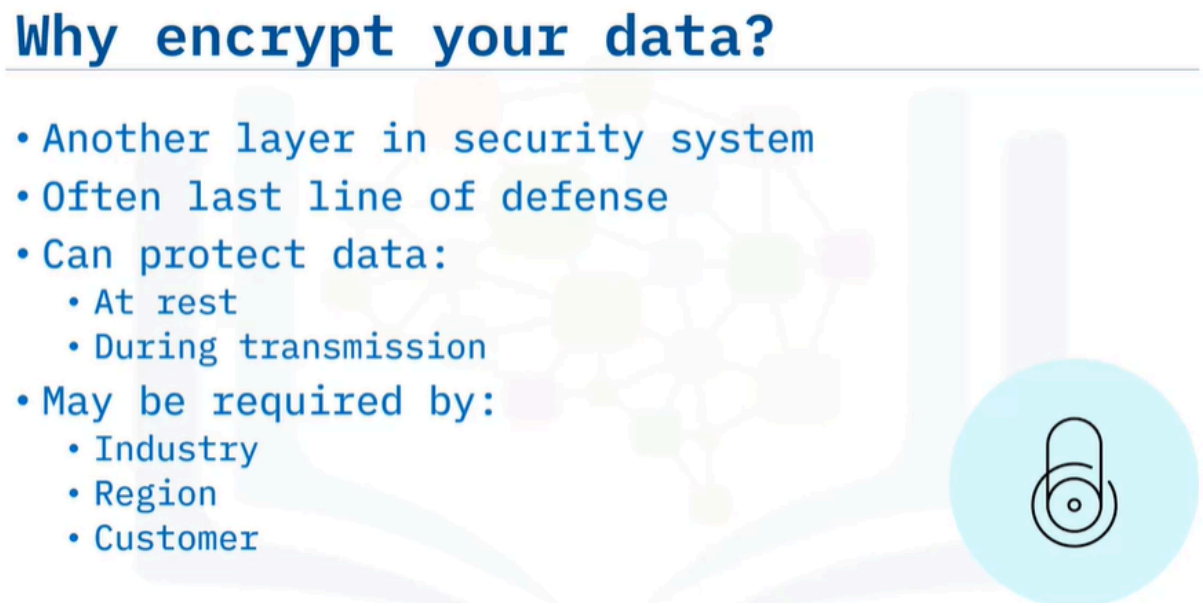
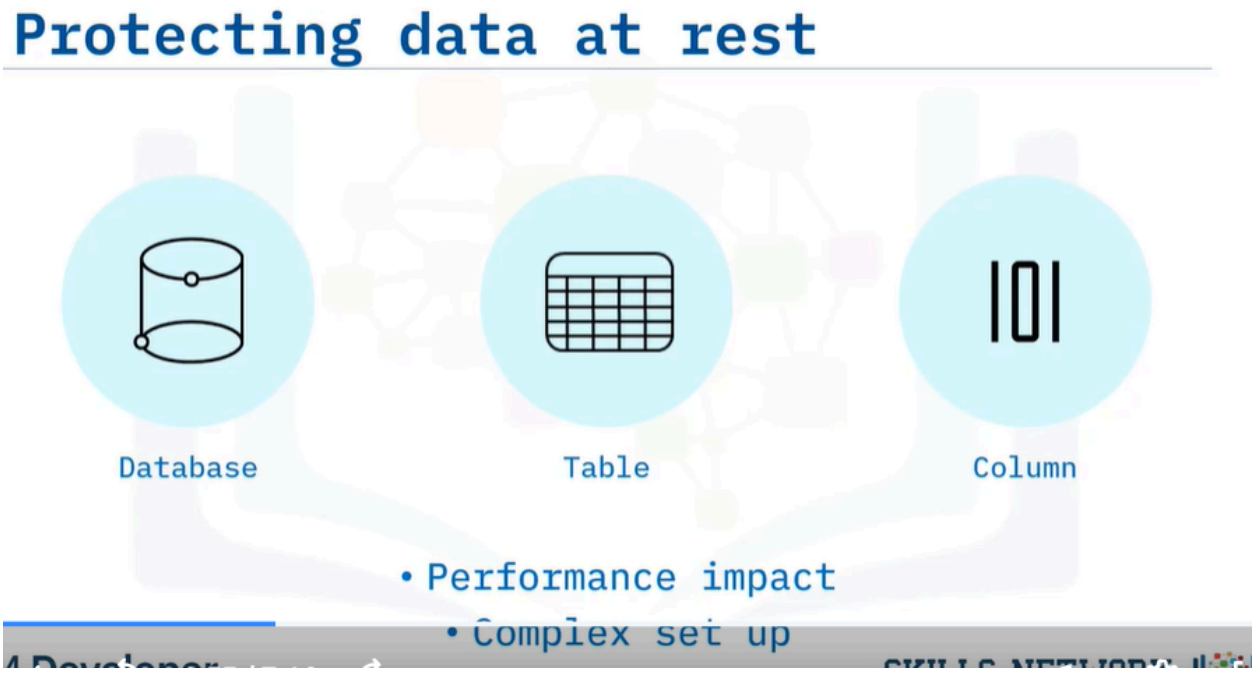
Encrypting Data

