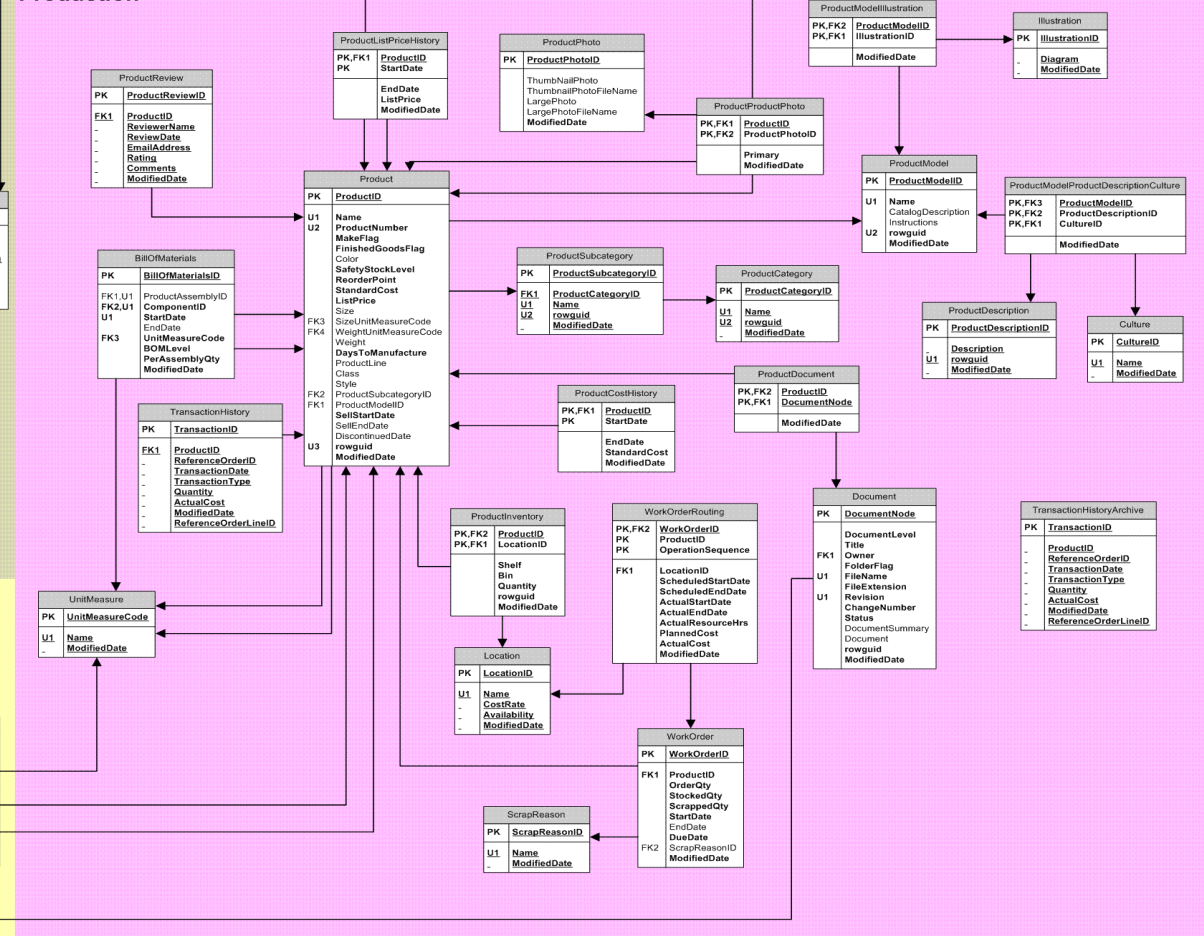
Sales



Person



Production



Schemas

Sales

Purchasing

Person

Production

HumanResources

dbo

Other Access Control Mechanisms

Views

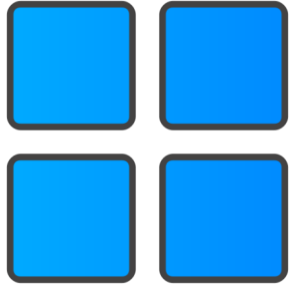
- Limit what columns or what rows can be seen
- Views can be inserted and updated in many cases
- ORM support is still a challenge

Stored Procedures

- Specify exactly what a user can do
- Use to eliminate ad-hoc data access

Connection Strings

Securing Your Connection String



Use Windows Authentication

Eliminates the password
in your connection string



Never store connection strings in code

These can be easily
reverse engineered



Use Key Vault For SQL Logins

If you must use SQL
logins, store the
password somewhere
secure

Transport Layer Security

Little Known Facts

Traditional SQL Server

The connection between your application and SQL Server is ***not encrypted*** by default

SQL Azure

The connection between your application and SQL Server ***is encrypted*** by default

