Niall Langley

Data Developer / Consultant

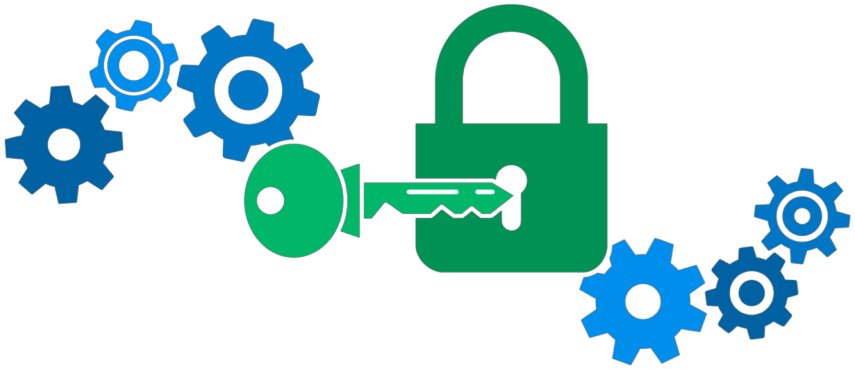
11 years experience with SQL Server doing OLTP and data warehousing / BI, now working with Azure data platform

Blog: https://www.sqlsmarts.com

LinkedIn: https://uk.linkedin.com/in/niall-langley

Twitter: @NiallLangley

SQL Server Encryption for the Layman



Introduction

- What is encryption
- Overview of types of encryption
- What do we want to protect
- How can SQL Server encrypt our data
- How else can we protect data in SQL server
- Summary

What is Encryption?

- SQL Server Encryption web page definition

  "Encryption is the process of obfuscating data by the use of a key or password"

- Encryption can make the data useless without the corresponding decryption key or password
- Encryption does not solve access control problems
- What do we need to encrypt?

Why do we Need to Encrypt Data?

- GDPR Compliance
  - Personally Identifiable Information
  - Fines for losing data are significant
- PCI Compliance
- Sensitive data or documents
- Losing data causes reputation damage
- Management who want to encrypt everything
- Reaction to a data loss incident

What do we want to Protect

- Data at Rest
  - Data files, log files, backups
  - Protect against losing copies of these
- Data in Transit
  - We don’t want data to be “sniffed” as it travels over the network
  - What is your application architecture?
- Data from Sysadmins (DBA’s)
  - Super users can see all data
  - Segregation of duties, can administer a server without access to sensitive data

Types of Encryption

- Symmetric Key Algorithms
  - AES, DES, Blowfish…
- Asymmetric Key Algorithms
  - RSA, Diffie–Hellman key exchange
- Cryptographic Hash Algorithms
  - MD5, SHA1…

Symmetric Key Algorithms

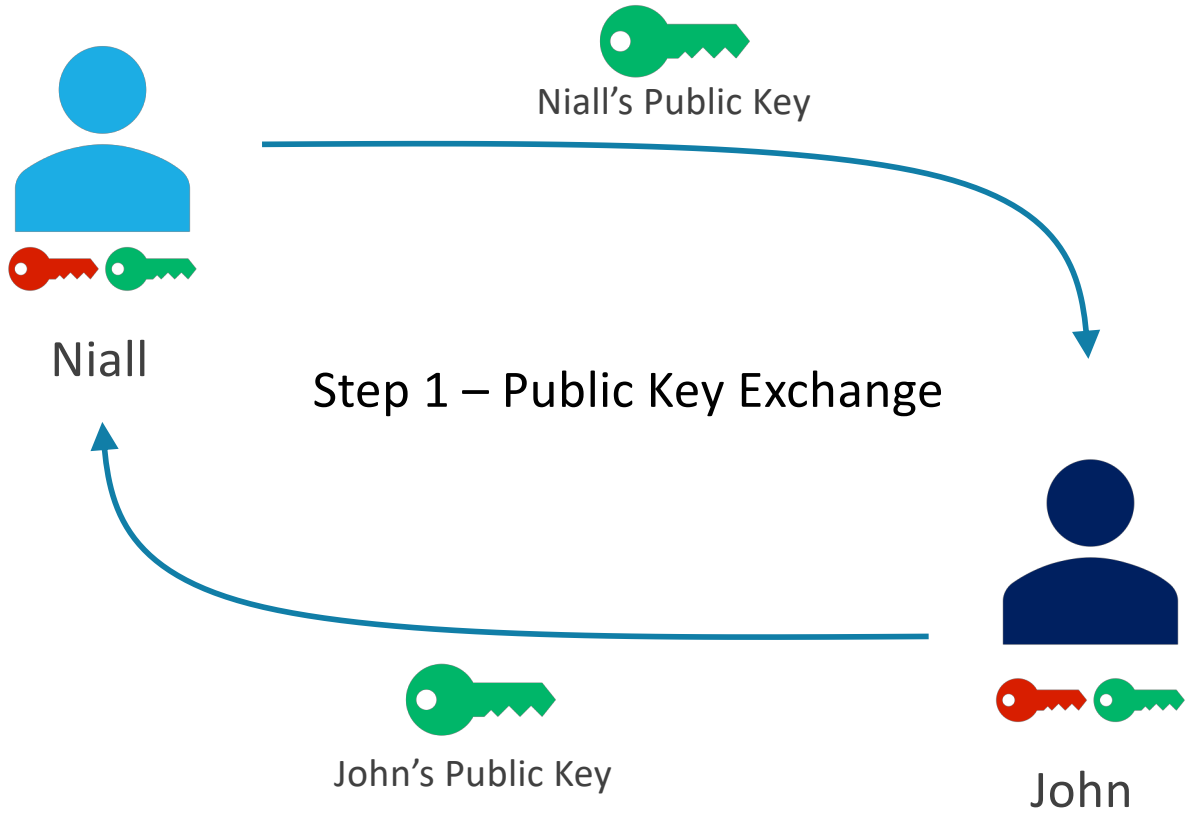
- Same key used to encrypt and decrypt the data

