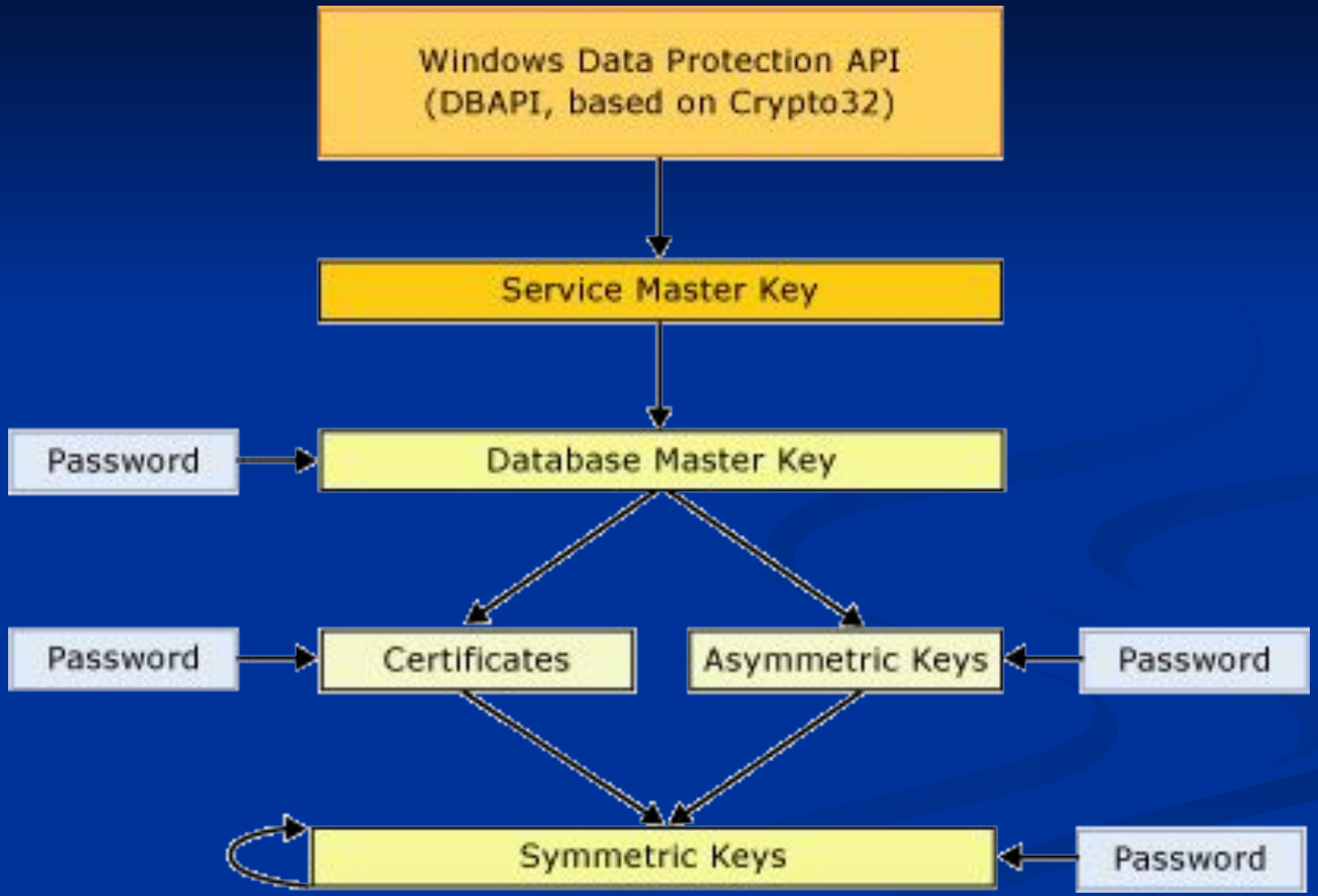
SQL SERVER ENCRYPTION



Key Management Hierarchy

Service Master Key

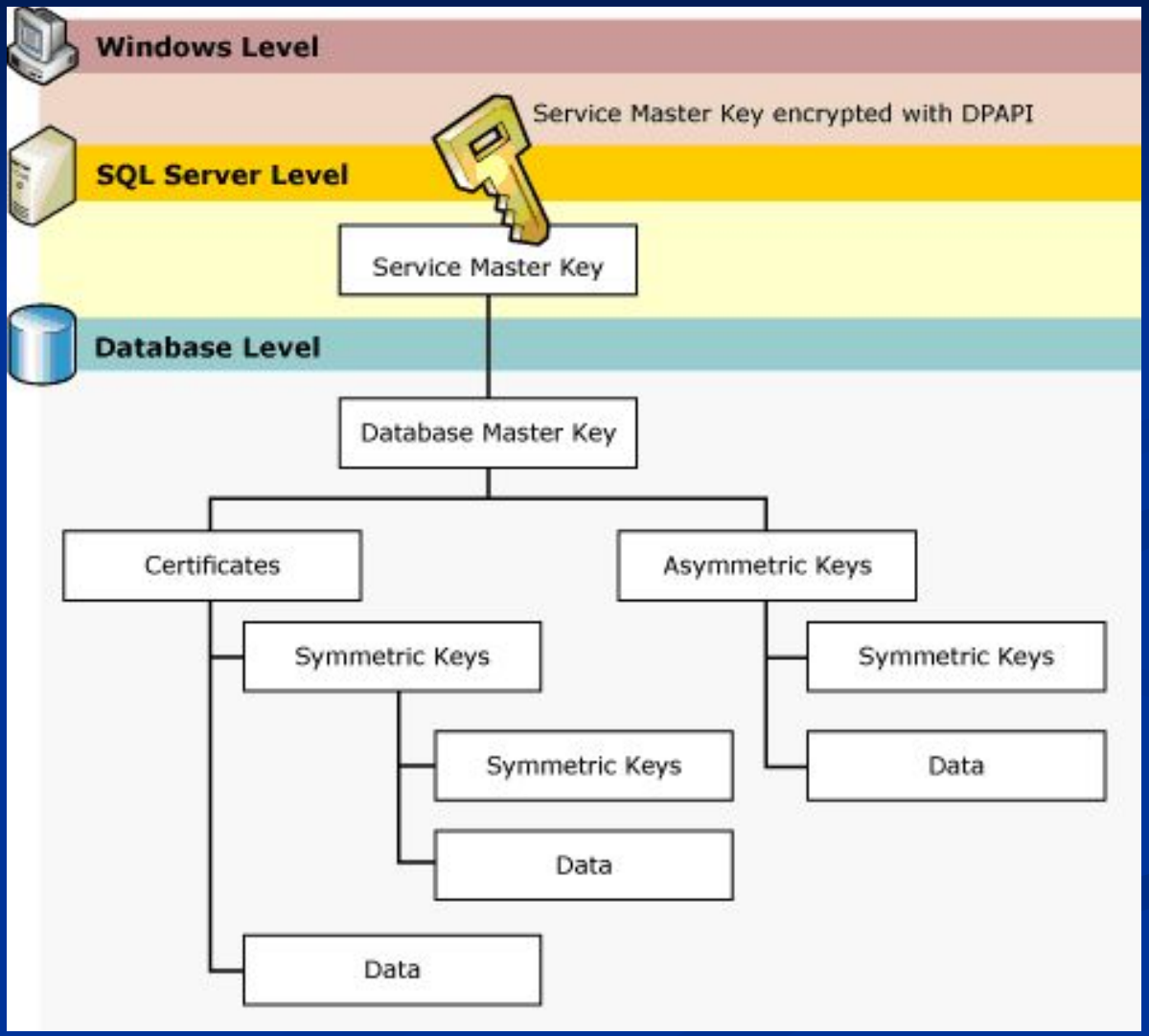
- Root – Top Level Key – symmetric key
- One Service Master key per installation
- Auto-Generated at time of installation
- Can not be directly access,
- Can be regenerated and exported
- Accessed by SQL Server Service account