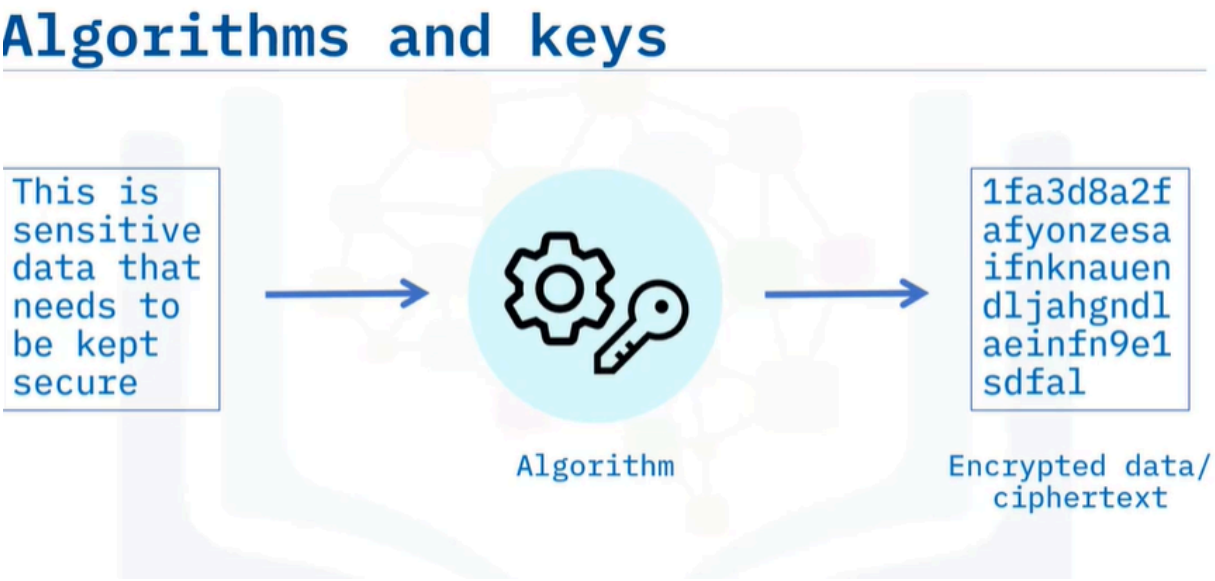


- Data encryption provides yet another layer in your security system and is often the last line of defense if data is compromised. It can help to protect your data at rest and during transmission. Some industries (such as financial and medical services), regions, or specific customers may also require that you encrypt certain types of data. For example, the Payment Card Industry Data Security Standard (PCI DSS) requires that all companies that process, transmit, or store credit card information encrypt that data.