- Typically based on block cyphers
  - Data is broken into blocks smaller than or equal to the key length
  - Iterate through the blocks transforming using the XOR logical operator with key to encrypt or decrypt
  - XOR is simple, and therefore fast

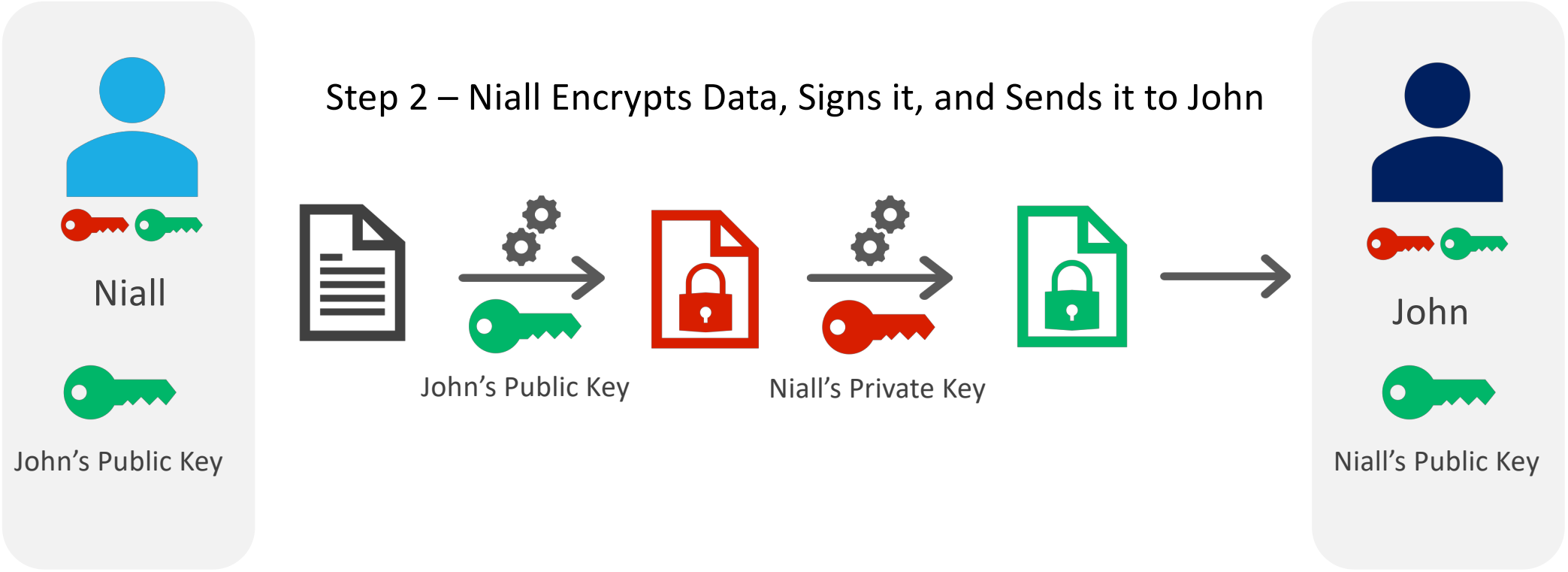
Asymmetric Key Algorithms

- Also know as Public Key Cryptography
- Uses two keys generated as a pair
  - Public one to encrypt data
  - Private one to decrypt data
- Typically based on trapdoor functions
  - RSA based on the factorization of the product of two prime numbers
- Asymmetric key algorithms tend to be slower than symmetric key algorithms
- Typically used for securing communication between two parties
- Certificates are based on Asymmetric Key Algorithms