Key Management Hierarchy

Key Management Hierarchy

- **Database Master Keys**
  - Symmetric key
  - One Database Master key per Database
  - Used to encrypt all user keys in the database
  - Copy stored encrypted with Service Master Key
  - Also stored encrypted with a password
  - Recommend removing copy stored with service master key if possible.
- **User Keys**
  - May be a certificates, asymmetric keys or symmetric key
  - Generated as needed by DBA or users
  - Stored encrypted with Database Master key

Key Management Hierarchy



MS SQL Server Protecting Password

- Avoid storing passwords if possible
- Use LDAP or Active Directory is possible
- Otherwise use Crypto API to generate secure salted hash:
  - *CryptGenRandom()* generates a salt
  - *CryptCreateHash()* creates hash object
  - *CryptHashData()* generated hash

MS SQL Server Checklist

1. Use either the AES192 or AES256 algorithm
2. Use randomly generated keys and passwords
   - generated by MS SQL Server,
   - or via CryptGenRandom() from MS Crypto API
3. Avoid storing keys or passwords in software
4. Remove the service key encrypted copy of the database master, if possible to reduce risk.

Increased size of encrypted data over its clear text

Algorithm	Maximum Increase	Minimum Increase	Average Increase
Triple DES	3.80	0.45	1.77
AES 128	5.40	0.54	2.51
AES 192	5.40	0.54	2.51
AES 256	5.40	0.54	2.51
Certificate	11.80	0.54	5.16
DES	3.80	0.45	1.77
DESX	3.80	0.45	1.77
RC2	3.80	0.45	1.77
RC4	3.60	0.43	1.73
RSA 1024	11.80	0.54	5.16
RSA 2048	24.60	2.08	11.31
RSA 512	5.40	1.06	3.23

SERVICE MASTER KEY MANAGEMENT

Backup and restore service master key:

```
BACKUP SERVICE MASTER KEY TO FILE = 'c:\key.dat'
   ENCRYPTION BY PASSWORD =
'S3@fBZir2D^P$x5P&tNr^uR!@wGW'

RESTORE SERVICE MASTER KEY
FROM FILE = 'c:\key.dat'
DECRYPTION
BY PASSWORD = 'S3@fBZir2D^P$x5P&tNr^uR!@wGW'
```

Regenerate service master key:

ALTER SERVICE MASTER KEY REGENERATE

DATABASE MASTER KEY MANAGEMENT

Create database master key:

CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'WeZ#6hv*XHq#akAEaqcr7%CUP3aQ'

Backup and restore database master key:

```
BACKUP SERVICE MASTER KEY TO FILE = 'c:\key.dat'
ENCRYPTION BY PASSWORD = 'S3@fBZir2D^P$x5P&tNr^uR!@wGW'

RESTORE SERVICE MASTER KEY
FROM FILE = 'c:\key.dat' DECRYPTION
BY PASSWORD = 'S3@fBZir2D^P$x5P&tNr^uR!@wGW'
```

Open and close database master key:

```
OPEN MASTER KEY DECRYPTION BY PASSWORD =
'WeZ#6hv*XHq#akAEaqcr7%CUP3aQ'

CLOSE MASTER KEY
```

ENCRYPTION FUNCTIONS

EncryptByKey
(DecryptByKey)

EncryptByPassPhrase
(DecryptByPassPhrase)

Key_ID

EncryptByAsmKey
(DecryptByAsmKey)

EncryptByCert
(DecryptByCert)

Cert_ID

! Use transparent encryption in SQL Server 2008

ENCRYPTION FUNCTIONS

USING PASSWORD FOR ENCRYPTION

```
EncryptByPassPhrase( { 'passphrase' | @passphrase } ,
                     { 'cleartext' | @cleartext }
                     [ , { add_authenticator | @add_authenticator } ,
                        { authenticator | @authenticator } ] )
```

SYMMETRIC ENCRYPTION WITH KEY

```
EncryptByKey ( key_GUID ,
                     { 'cleartext' | @cleartext }
          [ , { add_authenticator | @add_authenticator } ,
             { authenticator | @authenticator } ] )
```