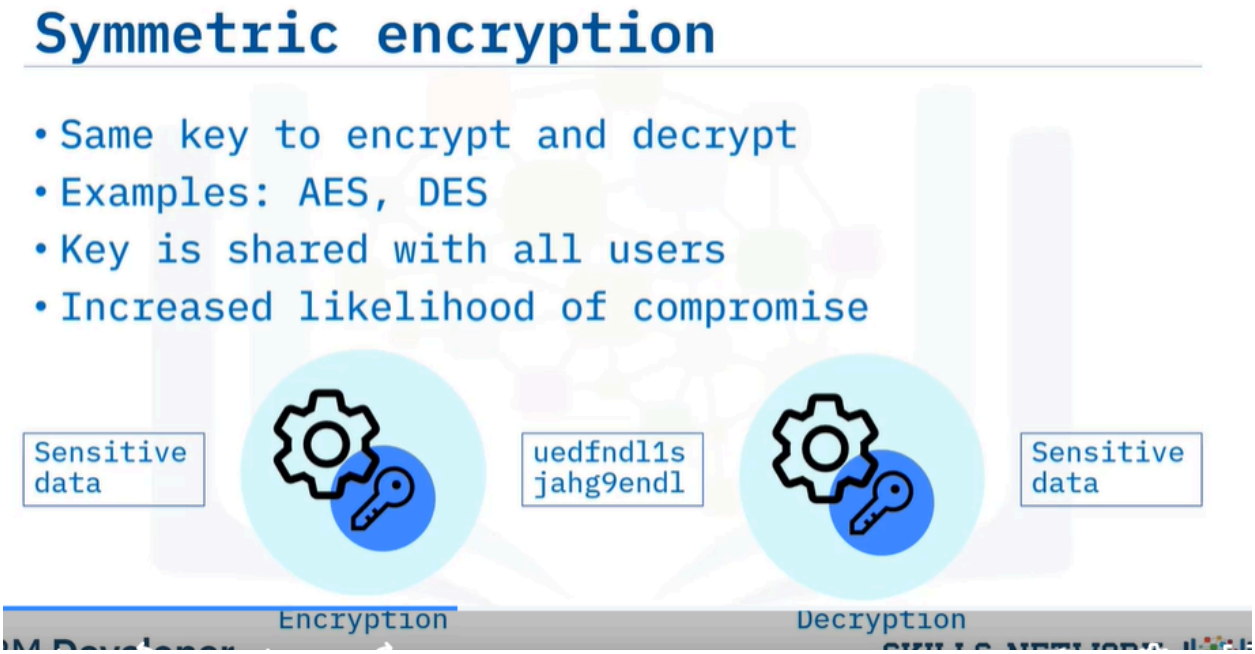


- “At rest data” refers to your data when it is stored, or at rest, in your RDBMS. It is important to encrypt data at rest to ensure that malicious users cannot directly access the data files, bypassing your database security measures. Some RDBMSs provide database-level encryption for the entire database. This is often termed transparent data encryption, or TDE. Some provide table or tablespace encryption for individual tables. And some provide column level for

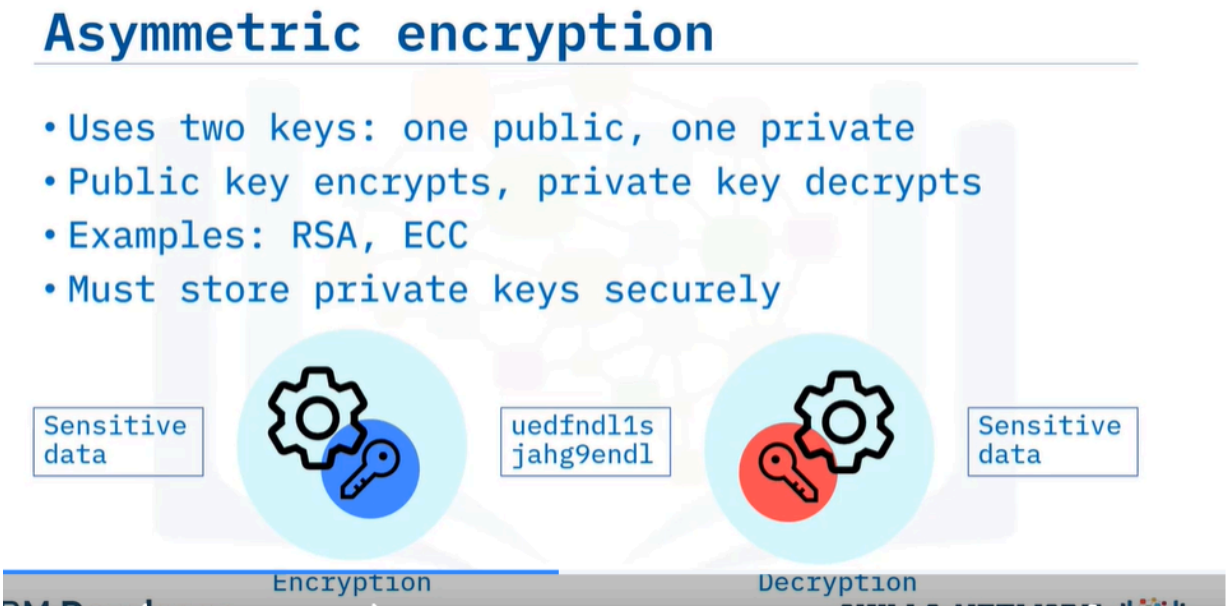
maximum granularity and flexibility. Do note though, that the flexibility that column level encryption provides is counteracted by performance degradation and complexity of set up.