Asymmetric Key Algorithms



Asymmetric Key Algorithms

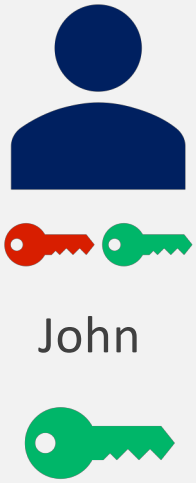


Asymmetric Key Algorithms

Step 3 – John Verifies the Data is from Niall, then Decrypts it Using his Private Key

Niall's Public Key

John's Private Key



John

Niall's Public Key

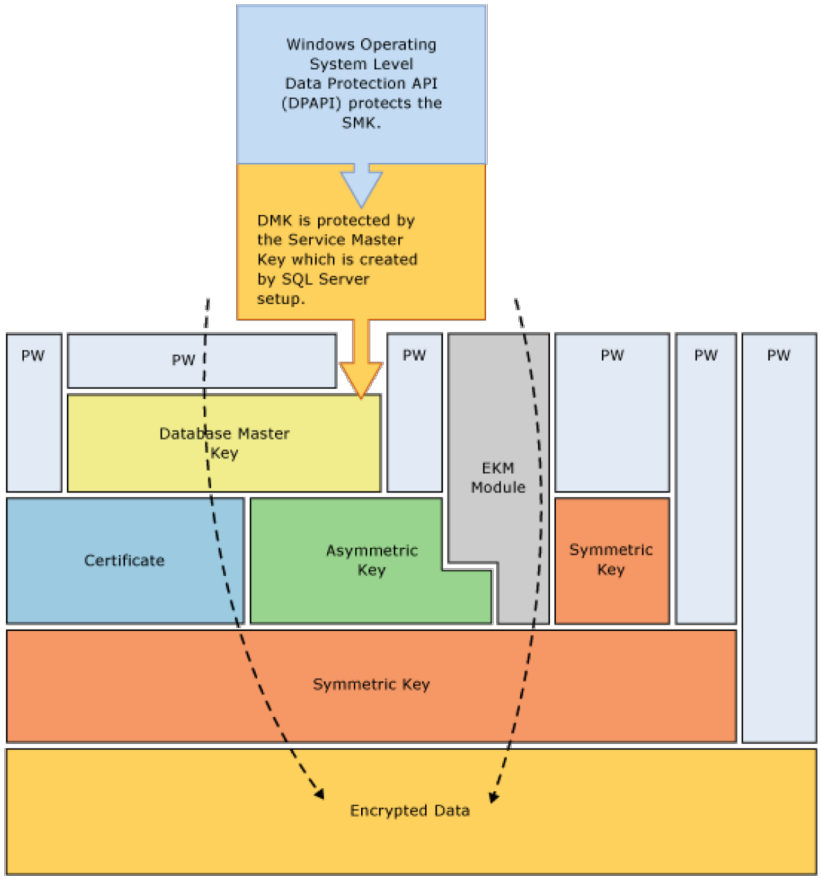
Cryptographic Hash Algorithms

- Produce a fixed length output from a variable length input
- Should be easy to calculate a hash for input data
- Should be extremely difficult to derive the input text from the output
  - One way functions
- Should have a very low chance two different inputs will produce the same output hash value
- Used to message integrity checks, digital signatures and authentication

So what are they for?

- Symmetric key encryption is typically used to:
  - Protect data
  - Protect asymmetric private keys at rest using a password
- Asymmetric key encryption is typically used to:
  - Protect symmetric keys in transit between parties
  - Create certificates used to protect data and verify the identity of third parties
- Cryptographic Hashes are typically used to:
  - Verify data and create digital signatures
  - Authentication by hashing passwords so they are not plaintext

SQL Server Encryption Hierarchy



- The Database Master Key is protected by a password, and optionally the Service Master Key
- This abstracts having to know the password to unlock a certificate or key away
- SQL Server permissions are used to grant access to encryption keys
  - Public key - `VIEW DEFINITION`
  - Private key - `VIEW DEFINITION` and `CONTROL`