Encryption for Oracle 8i-11g

- Oracle DBMS Obfuscation Toolkit (DOTK)
  (Only option for older Oracle 8g & 9g)
- Oracle DBMS_CRYPTO package
- Oracle Transparent Data Encryption (TDE)
- Oracle Advanced Security Option

Oracle DOTK Checklist

1. Use only the 3DES encryption rather than DES
2. Avoid for highly sensitive information, Use DBMS_CRYPTO when available
3. Use a randomly generated key of at least 128 bits generated from DES3GetKey().
4. Use a good source of entropy for the random seed used for generating the key such as /dev/random on Unix/Linux systems and CryptGenRandom() on MS windows.
5. Use a randomly generated IV (Initialization vector) of 8-16 bytes (64-128 bits) for each encrypted record.

Oracle DBMS Obfuscation Toolkit

DOTK Encryption Procedure:

```
DBMS_OBFUSCATION_TOOLKIT.DES3Encrypt(
   input_string IN VARCHAR2,
   key_string IN VARCHAR2,
   encrypted_string OUT VARCHAR2,
   which IN PLS_INTEGER DEFAULT TwoKeyMode,
   iv_string IN VARCHAR2 DEFAULT NULL);
```

- Also function with output returned
- Also function & procedures with **raw** parameters
- Default Null IV is Dangerous, should be random!

Oracle DBMS Obfuscation Toolkit

DOTK Decryption Procedure:

```
DBMS_OBFUSCATION_TOOLKIT.DES3Decrypt(
   input_string  IN   VARCHAR2,
   key_string    IN   VARCHAR2,
   decrypted_string  OUT  VARCHAR2,
   which         IN PLS_INTEGER DEFAULT TwoKeyMode
   iv_string     IN   VARCHAR2 DEFAUTL NULL);
```

- Also function with output returned
- Also function & procedures with **raw** parameters
- Need the same IV to decrypt.

Oracle DBMS Obfuscation Toolkit

DOTK DES3 Generate Key Procedure:

```
DBMS_OBFUSCATION_TOOLKIT.DES3GetKey(
 which       IN PLS_INTEGER DEFAULT TwoKeyMode,
 seed_string IN   VARCHAR2,
 key         OUT  VARCHAR2);
```

- Also function with output returned
- Also function & procedure with **raw** parameters
- Important to use Random seed.

Oracle DBMS Crypto Checklist

1. Use either the AES192 or AES256 algorithm
2. Use DBMS_CRYPTO.RANDOMBYTES() to generate random keys, not DBMS_RANDOM
3. Use CBC (Cipher Block Chaining) mode.
   CFB Cipher Feedback Mode and
   OFB Output Feedback Mode are both ok
4. Do not use ECB Electronic Codebook chaining mode (It is weak)
5. Use PKCS5 for cryptographic padding rather than null padding

Oracle DBMS Crypto Encryption

Sample Encrypt function

```
DBMS_CRYPTO.ENCRYPT(
 src IN RAW,
 typ IN PLS_INTEGER,
 key IN RAW,
 iv  IN RAW DEFAULT NULL)
RETURN RAW;
```

- Also procedure with output as a parameter
- Important to use Random IV.
- Decrypt function & procedure are very similar.

Oracle DBMS Crypto
Encrypt TYP Parameter

- **`TYP`** parameter specifies algorithms and modifiers

Feature	Options
Crypto algorithm	DES, 3DES, AES128, AES192, AES256, RC4, 3DES_2KEY
Padding forms	PKCS5, NONE, ZERO
Block Cipher chain mode	CBC, CFB, ECB, OFB

Oracle DBMS Crypto Encryption

DBMS Crypto Generate Random Bytes:

```
DBMS_CRYPTO.RANDOMBYTES (
   number_bytes IN POSITIVE)
 RETURN RAW;
```

- Use for Random IV and to generate random keys
- Do not use DBMS_RANDOM, as it's weak.

Custom Cryptographic functions based on DLLs

PL/SQL subprogram

Alias library

tnsnames.ora

Listener process

listener.ora

extpro
c
process

External procedure

Shared library