What Does Transport Layer Security Give Us

Encrypts our data in motion

Verifies who we are talking to



Connection Strings to Encrypt Data

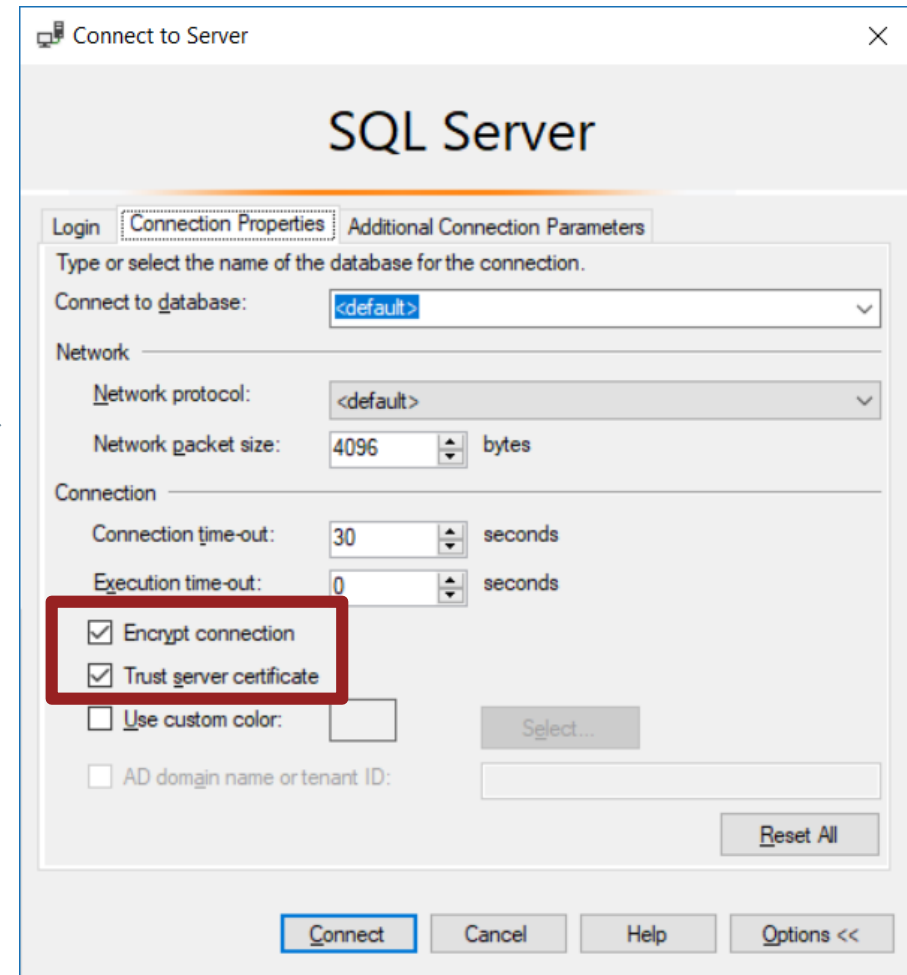
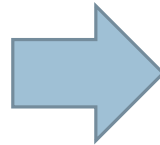
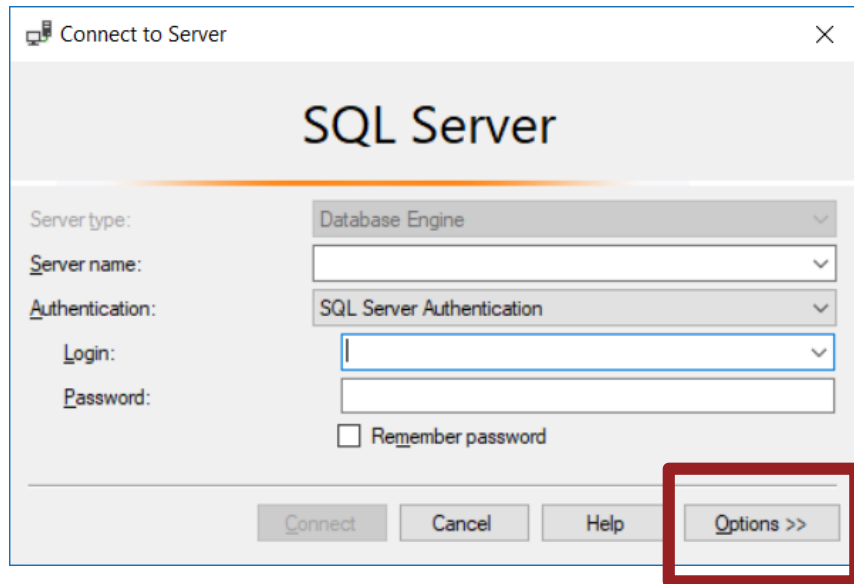
Encrypt Data and Allow a Self Signed Certificate

```
Server=<server>\<instance>;Database=<database>;Integrated  
Security=true;Encrypt=true;TrustServerCertificate=True
```

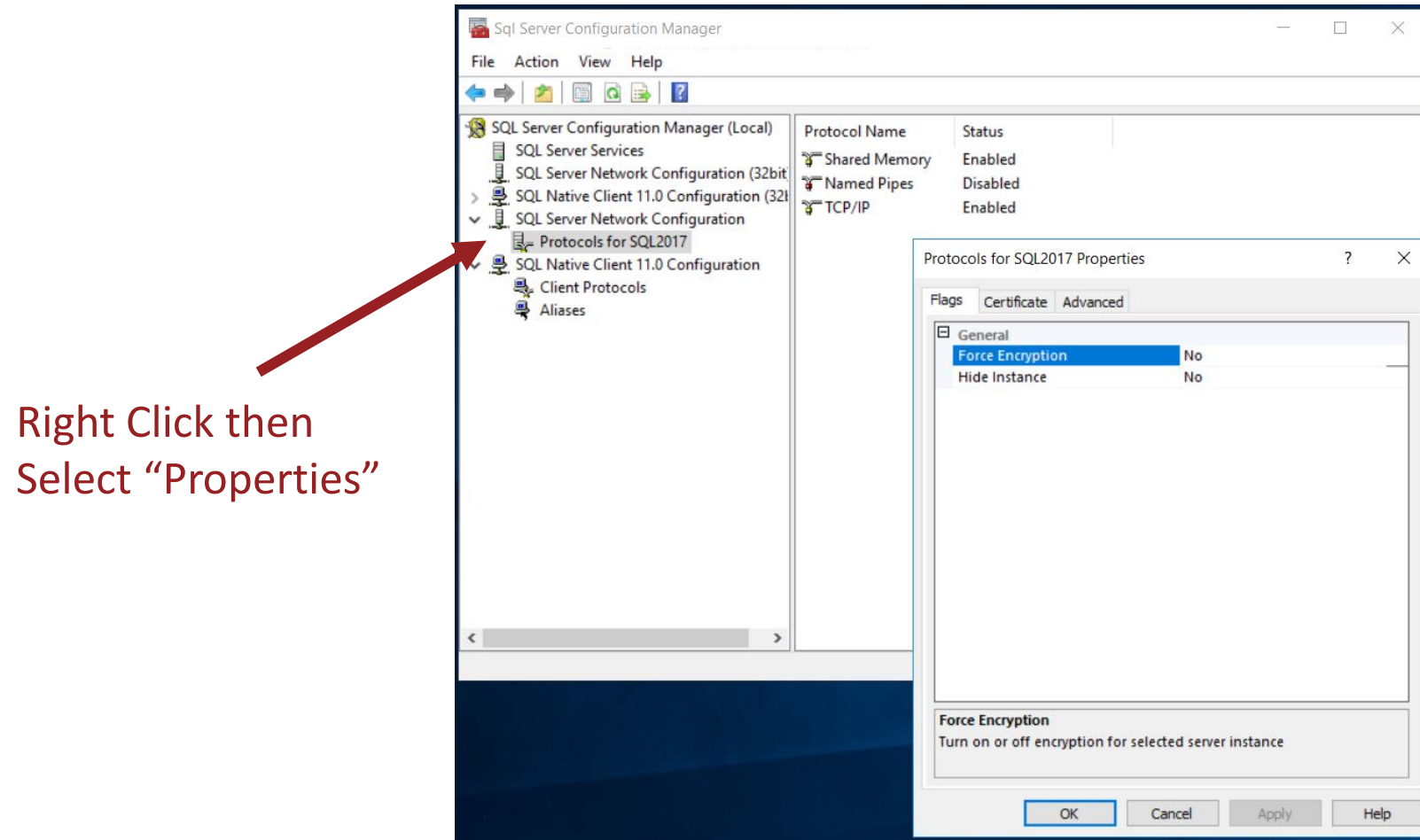
Encrypt Data and Validate Server Certificate

```
Server=<server>\<instance>;Database=<database>;Integrated  
Security=true;Encrypt=true;TrustServerCertificate=false
```

Decrypting Data from SSMS



Configuring SQL Server to Force TLS



Full instructions at: <https://www.sqlshack.com/how-to-set-and-use-encrypted-sql-server-connections/>

TLS Strategy

Set Encrypt=true
for you connection
strings

Work towards
installing a
verifiable
certificate

Discuss requiring
encrypted
connections with
your DBA team

Database Encryption

Transparent Data Encryption

Encrypts the data files of SQL Server (data at rest)

