Standard Operating Procedure (SOP) for **SQL Server Encryption Features and** common encryption best practices.

Source: https://red9.com/blog/sql-server-best-encryption-features/

SOP: SQL Server Encryption Implementation — Latest Standards

1. Purpose

Define standardized steps for selecting, configuring, and managing encryption in SQL Server to protect sensitive data **at rest**, **in motion**, and **from unauthorized access**, ensuring security, compliance, and operational usability.

2. Scope

Applies to all SQL Server instances (on-premises and cloud) requiring encryption for data protection, including database administrators, security engineers, and compliance teams.

3. Definitions

Term	Description
TDE	Transparent Data Encryption — encrypts entire database files at rest.
Always Encrypted	Client-side column encryption protecting sensitive column data from DBAs and unauthorized access.
Cell-Level Encryption	Encryption applied to individual values or columns via encryption functions.
SSL / TLS	Encryption for data in transit (network traffic).

https://www.sqldbachamps.com/

4. Encryption Options Overview

4.1 Transport Encryption (SSL/TLS)

- Ensure all client-server connections use SSL/TLS to encrypt data in motion.
- Configure certificates on SQL Server and enforce encrypted connections in client apps.
- Validate certificate trust via enterprise CA when possible.

4.2 Transparent Data Encryption (TDE)

Purpose: Encrypts *entire database and transaction logs at rest* without application changes.
Use When: Protecting against data theft from backups, media, or stolen servers.
Key Steps:

1. Create database master key (DMK).
2. Create or import server certificate/ key.
3. Create Database Encryption Key (DEK) encrypted by certificate.
4. Enable TDE (ALTER DATABASE … SET ENCRYPTION ON).

Notes:

- TDE does *not encrypt data in memory or network traffic*.
- Ensure regular backups and key backups.
- TDE key management can use service managed or customer managed keys (e.g., Azure Key Vault).

4.3 Always Encrypted

Purpose: Encrypts specific columns via *client-side encryption*, ensuring data is never exposed to SQL Server in plaintext.
Use When: Protecting highly sensitive data even from DBAs (SSN, credit card numbers, PII).

Key Steps:

1. Define Column Master Key (CMK) stored in external store (Windows certificate store, Azure Key Vault, HSM).
2. Create Column Encryption Keys (CEK) encrypted by CMK.
3. Enable encryption on specific columns.
4. Use supported client drivers for application encryption/decryption.

Notes:

- Requires *minimal application changes*.
- Supports limited operations on encrypted columns (simple filters, limited functions).
- Consider using *Always Encrypted with secure enclaves* for richer query support.

4.4 Cell-Level Encryption

Purpose: Encrypt specific values via explicit SQL encryption functions (e.g., ENCRYPTBYKEY).

Use When: You need flexibility over specific columns or fine-grained encryption decisions.

Key Steps:

1. Create database master key.
2. Create certificate and symmetric keys.
3. Use encryption functions to encrypt/decrypt.

Notes:

- Requires schema changes (e.g., VARBINARY) and explicit encryption/decryption logic.
- May impact performance and query optimization.

5. Encryption Configuration Standards

5.1 Key Management

https://www.sqldbachamps.com/

- Use **external key stores** (Azure Key Vault, HSM) where possible.
- Establish **key rotation policies** (e.g., quarterly).
- Backup all keys and certificates securely.

5.2 Encryption Testing & Validation

- Test encryption in non-production environments before deployment.
- Validate application compatibility, especially with Always Encrypted and CLE.
- Confirm clients use updated drivers supporting encryption.

5.3 Backup and Recovery

- Ensure encrypted databases' keys are included in backup strategy.
- Store encryption keys outside the SQL Server instance.
- Validate restore process includes key restoration.

6. Operational Procedures

6.1 Monitoring & Alerts

- Monitor encryption status via sys.dm_database_encryption_keys for TDE.
- Track access patterns and potential decryption failures.
- Alert on key expiry or failures.