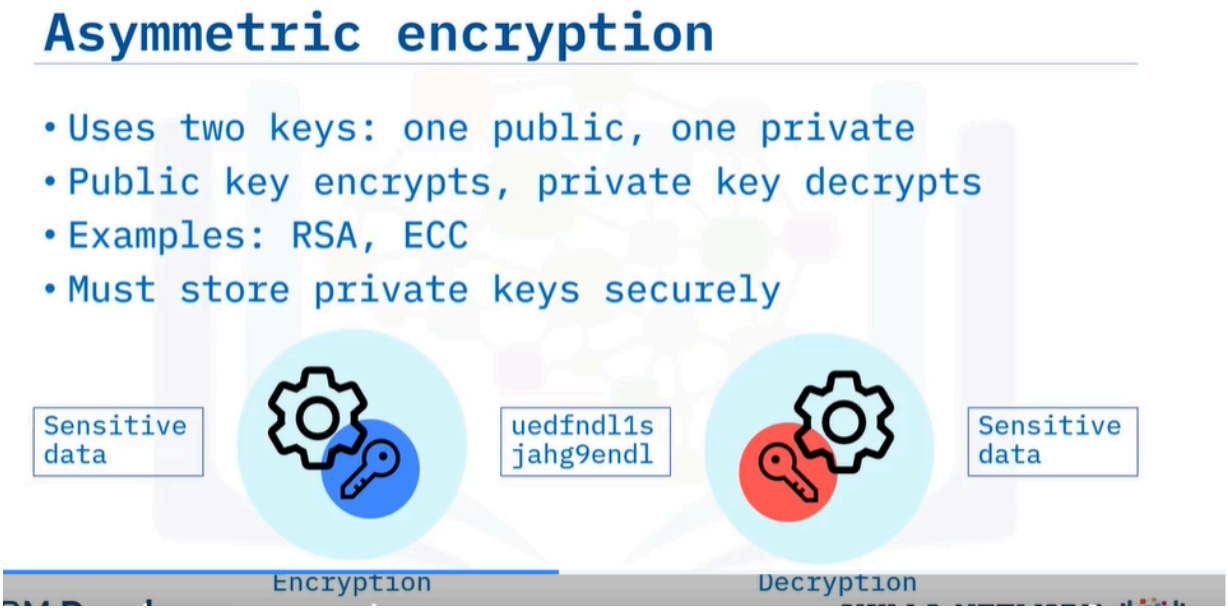
- You encrypt data by using an algorithm to change the data into an unreadable string, commonly known as ciphertext. These algorithms use a key to ensure that anyone trying to decipher the text cannot do so without the key.