Certificates in SQL Server

- We can create, import or export them
- SQL Server uses x509 certificates
  - Lots of utilities to create these
- Not required to be CA signed, or in date for securing data
- They are added to a specific database
  - SQL 2012 added support for import and export from binary blob
- SQL Server permissions are used to grant access to encryption keys
  - Public key - `VIEW DEFINITION`
  - Private key - `VIEW DEFINITION` and `CONTROL`

SSL - Data in Transit

- From SQL Server 2000 onwards
- Secures data in transit between server and client
- No code changes
- Self signed certificates can be used, but not advised
  - Risk of Man-in-the-Middle attack
  - Organisation Root CA certificates can be used to sign the server cert is installed on clients
- If the SQL Server is firewalled well, can be simpler to setup SSL on the router in front of it
  - This works really well for SSRS!

Column / Row Level Encryption – Data at Rest

- From SQL Server 2005 onwards
- Requires code changes, some queries not SARGable any more
  - SQL Server encryption functions

ENCRYPTBYPASSPHRASE	DECRYPTBYPASSPHRASE		
ENCRYPTBYKEY	DECRYPTBYKEY	DECRYPTBYKEYAUTOASYMKEY	DECRYPTBYKEYAUTOCERT
ENCRYPTBYASYMKEY	DECRYPTBYASYMKEY		
ENCRYPTBYCERT	DECRYPTBYCERT		

- SQL 2016 only AES_128, AES_192, and AES_256 are supported
- Data is protected from users without permissions
- Sysadmins always have control on the certificates and keys, so can access the data

Transparent Data Encryption (TDE) – Data at Rest

- From SQL Server 2008 onwards
- No code changes
- Protects data at rest, data files, log files and backups
  - But not FILESTREAM
- Only way to encrypt a backup on 2008
  - Until 2016, no backup compression
- Uses a server level certificate to protect the database encryption key
- Need the certificate to restore backups to another server
  - If you restore prod to dev, your prod certificate will be on dev!