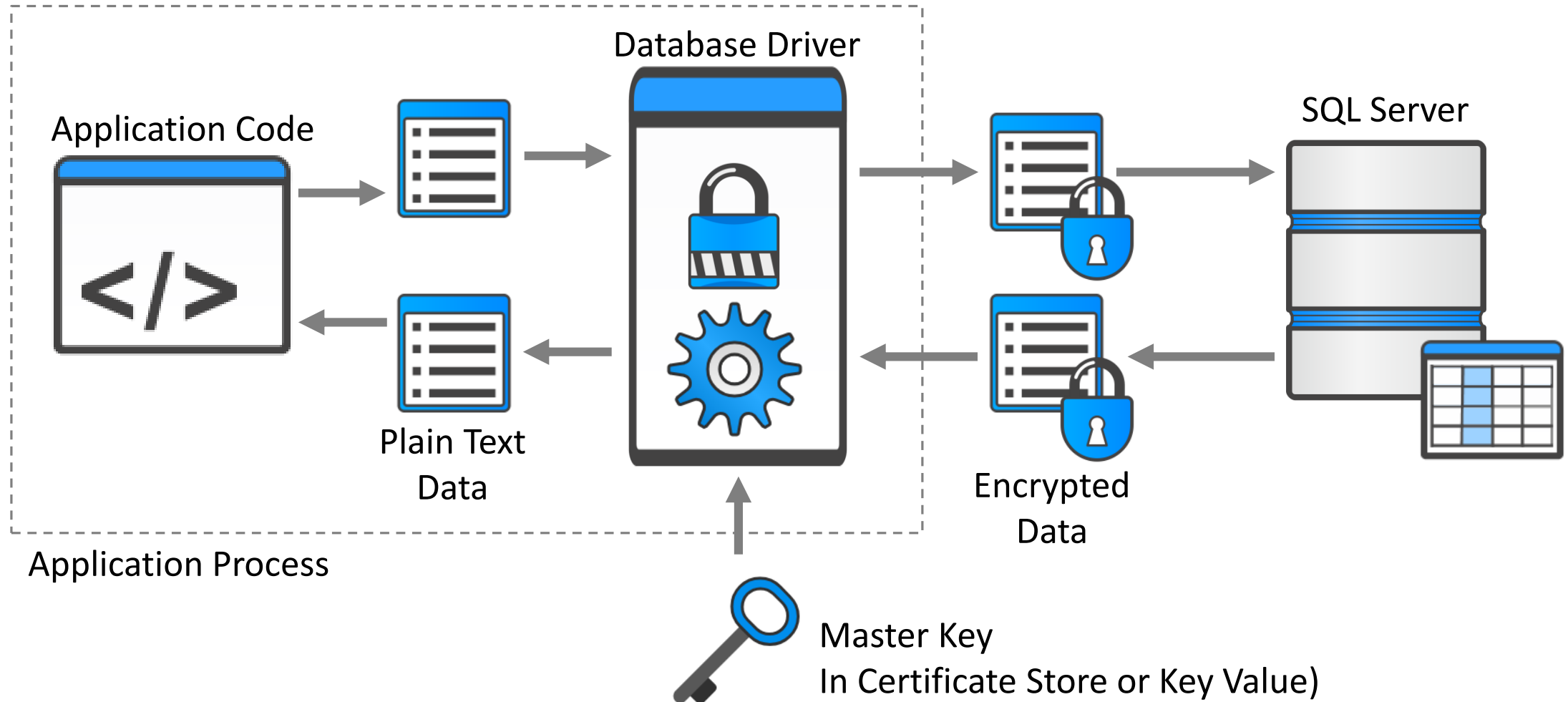
Protects you in case an attacker gets your data files or backups

Always Encrypted

Designed to protect data at a column level

Available in SQL Server 2016 and later

How Always Encrypted Works



What You Can/Cannot Encrypt

Can Be Encrypted

- VARCHAR/NVARCHAR
- INT/SMALLINT/BIGINT
- FLOAT/DOUBLE
- DECIMAL
- MONEY/SMALLMONEY
- DATETIME/DATE/TIME

Cannot Be Encrypted

- XML
- SQL_VARIANT
- IMAGE
- GEOGRAPHY
- GEOMETRY

Encryption Types

Deterministic

- Always generates the same encrypted value for a plain text value
- Column can be used in equality comparisons, joins, group by operations and indexes
- Not suitable for cardinality columns because the attacker can guess values based on probabilities

Randomized

- Different encrypted values will be generated for the same plain text input
- Column cannot be used in equality comparisons, joins, group by operations or indexes

Deterministic Encryption Example

```
SELECT CustomerId, FirstName, LastName, ZipCode, ZipCodeEncrypted  
FROM CustomerNames  
WHERE ZipCode = '78501';
```

100 %

Results Messages

	CustomerId	FirstName	LastName	ZipCode	ZipCodeEncrypted
1	200	Charles	Williams	78501	0x0134AEA12DA64A5219742363864E57335138CB56B2D3380EE73F27194A19E8AB3506...
2	327	John	Krumm	78501	0x0134AEA12DA64A5219742363864E57335138CB56B2D3380EE73F27194A19E8AB3506...
3	390	Curtis	Motz	78501	0x0134AEA12DA64A5219742363864E57335138CB56B2D3380EE73F27194A19E8AB3506...
4	396	Christian	Sanchez	78501	0x0134AEA12DA64A5219742363864E57335138CB56B2D3380EE73F27194A19E8AB3506...
5	1358	Edna	Sparks	78501	0x0134AEA12DA64A5219742363864E57335138CB56B2D3380EE73F27194A19E8AB3506...
6	1494	Nicolas	Ramsey	78501	0x0134AEA12DA64A5219742363864E57335138CB56B2D3380EE73F27194A19E8AB3506...

Connection Strings For Always Encrypted

Encrypt Data and Allow a Self Signed Certificate

```
Server=<server>\<instance>;Database=<database>;Integrated  
Security=true;Encrypt=true;TrustServerCertificate=True;  
Column Encryption Setting=enabled
```