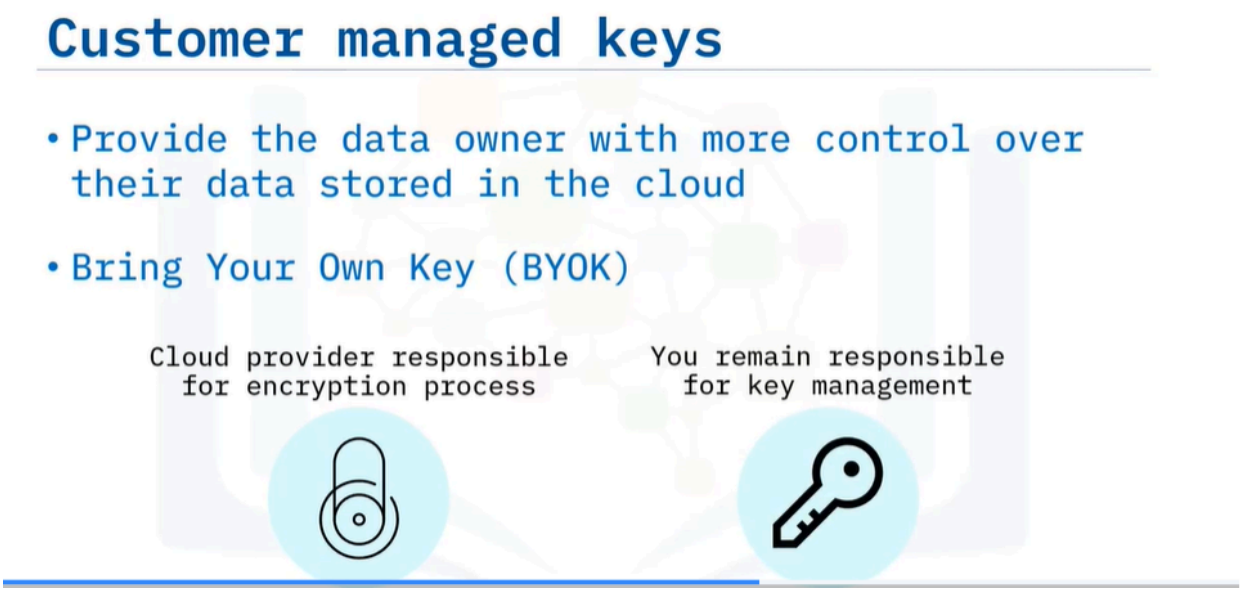
- The simplest form of encryption is symmetric encryption, where the same key is used to encrypt and decrypt the data. Mechanisms such as data encryption standard, or DES, and advanced encryption standard, or AES, use symmetric keys. When using only one key, it must be shared between all parties wanting to access that data so that they can all decrypt it. Sharing a key increases the likelihood of it being compromised and if the key to the encryption is compromised, your data is effectively compromised.

