* **Pre-requisite:**

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

|  |
| --- |
| 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.