The following should help in using the utility.

- **Pre-requisite:**

Application level development experience & knowledge of Oracle PL/SQL (basic)

- **Specification:**

Supported Oracle Datatype(s)
VARCHAR2
NUMBER
RAW

Utility Element Name	Type	Description
ex_crypto	Package	Container for all encryption/decryption API bundle
ex_crypto.f_encrypt	Function	Generic Function to encrypt VARCHAR2 and NUMBER data type*
ex_crypto.f_encrypt_raw	Function	Function to encrypt RAW data type*
ex_crypto.f_decrypt	Function	Generic Function to decrypt VARCHAR2 data type*
ex_crypto.f_decrypt_raw	Function	Generic Function to decrypt RAW data type*
**: See description for details on arguments, return type and usage*		

SYNTAX:

```
EX_CRYPTO. F_ENCRYPT (
    in_encrypt          VARCHAR2,
    in_n_3des_flag      PLS_INTEGER DEFAULT 0) RETURN VARCHAR2;
```

Parameters

in_encrypt	The string to be encrypted. Max size 32767
In_n_3des_flag	Encrypt using 2-pass or 3-pass 3DES algorithm. Indicated by 0 or 1 value respectively. Default 0

```
EX_CRYPTO.F_ENCRYPT (
    in_encrypt          NUMBER,
    in_n_3des_flag      PLS_INTEGER DEFAULT 0) RETURN VARCHAR2;
```

Parameters

in_encrypt	The Number to be encrypted.
In_n_3des_flag	Encrypt using 2-pass or 3-pass 3DES algorithm. Indicated by 0 or 1 value respectively. Default 0

```
EX_CRYPTO.F_ENCRYPT (
    in_encrypt          RAW,
    in_n_3des_flag      PLS_INTEGER DEFAULT 0) RETURN RAW;
```

Parameters

in_encrypt	The RAW string to be encrypted.
In_n_3des_flag	Encrypt using 2-pass or 3-pass 3DES algorithm. Indicated by 0 or 1 value respectively. Default 0

```
EX_CRYPTO.F_DECRYPT (
    in_decrypt          VARCHAR2,
    in_n_3des_flag      PLS_INTEGER DEFAULT 0) RETURN VARCHAR2;
```

Parameters

in_decrypt	The string to be decrypted.
In_n_3des_flag	Decrypt using 2-pass or 3-pass 3DES algorithm. Indicated by 0 or 1 value respectively. Default 0

```
EX_CRYPTO.F_DECRYPT_RAW (
    in_decrypt          RAW,
    in_n_3des_flag      PLS_INTEGER DEFAULT 0) RETURN RAW;
```

Parameters

in_decrypt	The RAW string to be decrypted.
In_n_3des_flag	Decrypt using 2-pass or 3-pass 3DES algorithm. Indicated by 0 or 1 value respectively. Default 0

Subject Area	Sample Usage Syntax
ENCRYPTION	SELECT ex_crypto.f_encrypt('Exilant Technolgies Limited') FROM DUAL;
	SELECT id, ex_crypto.f_encrypt(nric_id) FROM customer;
	SELECT ex_crypto.f_encrypt_raw(raw_data_column) FROM customer_master;
	SELECT id, first_name, last_name, phone_nr FROM customer_master WHERE hashed_email_id IN (SELECT ex_crypto.f_encrypt(hashed_email) FROM hni_email_list);
	SELECT id,ex_crypto.f_encrypt('Exilant Technolgies Limited',1) FROM DUAL;
DECRYPTION	SELECT ex_crypto.f_decrypt(encrypted_nric) FROM customer_master;
	SELECT TO_NUMBER(ex_crypto.f_decrypt(hashed_cc)) FROM customer_master;
	SELECT id,ex_crypto.f_decrypt_raw(hashed_ssn) FROM customer_master;

FAQ:

- ✓ NLS_LANG support for multi-byte character set
- ✓ The APIs are designed to be fault tolerant. In case the encryption and/or decryption module fails, it will return NULL output. The error will be logged into a table named 'error_log' with suitable comments and module/function name. The generic error logging mechanism comes built-in along with the encryption package and can be leverage at DB level for centralized error logging if deemed fit.
- ✓ Customization to any package can be done suitably to meet performance/ functional needs. However, no support shall be provided for the same.
- ✓ Decryption algorithms are provided primarily for data quality/audit checks and debugging purposes.
- ✓ Storage of encrypted data usually takes 3-times the space of that of it's normal space. Hence encrypted columns in table should have sufficient width to hold the data.
- ✓ Oracle advises to not compress encrypted columns and not indexing them unless there is an absolute need.
- ✓ For ease and clarity of use, the common API for handling VARCHAR2 and NUMBER data types are overloaded (f_encrypt, f_decrypt) while those required for handling of RAW data type are named as f_encrypt_raw, f_decrypt_raw)
- ✓ The APIs return NULL when they hit any error while processing either encryption or decryption. The calling program needs to handle and process accordingly. The errors are neatly logged into the centralized error table: error_log and can be looked up suitably.