# T-SQL Project: Data Encryption & Security

This **T-SQL project** implements **data encryption** for **sensitive information** like Social Security Numbers (SSNs) and credit card numbers. It uses **SQL Server's encryption functions** and provides a stored procedure for **authorized decryption**.

**1. Directory Structure**

Organize SQL scripts into a structured format:

DataEncryptionSecurity/

│

├── Tables/

│   ├── Create\_Customers\_Table.sql

│

├── Security/

│   ├── Encrypt\_Data.sql

│   ├── Decrypt\_Data.sql

│   ├── Security\_Policy.sql

│

├── Procedures/

│   ├── sp\_EncryptSensitiveData.sql

│   ├── sp\_DecryptSensitiveData.sql

│

├── Scripts/

│   ├── Setup\_Encryption.sql

│

└── README.md

**2. SQL Components**

**A. Creating the Customers Table with Encrypted Columns**

The **Customers table** stores sensitive data using SQL Server's **ENCRYPTION BY PASSWORD** feature.

CREATE TABLE Customers (

    CustomerID INT IDENTITY(1,1) PRIMARY KEY,

    Name NVARCHAR(100) NOT NULL,

    SSN VARBINARY(MAX) NOT NULL,

    CreditCardNumber VARBINARY(MAX) NOT NULL

);

**B. Encrypt Data Using a Secure Password**

This script encrypts **SSNs and Credit Card Numbers** when inserting records.

DECLARE @EncryptionKey NVARCHAR(50) = 'SecureKey2025!';

INSERT INTO Customers (Name, SSN, CreditCardNumber)

VALUES

    ('John Doe', ENCRYPTBYPASSPHRASE(@EncryptionKey, '123-45-6789'), ENCRYPTBYPASSPHRASE(@EncryptionKey, '4111-1111-1111-1111')),

    ('Jane Smith', ENCRYPTBYPASSPHRASE(@EncryptionKey, '987-65-4321'), ENCRYPTBYPASSPHRASE(@EncryptionKey, '5500-0000-0000-0004'));

**C. Decrypt Data (Authorized Users Only)**

Only **authorized users** can decrypt the data using a stored procedure.

CREATE PROCEDURE sp\_DecryptSensitiveData

    @CustomerID INT,

    @Password NVARCHAR(50)

AS

BEGIN

    DECLARE @SSN NVARCHAR(11);

    DECLARE @CreditCard NVARCHAR(19);

    SELECT

        @SSN = CONVERT(NVARCHAR(50), DECRYPTBYPASSPHRASE(@Password, SSN)),

        @CreditCard = CONVERT(NVARCHAR(50), DECRYPTBYPASSPHRASE(@Password, CreditCardNumber))

    FROM Customers WHERE CustomerID = @CustomerID;

    IF @SSN IS NOT NULL AND @CreditCard IS NOT NULL

        SELECT @CustomerID AS CustomerID, @SSN AS SSN, @CreditCard AS CreditCardNumber;

    ELSE

        RAISERROR('Unauthorized access or incorrect password!', 16, 1);

END;

**Usage:**

EXEC sp\_DecryptSensitiveData @CustomerID = 1, @Password = 'SecureKey2025!';

**D. Security Policy - Restrict Decryption to Admins**

To **enhance security**, limit access to decryption using a **database role**.

CREATE ROLE SecurityAdmin;

GRANT EXECUTE ON sp\_DecryptSensitiveData TO SecurityAdmin;

Only users in the SecurityAdmin role can decrypt data:

EXEC sp\_addrolemember 'SecurityAdmin', 'admin\_user';

**Stored Procedures**

**A. Procedure to Encrypt Data (sp\_EncryptSensitiveData.sql)**

This stored procedure **inserts encrypted sensitive data** into the Customers table.

CREATE PROCEDURE sp\_EncryptSensitiveData

    @Name NVARCHAR(100),

    @SSN NVARCHAR(50),

    @CreditCardNumber NVARCHAR(50),

    @Password NVARCHAR(50)

AS

BEGIN

    SET NOCOUNT ON;

    INSERT INTO Customers (Name, SSN, CreditCardNumber)

    VALUES

        (@Name,

         ENCRYPTBYPASSPHRASE(@Password, @SSN),

         ENCRYPTBYPASSPHRASE(@Password, @CreditCardNumber));

    PRINT 'Data has been securely encrypted and stored.';

END;

**Usage:**

EXEC sp\_EncryptSensitiveData

    @Name='Alice Brown',

    @SSN='222-33-4444',

    @CreditCardNumber='1234-5678-9012-3456',

    @Password='SecureKey2025!';

**B. Procedure to Decrypt Data (sp\_DecryptSensitiveData.sql)**

This stored procedure **retrieves and decrypts sensitive data** for authorized users.

CREATE PROCEDURE sp\_DecryptSensitiveData

    @CustomerID INT,

    @Password NVARCHAR(50)

AS

BEGIN

    SET NOCOUNT ON;

    DECLARE @DecryptedSSN NVARCHAR(50);

    DECLARE @DecryptedCreditCard NVARCHAR(50);

    SELECT

        @DecryptedSSN = CONVERT(NVARCHAR(50), DECRYPTBYPASSPHRASE(@Password, SSN)),

        @DecryptedCreditCard = CONVERT(NVARCHAR(50), DECRYPTBYPASSPHRASE(@Password, CreditCardNumber))

    FROM Customers WHERE CustomerID = @CustomerID;

    IF @DecryptedSSN IS NOT NULL AND @DecryptedCreditCard IS NOT NULL

    BEGIN

        PRINT 'Decryption Successful!';

        SELECT @CustomerID AS CustomerID, @DecryptedSSN AS SSN, @DecryptedCreditCard AS CreditCardNumber;

    END

    ELSE

    BEGIN

        RAISERROR('Unauthorized access or incorrect password!', 16, 1);

    END

END;

**Usage:**

EXEC sp\_DecryptSensitiveData

    @CustomerID=1,

    @Password='SecureKey2025!';

**E. Master Script for Full Setup**

This script sets up the entire encryption system in one execution.

-- Create Tables

PRINT 'Creating Customers Table...';

GO

:r Tables/Create\_Customers\_Table.sql

GO

-- Encrypt Data

PRINT 'Encrypting Data...';

GO

:r Security/Encrypt\_Data.sql

GO

-- Create Security Policies

PRINT 'Setting Security Policies...';

GO

:r Security/Security\_Policy.sql

GO

-- Create Stored Procedures

PRINT 'Creating Stored Procedures...';

GO

:r Procedures/sp\_EncryptSensitiveData.sql

GO

:r Procedures/sp\_DecryptSensitiveData.sql

GO

PRINT 'Data Encryption & Security Setup Completed!';

**Execution:**

:r Scripts/Setup\_Encryption.sql

**3. Usage and Monitoring**

1. **Insert encrypted data:**

EXEC sp\_EncryptSensitiveData @Name='Alice Brown', @SSN='222-33-4444', @CreditCardNumber='1234-5678-9012-3456';

1. **Decrypt data (authorized users only):**

EXEC sp\_DecryptSensitiveData @CustomerID=1, @Password='SecureKey2025!';

1. **Restrict decryption to SecurityAdmin role:**

EXEC sp\_addrolemember 'SecurityAdmin', 'admin\_user';

**4. Benefits of This Approach**

**Strong encryption** for sensitive data.  
**Role-based security** for controlled decryption.  
**Modular design** for easy maintenance and scalability.

This **Data Encryption & Security project** ensures that sensitive data remains **secure and only accessible to authorized users**. Let me know if you need any modifications!