

Data Encryption

Last updated by | Hamza Aqel | Jan 9, 2023 at 10:03 AM PST

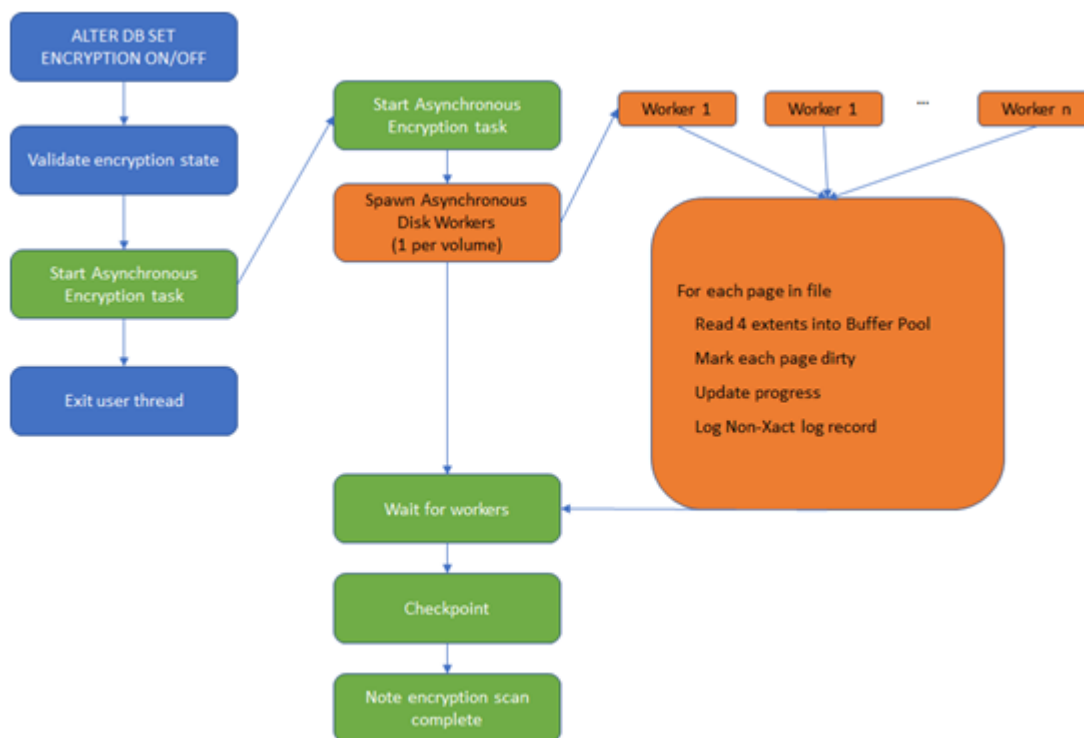
1. Transparent Data Encryption and PostgreSQL:

Some commercial databases such as Oracle and SQL Server provide Transparent Data Encryption (TDE) feature, not currently available in PostgreSQL database. Its currently a [proposed feature](#), so it may be released in near future, but as of current mainstream version (PostgreSQL 14) its not available. Microsoft offers TDE as part of its Microsoft SQL Server 2008, 2008 R2, 2012, 2014, 2016, 2017 and 2019. Oracle requires the Oracle Advanced Security option for Oracle 10g and 11g to enable TDE.

So how does TDE work?

Although product by product there are differences in TDE implementation between major commercial database vendors, we can take a look at SQL Server\SQL Azure design as good example. TDE performs real-time I/O encryption and decryption of the data at the database page level. Each page is decrypted when it's read into memory and then encrypted before being written to disk. TDE encrypts the storage of an entire database by using a symmetric key called the Database Encryption Key (DEK). On database startup, the encrypted DEK is decrypted and then used for decryption and re-encryption of the database files in the SQL Server database engine process. DEK is protected by the TDE protector. TDE protector is either a service-managed certificate (service-managed transparent data encryption) or an asymmetric key stored in Azure Key Vault (customer-managed transparent data encryption). With Oracle TDE can be done on tablespace vs database level, but as stated architecture is similar.

Internals of SQL Server TDE Encryption Scan are illustrated below:



So in all cases TDE encrypts and decrypts on **higher level than disk, usually database page level**. Once data reaches buffer its unencrypted, so anyone who has access to the database can read data. Main advantage here

is Without the original encryption certificate and master key, the data cannot be read when the drive is accessed or the physical media is stolen.

2. Azure storage encryption and Azure Database for PostgreSQL:






For storage encryption, Azure Database for PostgreSQL uses the FIPS 140-2 validated cryptographic module. Data is encrypted on disk, including backups and the temporary files created while queries are running. The service uses the AES 256-bit cipher included in Azure storage encryption, and the keys are system managed. This is similar to other at-rest encryption technologies, like TDE. Storage encryption is always on and can't be disabled. Encryption at rest provides data protection for stored data (at rest). Attacks against data at-rest include attempts to obtain physical access to the hardware on which the data is stored, and then compromise the contained data. In such an attack, a server's hard drive may have been mishandled during maintenance allowing an attacker to remove the hard drive. Later the attacker would put the hard drive into a computer under their control to attempt to access the data. Storage encryption at rest is designed to prevent the attacker from accessing the unencrypted data by ensuring the data is encrypted when on disk. If an attacker obtains a hard drive with encrypted data but not the encryption keys, the attacker must defeat the encryption to read the data. This attack is much more complex and resource consuming than accessing unencrypted data on a hard drive. For this reason, encryption at rest is highly recommended and is a high priority requirement for many organizations.

3. Similarities and differences between TDE and Storage Encryption.

In both cases , either with TDE or storage encryption we are talking about securing data at rest through encryption. Difference here is what mechanism is doing encryption at rest (Database Service vs Storage Service)

- With TDE database software is responsible for encrypting and decrypting data as it is moved out of\into the buffer pool on database page level. Encryption keys usually reside in database , although in the cloud may reside in third party service like Azure Key Vault. To get to the data one can either legitimately log into the database and query the data via SQL or read database buffer by using memory debugger tool by attaching to database process\daemon (WinDBG, GDB, LLDB)
- With storage encryption files are encrypted by the storage service using Cryptography APIs. To get to the data one can either legitimately login to database and query data via SQL , otherwise a hacker can read either OS or database process memory (aka buffer pool \buffers) using same memory debugging tools (WinDBG, GDB, LLDB). In both cases encryption is transparent to applications.

For more info please see following materials:

- [Transparent data encryption - Azure SQL Database & SQL Managed Instance & Azure Synapse Analytics | Microsoft Docs](#) 
- [Internals of TDE Encryption scan - Microsoft Tech Community](#) 
- [Transparent Data Encryption \(TDE\) - SQL Server | Microsoft Docs](#) 
- [Security in Azure Database for PostgreSQL - Flexible Server | Microsoft Docs](#) 
- [Azure Data Encryption-at-Rest - Azure Security | Microsoft Docs](#) 
- [Cryptography API: Next Generation - Win32 apps | Microsoft Docs](#) 