Using Azure Key Vault Provider

```
// https://docs.microsoft.com/en-us/azure/sql-database/sql-database-always-encrypted-azure-key-vault
// Required NuGet Packages
//     Microsoft.SqlServer.Management.AlwaysEncrypted.AzureKeyVaultProvider
//     Microsoft.IdentityModel.Clients.ActiveDirectory

private static ClientCredential _clientCredential;

static void InitializeAzureKeyVaultProvider()
{
    _clientCredential = new ClientCredential(applicationId, clientKey);

    SqlColumnEncryptionAzureKeyVaultProvider azureKeyVaultProvider =
        new SqlColumnEncryptionAzureKeyVaultProvider(GetToken);

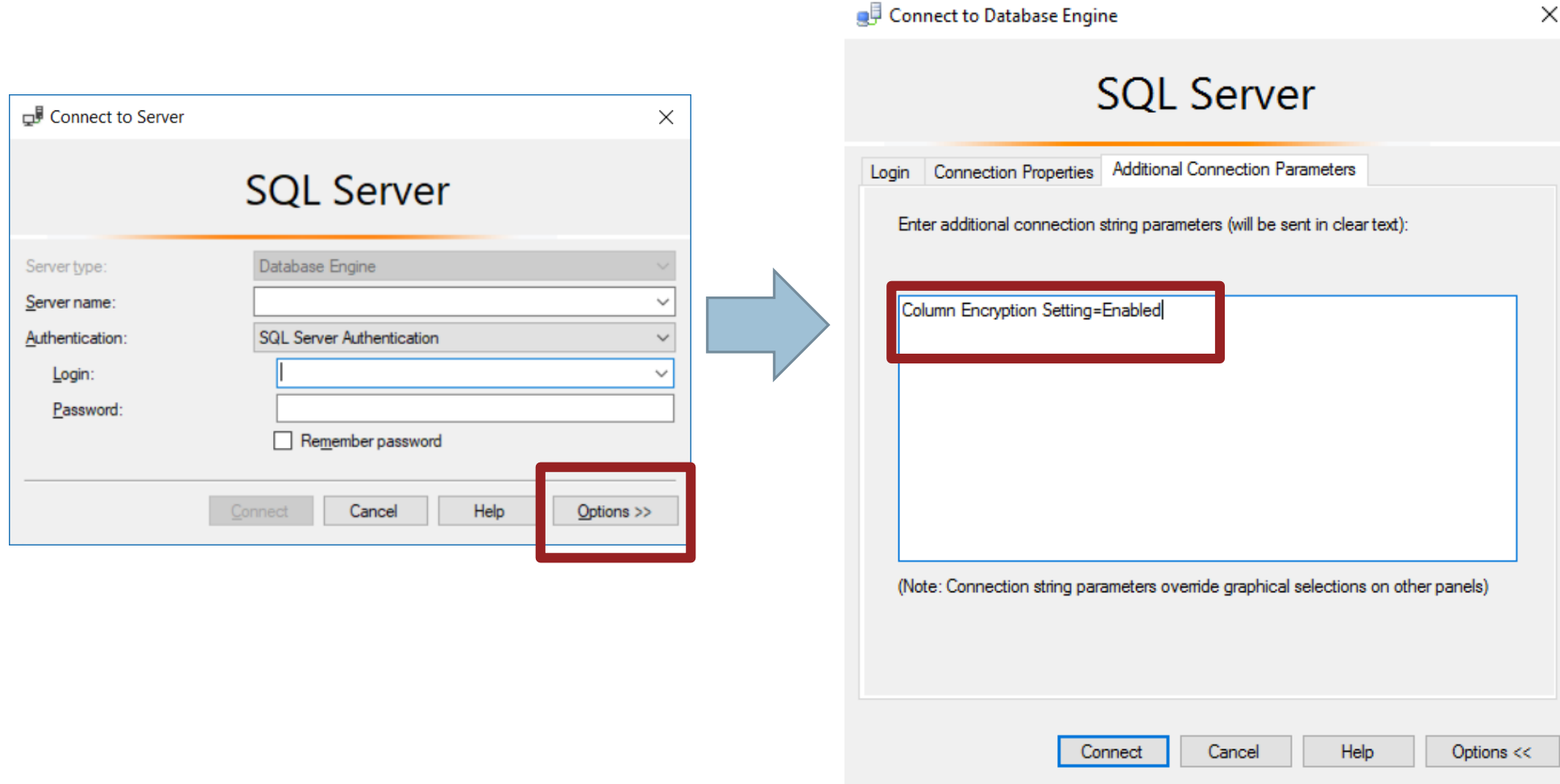
    Dictionary<string, SqlColumnEncryptionKeyStoreProvider> providers =
        new Dictionary<string, SqlColumnEncryptionKeyStoreProvider>();

    providers.Add(SqlColumnEncryptionAzureKeyVaultProvider.ProviderName, azureKeyVaultProvider);
    SqlConnection.RegisterColumnEncryptionKeyStoreProviders(providers);
}

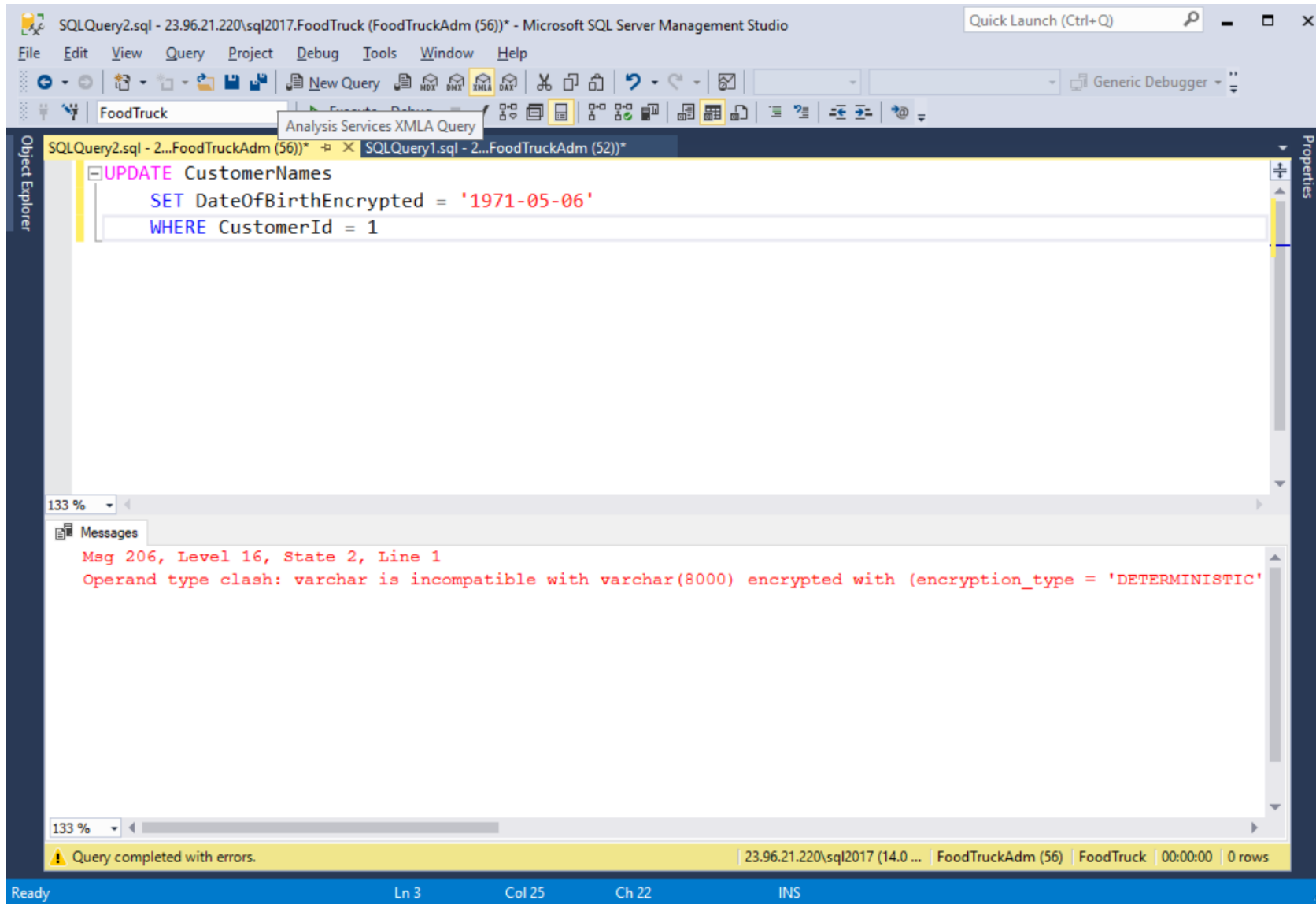
public async static Task<string> GetToken(string authority, string resource, string scope)
{
    var authContext = new AuthenticationContext(authority);
    AuthenticationResult result = await authContext.AcquireTokenAsync(resource, _clientCredential);

    if (result == null)
        throw new InvalidOperationException("Failed to obtain the access token");
    return result.AccessToken;
}
```

Encrypting Connection from SSMS



SSMS Functionality Changes



Editing Encrypted Data in SSMS

Microsoft SQL Server Management Studio window showing a query result for the CustomerNames table.