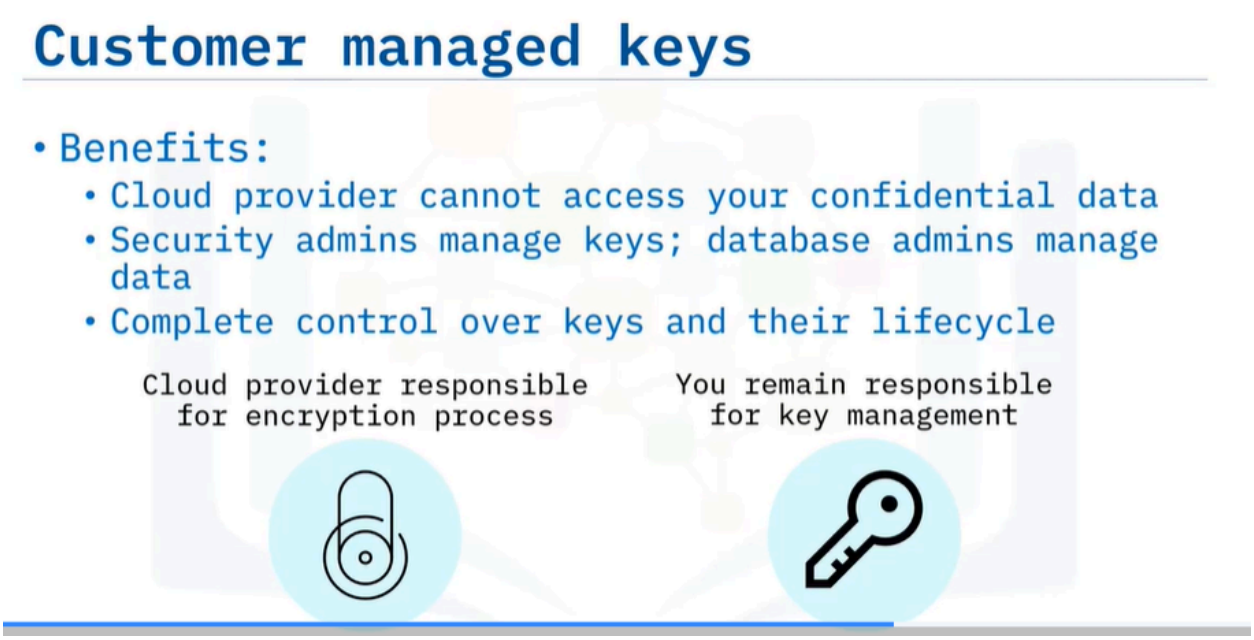


- Alternatively, some databases use asymmetric encryption, also known as public key encryption. This uses two keys: a public key and a private key, known as a key pair. You encrypt the data with the public key and valid users have a unique matching private key to decrypt it. Rivest-Shamir-Adleman (RSA) and elliptic curve cryptography (ECC) are both examples of asymmetric encryption. Because asymmetric encryption involves the use of multiple private keys, it is essential that those keys are held securely.



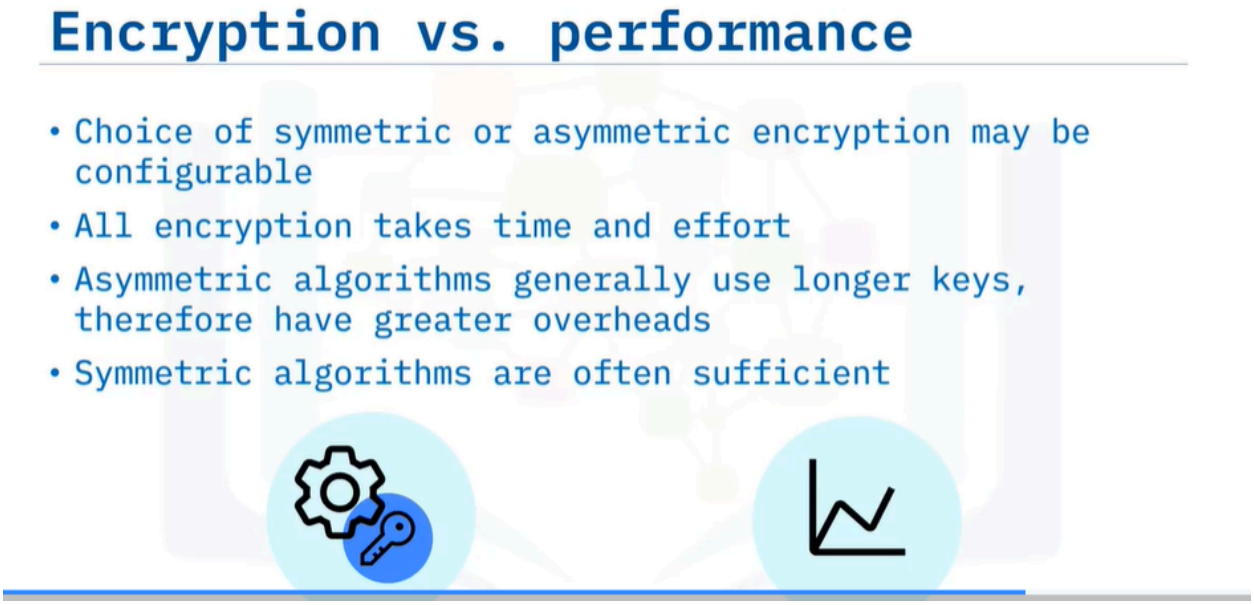
- Most modern databases take care of the complexity of encryption and key management for you. Many of them support transparent data encryption, or TDE. The “transparent” part of its name alludes to the fact that the encryption and decryption is not visible to your users. The database engine encrypts the data before storing it and then decrypts it when an authorized user or application requests information. Because the database engine is performing the encryption tasks, it also encrypts your database backups too..