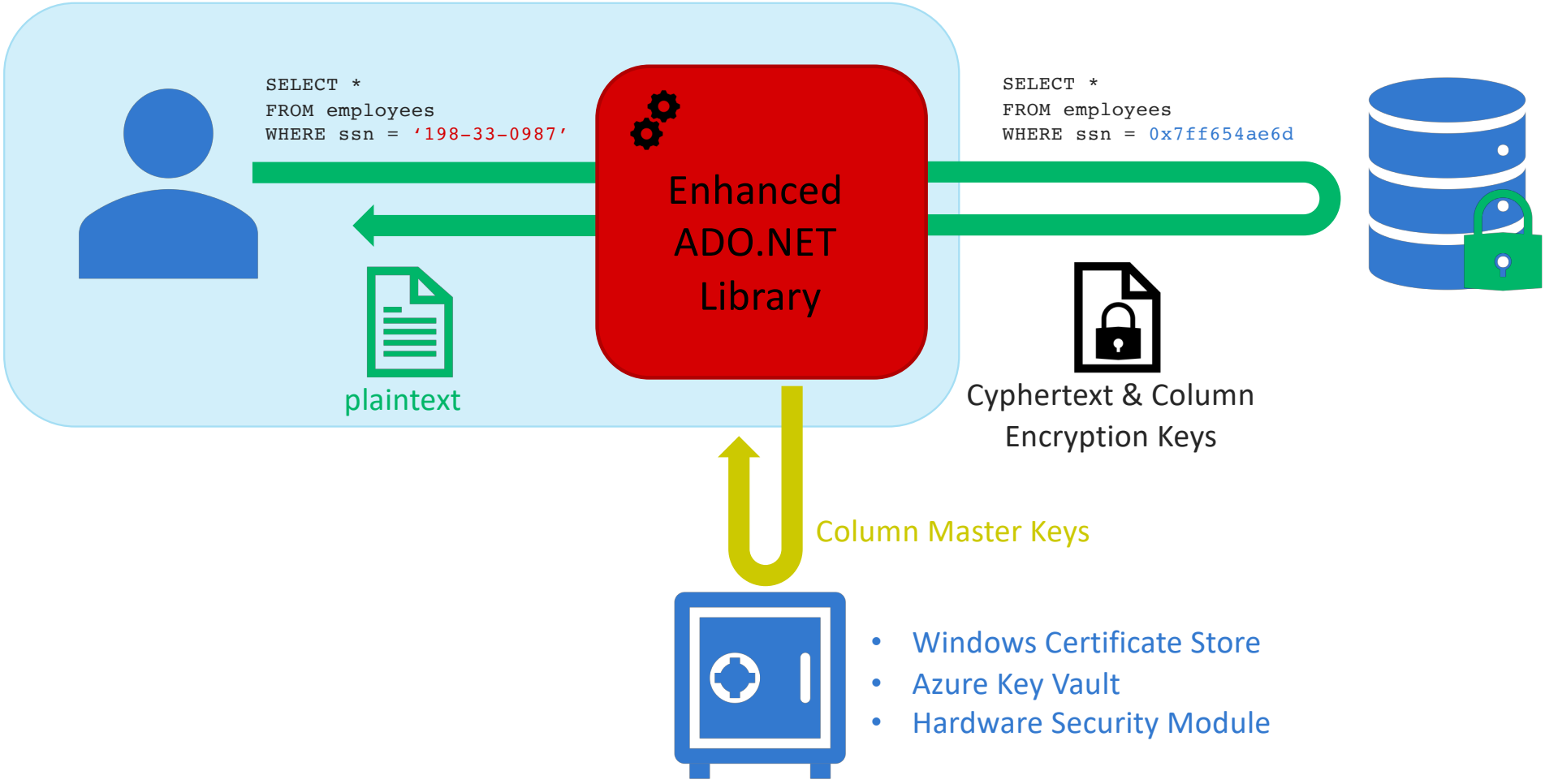
Backup Encryption – Data at Rest

- From SQL Server 2014 onwards
- Similar to TDE backup encryption, we can use a certificate or symmetric key
- Certificate or key must be available on server to restore an encrypted backup
- 2016 added compression support, but there are bugs in the RTM release
  - Make sure you are on the right CU
  - https://www.brentozar.com/archive/2016/07/tde-backup-compression-together-last/

Always Encrypted

- From SQL Server 2016 onwards
- End-to-end Encryption of individual columns
  - Data protected at rest, in transit and from sysadmins
  - Data is decrypted at the client
- Column master keys are stored in an external key store
  - Windows Certificate Store
  - Azure Key Vault
  - Hardware Security Module
- Column encryption keys are encrypted with the column master key and stored in the database

Always Encrypted



Always Encrypted

- Two Encryption Types
  - Deterministic - Always has same encrypted value for given plain text value
  - Randomized – Less predictable values, but doesn't support equality joins, indexing, lookups or grouping
- Code changes are required
  - Column collation must be Latin1_General_BIN2
  - Inequality predicates not supported
  - Some datatypes not supported – XML, IMAGE, SQL_VARIANT, etc.
  - https://www.red-gate.com/simple-talk/sql/database-administration/sql-server-encryption-always-encrypted/
- Requires a new enough version of the client driver

Encrypting Passwords

- Common in data leaks is a list of usernames and passwords
  - Biggest list found in 2018 was 770 Million usernames with passwords
- Passwords should be salted hashes, computed on the client
- Algorithm choice is key – 8 x High end graphic cards
  - MD5 - 200 Billion hashes per second
  - SHA1 – 69 Billion hashes per second
  - bcrypt (work factor 5) – 105 Thousand hashes per second
  - https://gist.github.com/epixoip/a83d38f412b4737e99bbef804a270c40
- Talk to your developers and understand their choices
- https://arstechnica.com/information-technology/2013/03/how-i-became-a-password-cracker/

Dynamic Data Masking

- SQL Server 2016 onwards
- Not actually encryption
- Requires simple code changes
- Results are returned with some data masked out
- Data can be read by sysadmins
  - Permissions
- Is susceptible to brute force attacks
  - We can infer the data of a column by filtering the masked data using a where clause
  - https://sqlsunday.com/2018/02/05/an-alternative-to-data-masking/

Summary

- Know what you need protect
  - At rest, in flight, from sysadmins
  - Types of data PII, sensitive data, specific data for compliance
- Understand trade-offs
  - Performance impact
  - Ease of use
  - Code / architecture changes
- Protect and backup keys
  - Don’t be the DBA in a DR situation with good backups but no keys!

Summary

	SQL Server Version	Data at Rest	Data in Transit	Data from Sysadmins	Data from unprivileged users	Requires Code Changes
SSL	2000	✗	✓	✗	✗	✗
Column / Row Level Encryption	2005	✗	✗	✗	✓	✓
Transparent Data Encryption (TDE)	2008	✓	✗	✗	✗	✗
Backup Encryption	2014	✓	✗	✗	✗	✗
Always Encrypted	2016	✓	✓	✓	✓	✓
Hashing Passwords	N/A – Client Side	✓	✗	✓	✓	✓
Dynamic Data Masking	2016	✗	✗	✗	✓	✗