Query Text:

```
SELECT CustomerId, FirstName, MiddleInitial, LastName, StreetAddress, City, State, ZipCode, EmailAddress, Telephone, DateOfBirth, Ssn, ZipCodeEncrypted, DateOfBirthEncrypted, SsnEncrypted
FROM CustomerNames
WHERE (CustomerId = 587)
```

Result Set:

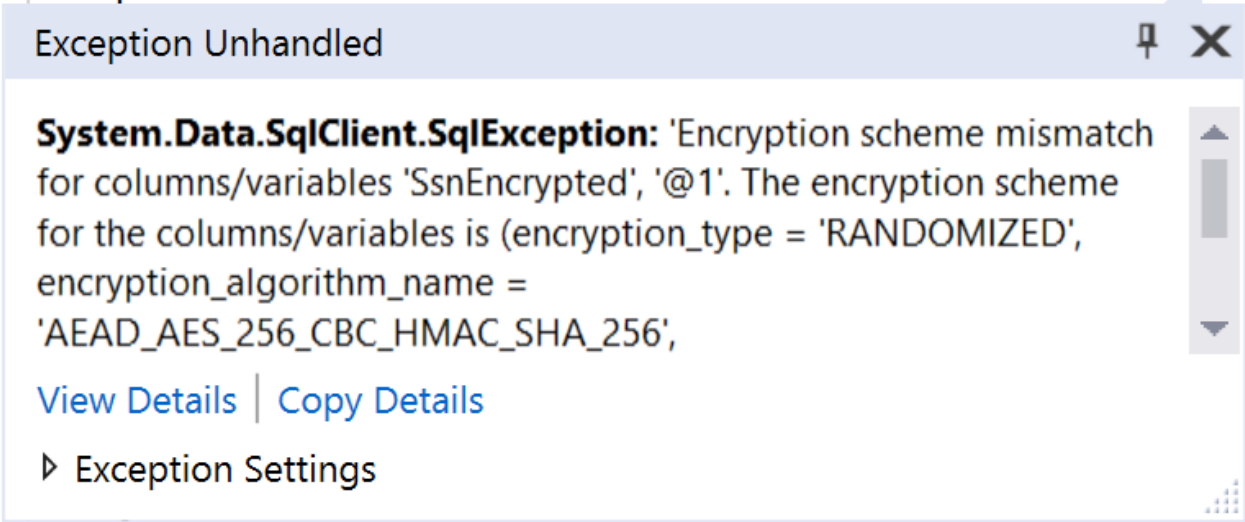
	CustomerId	FirstName	MiddleInitial	LastName	StreetAddress	City	State	ZipCode	EmailAddress	Telephone
▶	587	Margaret	E	Cherry	2486 Chenowet...	Spring Hill	TN	37174	MargaretECherr...	931-489-7239
•	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Properties window is visible on the right side of the window.

Status bar: Ready

What Happens When Searching Randomized Data?

```
con.Open();
using (SqlCommand cmd = new SqlCommand("SELECT FirstName, LastName, ZipCode
{
    using (SqlDataReader dr = cmd.ExecuteReader())
    {
        while (dr.Read())
        {
            Console.WriteLine($"{dr.GetString(0)} {dr.GetString(1)} {dr.GetString(2)}");
        }
    }
}
```



The dialog box displays the following error message:

System.Data.SqlClient.SqlException: 'Encryption scheme mismatch for columns/variables 'SsnEncrypted', '@1'. The encryption scheme for the columns/variables is (encryption_type = 'RANDOMIZED', encryption_algorithm_name = 'AEAD_AES_256_CBC_HMAC_SHA_256',

Below the message are two links: [View Details](#) and [Copy Details](#).

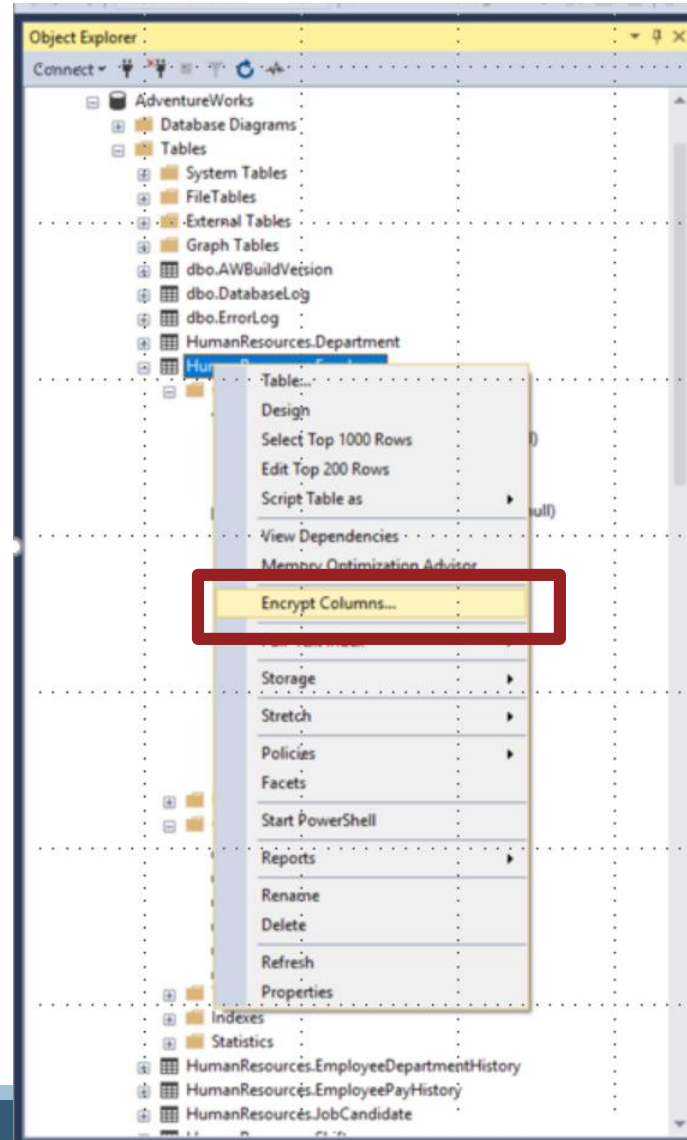
At the bottom left of the dialog is a link: [Exception Settings](#).