6.2 Maintenance and Compliance

- Regularly review encryption compliance against standards (PCI, HIPAA, GDPR).
- Document changes to encryption configuration and key rotation activities.

7. Limitations & Considerations

- **Always Encrypted** limits some query functions and requires client-side support.
- **TDE** does not protect data in memory or network; combine with TLS/SSL.
- **Cell-Level Encryption** increases complexity and application changes.
- Encryption increases CPU overhead; size performance testing.

8. Summary

Encryption Type	Protects Data At Rest	Protects Sensitive Columns	Requires App Changes	Best Used For
SSL/TLS	No	No	No	Network traffic encryption
TDE	Yes	No	No	Backup/disk encryption
Always Encrypted	Yes (column only)	Yes	Yes	High-sensitivity columns
Cell Level	Yes (column only)	Yes	Yes	Custom encryption needs

This SOP aligns with **industry practices and encryption feature guide** and supports secure, standardized encryption implementations for SQL Server.

https://www.sqldbachamps.com/

Sample SQL Server Encryption Policy Template that can be used for documentation, audit, and compliance purposes. It is aligned with the latest best practices and covers TDE, Always Encrypted, SSL/TLS, and Cell-Level Encryption.

SQL Server Encryption Policy Template

1. Purpose

This document defines the standard policy for encrypting SQL Server databases and associated data to ensure confidentiality, integrity, and compliance with regulatory standards (e.g., GDPR, PCI, HIPAA).

2. Scope

- Applies to all SQL Server instances (production, development, test).
- Covers encryption of data **at rest**, **in transit**, and **specific sensitive columns**.

3. Encryption Standards

Encryption Type	Scope / Coverage	Configuration / Notes	Responsible
TDE (Transparent Data Encryption)	Entire database and transaction logs	- Create Database Encryption Key (DEK) - Backup certificate securely - Enable TDE on production databases	DBA Team
Always Encrypted	Sensitive columns (PII, financial, health data)	- Define Column Master Key (CMK) in secure external store - Define Column Encryption Key (CEK) - Enable Always Encrypted for targeted columns	DBA / App Team
Cell-Level Encryption	Specific columns or values	- Use database master key and symmetric keys - Apply ENCRYPTBYKEY / DECRYPTBYKEY functions	DBA / Dev Team
SSL/TLS	Network traffic	- Enforce encrypted client connections - Configure certificates from trusted CA	Network / DBA Team

https://www.sqldbachamps.com/

4. Key Management Policy

- **Master Keys, Certificates, and DEKs** must be backed up and stored securely.
- Keys must be rotated according to organizational standards (e.g., every 6–12 months).
- Keys stored in **external key stores** (Azure Key Vault, HSM) wherever possible.
- Lost or compromised keys must be reported immediately.

5. Backup & Restore

- Encrypted databases require **backup of encryption keys**.
- Backup verification must include **restoration tests** with keys.
- Ensure disaster recovery and high availability environments also follow encryption standards.

6. Monitoring & Compliance

- Use system views (sys.dm_database_encryption_keys) to monitor TDE status.
- Log all encryption changes, key rotations, and access to sensitive columns.
- Audit for compliance with regulations and internal security policies.
- Alert for failed decryption attempts or disabled encryption features.

7. Operational Guidelines

- Autogrow and disk management should consider encrypted file sizes.
- Minimize performance impact by monitoring CPU usage for encrypted workloads.
- For Always Encrypted, ensure applications and drivers support client-side encryption.

8. Exceptions & Approvals

- Any deviation from this policy requires written approval from **Security Officer / DBA Manager**.
- Exceptions must include mitigation plans and risk assessment.

9. References

- Microsoft SQL Server Encryption: TDE, Always Encrypted, Cell-Level Encryption.
- Red9 Blog: Best SQL Server Encryption Features
- Microsoft Documentation: SQL Server Security Documentation

10. Review & Update

- This policy must be **reviewed annually** or when major SQL Server versions/features are introduced.
- Updates must be approved by **DBA Lead / Security Officer**.