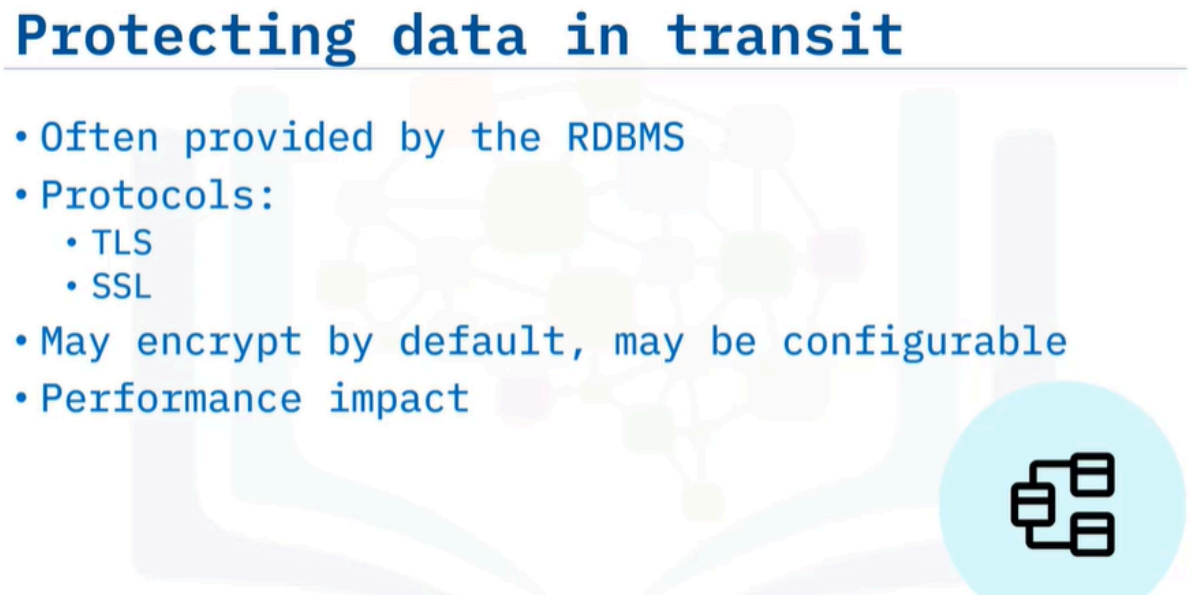


- Although TDE simplifies the process of encrypting your database, you may have concerns about storing data in the cloud and relinquishing the key management process to a third party. Some cloud databases support customer managed keys, also known as the Bring Your Own Key (or BYOK) protocol.

- This provides a separation of responsibility, so that your cloud database system is responsible for the encryption process, but you retain control of the keys that it uses. BYOK provides an extra level of protection by ensuring that the cloud platform storing and encrypting your data does not have access to the decryption keys, so it cannot access your confidential data. It also enables your security admins to manage your corporate keys, while your database admins can continue to manage the data. And for businesses whose customers require regulatory compliance around key rotation and expiration, you have complete control over the keys and their lifecycle.



- Some RDBMSs support both symmetric and asymmetric encryption and enable you to configure which to use. Because of the time and effort that it takes the algorithm to process the data, any type of encryption and decryption will decrease the performance of your database. Because asymmetric algorithms generally use longer keys, they tend to have a greater performance impact than symmetric algorithms. Therefore, unless you specifically require asymmetric encryption, it is often best to use the symmetric algorithms such as AES and DES. In fact, AES is the recommended method for the earlier PCI DSS credit card example.



- As well as encrypting data at rest, you should also consider encrypting it in transit, that is, when it is being transmitted across networks or the internet. As with data at rest encryption, many RDBMSs provide this functionality for you. Some cloud databases encrypt data in transit using transport layer security (or TLS) protocol, while others use the secure sockets layer (SSL) protocol. Some will encrypt your data by default, while others use a configuration setting that you can enable and disable. Similarly to at rest encryption, encrypting data in transit will affect the overall performance of your system.

Summary

In this video, you learned that:

- Data encryption helps make compromised data unusable
- You can encrypt data at rest or in transit
- You use algorithms to encrypt and decrypt data
- Symmetric encryption is easier to use, but less secure than asymmetric encryption
- TDE can simplify encryption setup
- Encryption decreases performance
- You can use TLS and SSL to protect data in transit