To the right of the dialog, the code continues with:

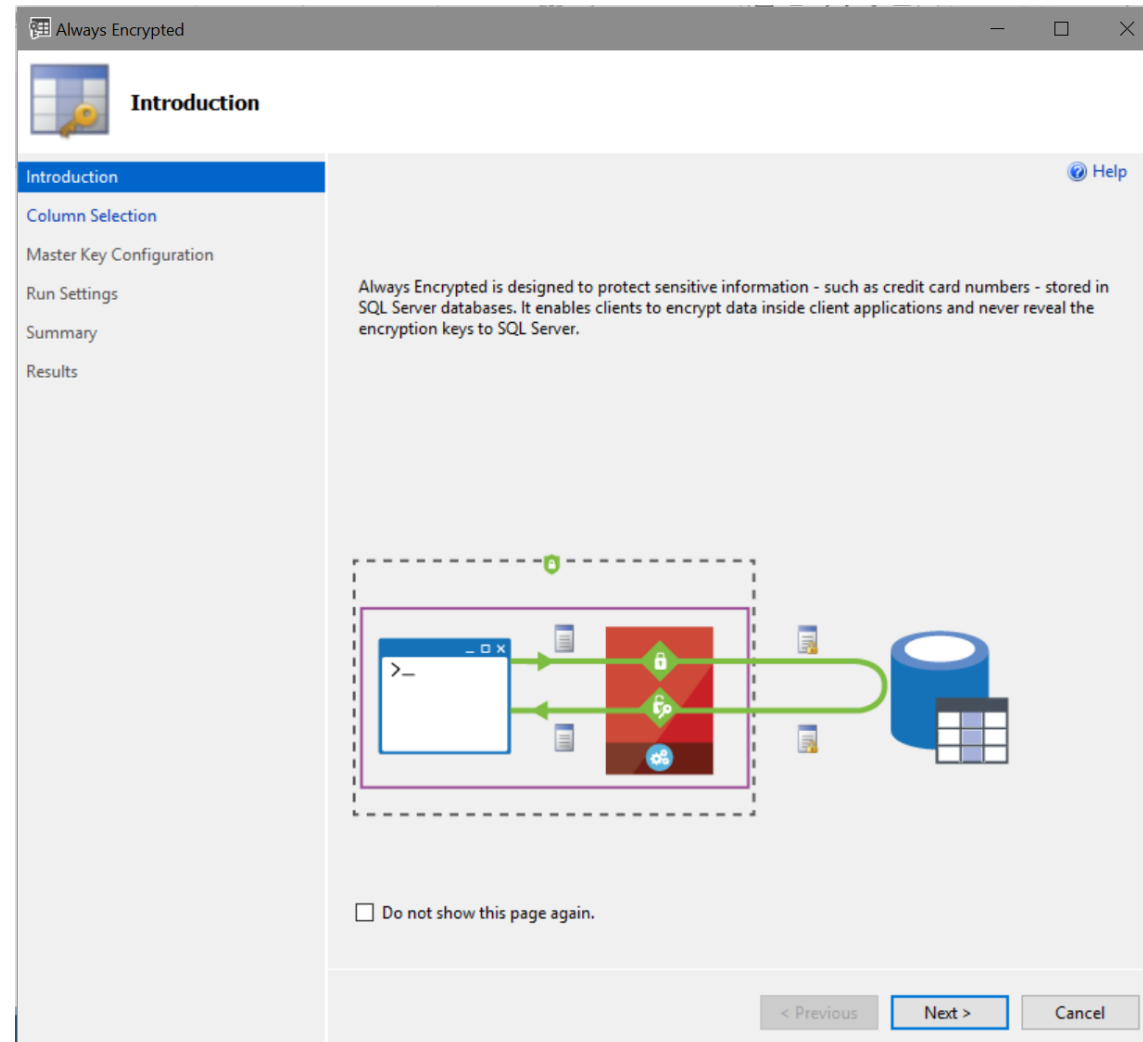
```
    }
}
```

{zipCode} {c

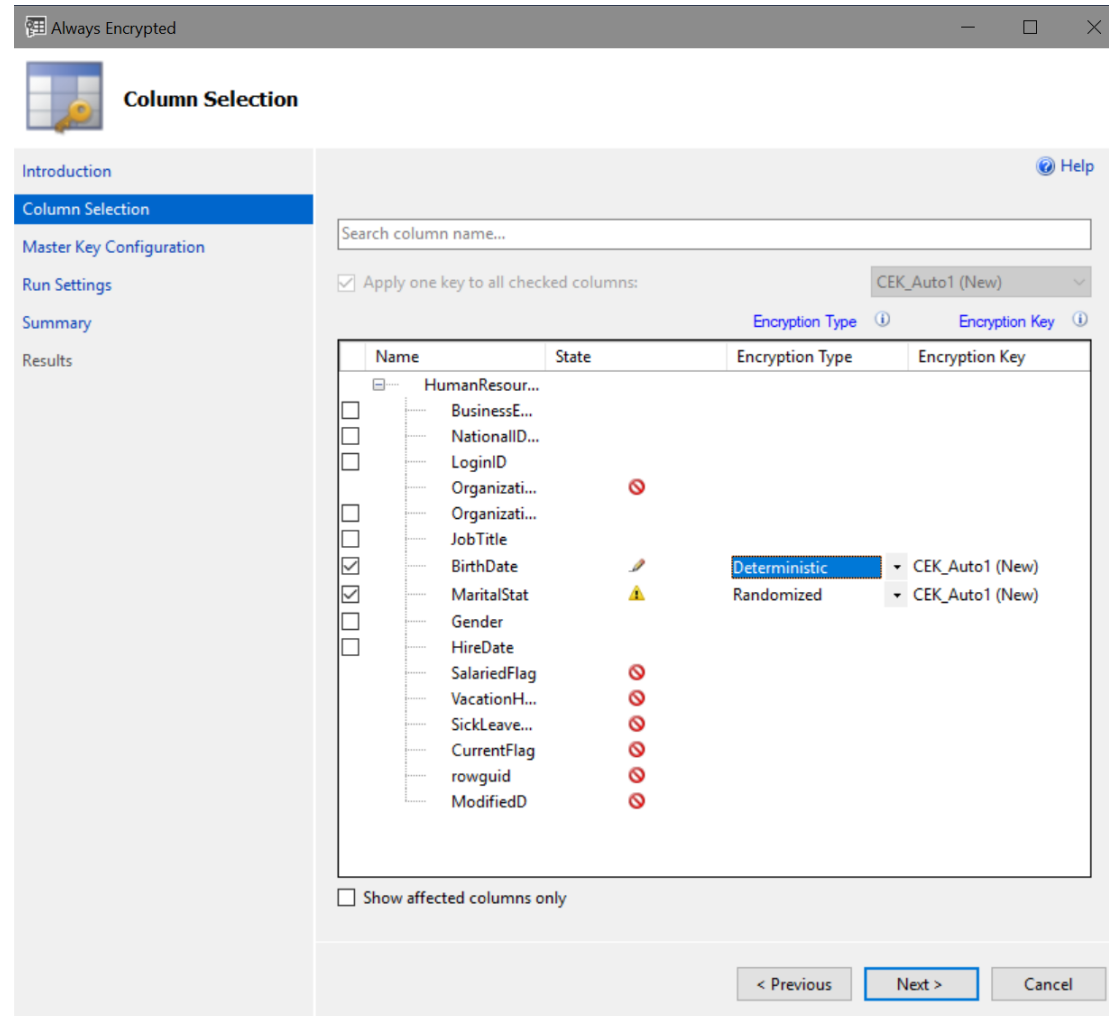
Always Encrypted Wizard



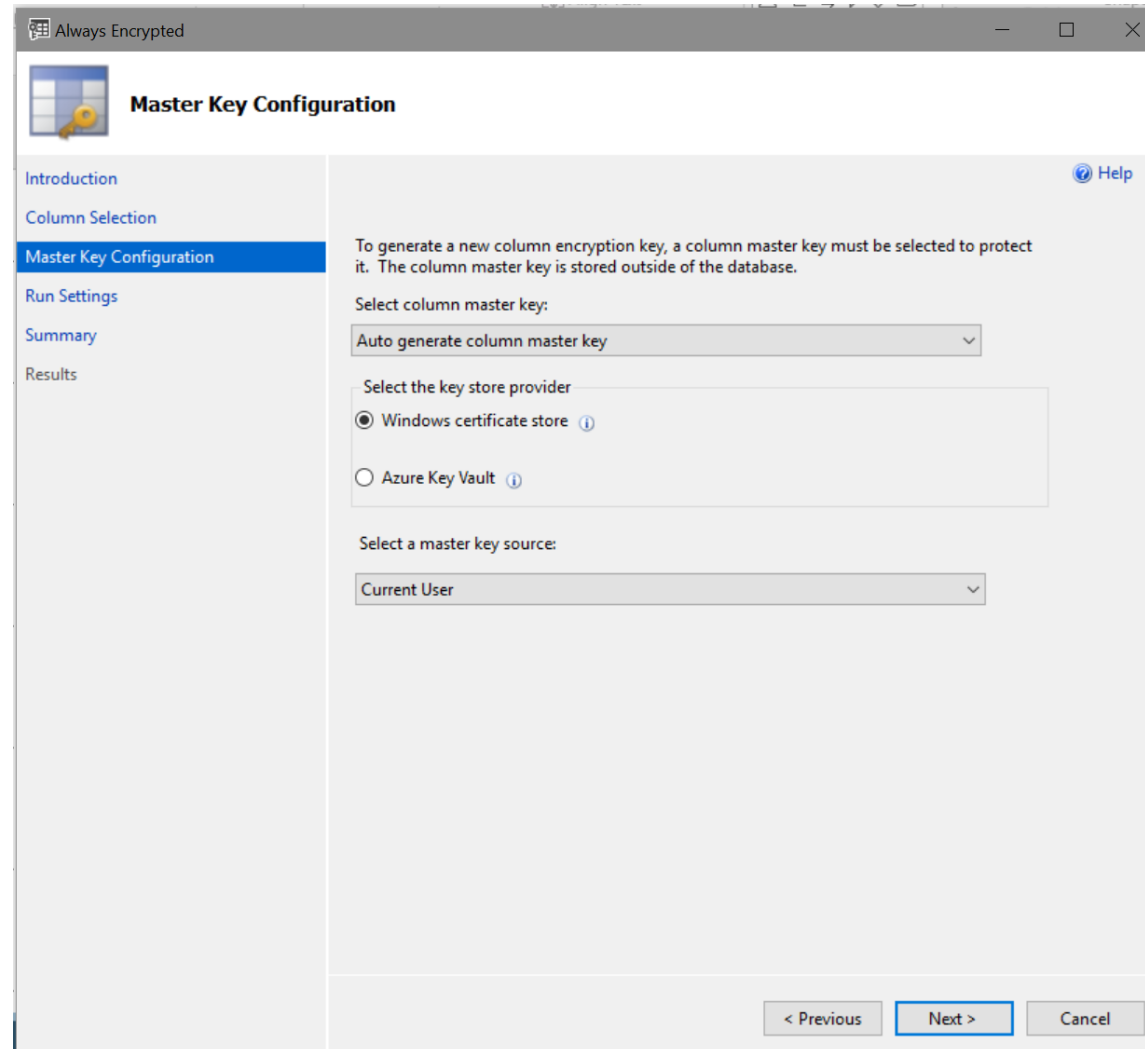
Always Encrypted Wizard



Selecting Columns to be Encrypted



Creating a Master Encryption Key



The screenshot shows the 'Master Key Configuration' window in SQL Server Enterprise Manager. The window title is 'Always Encrypted'. The left sidebar contains a list of steps: 'Introduction', 'Column Selection', 'Master Key Configuration' (which is highlighted), 'Run Settings', 'Summary', and 'Results'. The main area of the window is titled 'Master Key Configuration' and contains the following text and controls:

To generate a new column encryption key, a column master key must be selected to protect it. The column master key is stored outside of the database.

Select column master key:

Auto generate column master key

Select the key store provider:

☒ Windows certificate store ⓘ

☐ Azure Key Vault ⓘ

Select a master key source:

Current User

At the bottom right, there are three buttons: '< Previous', 'Next >', and 'Cancel'.

Always Encrypted Columns DDL

```
CREATE TABLE [HumanResources].[Employee]
(
    [BusinessEntityID] [int] NOT NULL,
    [NationalIDNumber] [nvarchar](15) NOT NULL,
    [LoginID] [nvarchar](256) NOT NULL,
    [OrganizationNode] [hierarchyid] NULL,
    [OrganizationLevel] AS ([OrganizationNode].[GetLevel]()),
    [JobTitle] [nvarchar](50) NOT NULL,
    [BirthDate] [date] ENCRYPTED WITH (COLUMN_ENCRYPTION_KEY = [CEK_Auto1],
        ENCRYPTION_TYPE = Deterministic, ALGORITHM = 'AEAD_AES_256_CBC_HMAC_SHA_256') NOT NULL,
    [MaritalStatus] [nchar](1) NOT NULL,
    [Gender] [nchar](1) COLLATE Latin1_General_BIN2 ENCRYPTED WITH
        (COLUMN_ENCRYPTION_KEY = [CEK_Auto1],
        ENCRYPTION_TYPE = Randomized, ALGORITHM = 'AEAD_AES_256_CBC_HMAC_SHA_256') NOT NULL,
    [HireDate] [date] NOT NULL,
    [SalariedFlag] [dbo].[Flag] NOT NULL,
    CONSTRAINT [PK_Employee_BusinessEntityID]
        PRIMARY KEY (BusinessEntityID)
)
GO
```

Always Encrypted Search Implications

- To be searched, columns must use deterministic encryption
- You can only search on the full value (no range searches or LIKE clauses)
- Consider storing a subset of the field to be searched (last 4 of social)

Always Encrypted Documentation

Microsoft Documentation

<https://docs.microsoft.com/en-us/sql/relational-databases/security/encryption/always-encrypted-database-engine>

Simple Talk Article

<https://www.red-gate.com/simple-talk/sql/database-administration/sql-server-encryption-always-encrypted/>

Simple Demographics Often Identify People Uniquely

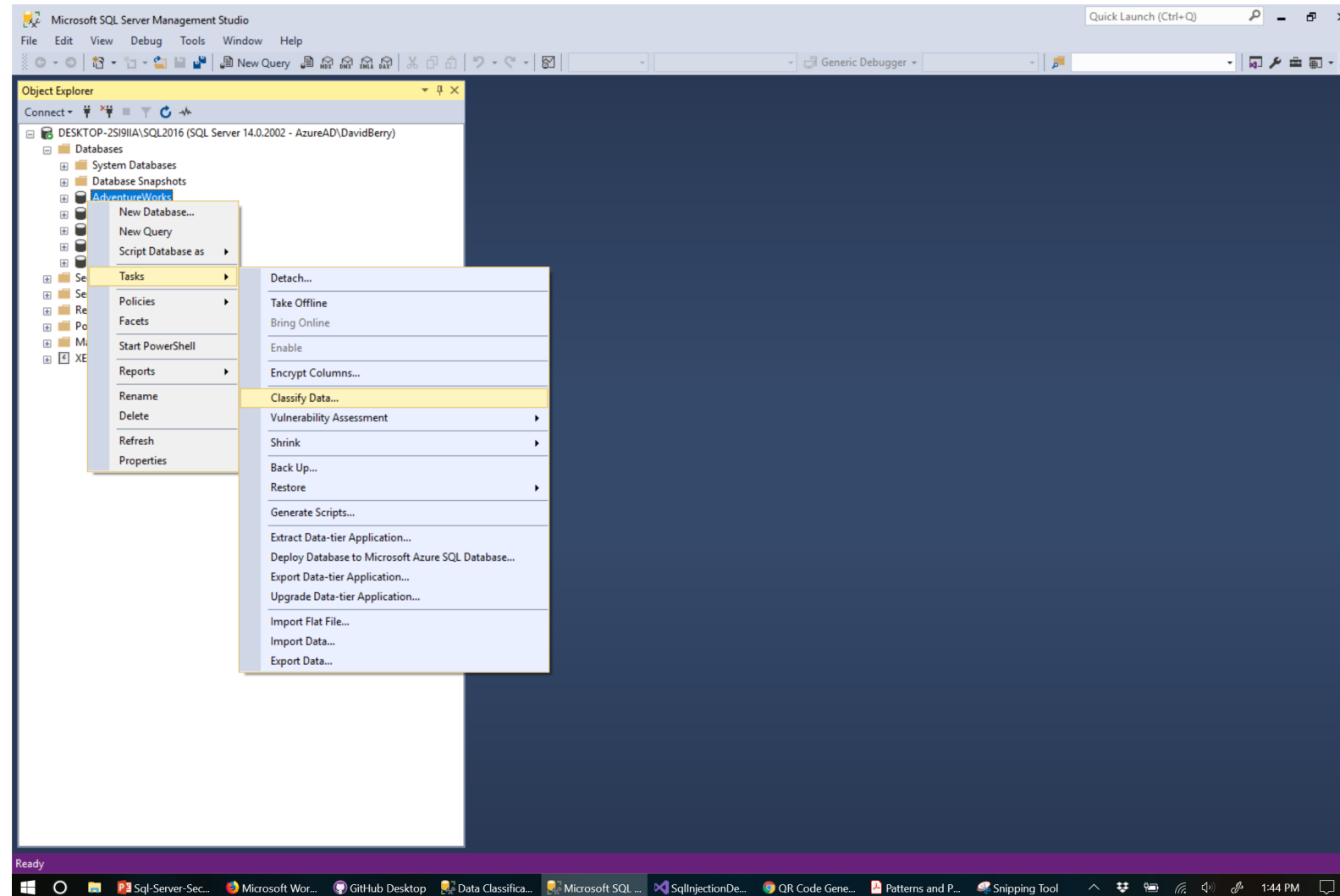
L. Sweeney, Simple Demographics Often Identify People Uniquely. Carnegie Mellon University, Data Privacy Working Paper 3. Pittsburgh 2000.

Simple Demographics Often Identify People Uniquely


Latanya Sweeney
Carnegie Mellon University
latanya@andrew.cmu.edu

<https://dataprivacylab.org/projects/identifiability/paper1.pdf>

SQL Server Data Classification



Data Classification in SQL Server

 39 columns with classification recommendations (click to minimize)

0 classified columns [Learn more – Getting Started Gui](#)

Schema	Table	Column	Information Type	Sensitivity Label
--------	-------	--------	------------------	-------------------

39 columns with classification recommendations (click to minimize)

Accept selected recommendations

<input type="checkbox"/>	Schema	Table	Column	Information Type	Sensitivity Label
<input type="checkbox"/>	dbo	ErrorLog	UserName	Credentials	Confidential
<input type="checkbox"/>	HumanResources	Employee	NationalIDNumber	National ID	Confidential - GDPR
<input type="checkbox"/>	Person	Address	AddressLine1	Contact Info	Confidential - GDPR
<input type="checkbox"/>	Person	Address	AddressLine2	Contact Info	Confidential - GDPR
<input type="checkbox"/>	Person	Address	City	Contact Info	Confidential - GDPR
<input type="checkbox"/>	Person	Address	PostalCode	Contact Info	Confidential - GDPR
<input type="checkbox"/>	Person	EmailAddress	EmailAddress	Contact Info	Confidential - GDPR
<input type="checkbox"/>	Person	Password	PasswordHash	Credentials	Confidential
<input type="checkbox"/>	Person	Password	PasswordSalt	Credentials	Confidential
<input type="checkbox"/>	Person	Person	FirstName	Name	Confidential - GDPR

SQL Injection

Injection attacks are still ranked #1 on the OWASP top ten list of security vulnerabilities

But we are actually seeing SQL Injection decline

Vulnerable Code

```
String sql = @"SELECT CustomerId, FirstName, LastName, Ssn
                FROM Customers WHERE CustomerId = '" + customerId + "'";

using (SqlCommand cmd = new SqlCommand(sql, dbConnection))
{
    using (SqlDataReader reader = cmd.ExecuteReader())
    {
        if (reader.Read())
        {
            customer = new Customer()
            {
                CustomerId = reader.GetString(0),
                FirstName = reader.GetString(1),
                LastName = reader.GetString(2),
                Ssn = reader.GetString(3)
            };
        }
    }
}
```

Safe Code

```
String sql = @"SELECT CustomerId, FirstName, LastName, Ssn
                FROM Customers WHERE CustomerId = @customerId";

using (SqlCommand cmd = new SqlCommand(sql, dbConnection))
{
    cmd.Parameters.Add("@customerId", customerId);
    using (SqlDataReader reader = cmd.ExecuteReader())
    {
        if (reader.Read())
        {
            customer = new Customer()
            {
                CustomerId = reader.GetString(0),
                FirstName = reader.GetString(1),
                LastName = reader.GetString(2),
                Ssn = reader.GetString(3)
            };
        }
    }
}
```

Defeating SQL Injection

Parameterize Your SQL

Don't try to escape your own SQL strings. There are too many scenarios for us to cover

Use an ORM

Modern ORMs automatically parameterize SQL statements for you

Scan Your Code

Code analysis in Visual Studio or tools like Sonarqube and Puma Scan can detect vulnerable SQL

Final Thoughts

Thank You

David Berry

@DavidCBerry13

<https://github.com/DavidCBerry13/sql-server-security/>

