

BaroCRYPT(Oracle, Tiberio) 가이드

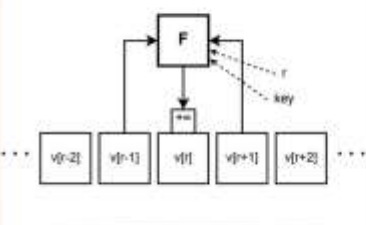
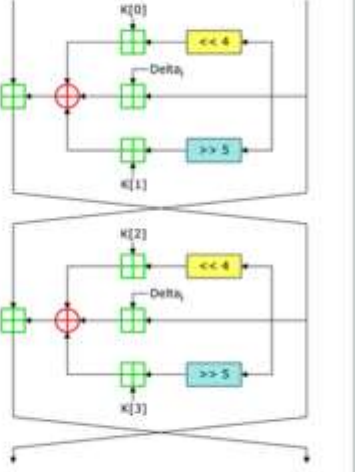
목차

목차.....	0
1. BaroCRYPT	1
1.1 BaroCRYPT 개요.....	1
1.2 BaroCRYPT 특/장점	1
2. BaroCRYPT 연동 API	3
2.1 연동 API 사용 전 준비사항	3
2.2 BaroCRYPT 연동 API	3
3. BaroCRYPT 연동 API (DB)	8
3.1 External Procedure 란?	8
3.2 장점 및 단점	8
3.3 Java 모듈(barocrypts.java).....	8
3.4 C 모듈(libbarocrypts.so).....	10
4. About BaroCRYPT.....	15

1. BaroCRYPT

1.1 BaroCRYPT 개요

BaroCRYPT 솔루션은 Feistel 암호를 사용하여 크기가 작고 구현이 쉬운 블록 암호화 알고리즘인 XXTEA (Extended Extended Tiny Encryption Algorithm)를 기반으로 하는 가볍고 가장 빠른 암호화 알고리즘이다.

General	Best public cryptanalysis	Two Feistel rounds(one cycle) of TEA
	<pre> void xorshift32_xorshift32_32(int * state, int cycle) { xorshift32_xor_32(state, 1); int i; int * p = cycle%state+state; int * q = state+2; for(i=0; i<cycle; i++) { int t = *(state+12); // the left weighting word int u = *(state+12); // the right weighting word int v = ((int)*q<>8+((int)*q<>16)*t)*v*(int)*p*(int)*p); state[state+12] = v; swapstate(state, state+2); } for(i=0; i<cycle; i++) state[state+12] = v; } </pre>	
<ul style="list-style-type: none"> ✓ Designers – Roger Needham David Wheeler . ✓ First published – 1994 ✓ Successors – XXTEA ✓ Key sizes – 128 bits ✓ Block sizes – 64 bits ✓ Structure – Feistel network ✓ Rounds – variable; recommended 64 Feistel rounds (32 cycles) 	<ul style="list-style-type: none"> ✓ TEA는 동일한 키 (Kelsey et al., 1996)를 갖고 223 개의 선택된 평문과 232의 시간 복잡성을 요구하는 관련 키 공격을 사용하여 손상 될 수 있음. ✓ 표준 단일 비밀 키 설정에서 TEA의 최상의 구조적 암호 해독은 전체 Code book 보다 적은 2121.5 시간에 21 라운드를 돌파하는 무결성 암호 해독임. 	

1.2 BaroCRYPT 특/장점

BaroCRYPT 솔루션은 XXTEA(일명 Corrected Block TEA) 암호화 알고리즘을 기반으로 사용 가능한 RAM의 양이 최소인 레거시 하드웨어 시스템(임베디드)과 같이 극한의 제약이 있는 상황에서도 빠르게 데이터 암호화를 실행 가능한 최적의 솔루션으로 특징점은 다음과 같다.

- 작고 구현이 쉬운 블록 암호화 알고리즘으로 페이스텔 암호를 기반으로 하여 크기가 작고 빠르면서 구현이 쉬움
- 페이스텔 암호를 기반으로 한 작은 크기의 알고리즘으로 그 크기에 비해서 암호화 강도가 높음
- 알고리즘의 크기는 작지만 현존하는 가장 빠르고 안전한 알고리즘
- 다른 블록 암호화 알고리즘에 비해 구현이 용이하고 하드웨어 사양 제약 조건이 큰 환경에 적용이 용이하며 자유롭게 사용
- 64 bit(8byte)를 암호화하는 블록 암호화 알고리즘으로 128 bit(16byte) 키를 사용
- Corrected Block TEA(XXTEA)는 원래 Block TEA의 약점을 수정하기 위해 고안된 블록 암호화 알고리즘
- 자유로운 Customizing 및 다양한 응용프로그램과 연동 개발의 편의성 제공 (Java, C 언어로 된 API 연동)
- SQL 문장에서 쉽게 사용할 수 있도록 TO_ENCRYPTS(암호화), TO_DECRYPTS(복호화) 함수 제공

※ Feistel 암호란?

동일한 대치와 치환을 반복하면서 암호문이 평문으로부터 암호화되는 반복 블록 암호로 데이터 암호화 표준(DES)과 유사한 암호로서, 평문을 반씩 2개 블록으로 나누어 한쪽은 서브 키를 사용한 기능 F로 치환하고 그 결과를 다른 반쪽에서 배타적 논리합(XOR)한 다음 서로 교환한다. 이러한 과정을 각 치환마다 동일

한 패턴으로 하되 마지막 치환에서는 서로 교환하지 않는다. 암호화시 사용된 서브키는 복호화시 역으로 사용된다.

2. BaroCRYPT 연동 API

2.1 연동 API 사용 전 준비사항

BaroCRYPT 모듈은 Java(barocrypt.jar), C(libbarocrypt.so)를 기반으로 작성되었기 때문에 반드시 최신 JDK 6.x 이상이 설치되어 있어야 하며, Java 모듈을 사용하기 위한 환경설정은 다음과 같다.

① Java 환경설정(.profile)

```
export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.131.x86_64
export
CLASSPATH=$JAVA_HOME/lib/tools.jar:$JAVA_HOME/lib/classes12.jar:$JAVA_HOME/lib/barocrypt.jar
```

② Java version 확인

```
> java -version
java version "1.7.0_131"
OpenJDK Runtime Environment (rhel-2.6.9.0.el5_11-x86_64 u131-b00)
OpenJDK 64-Bit Server VM (build 24.131-b00, mixed mode)
```

2.2 BaroCRYPT 연동 API

1) C 모듈(libbarocrypt.so)

필드 또는 데이터 암복호화에 사용되는 대칭 키(64byte)를 프로그램 내부에 고정되어 있으며, Shared object(libbarocrypt.so)를 사용하기 위해서는 반드시 shared object file이 존재하는 디렉토리 (/home/baropam/crypt)를 Library path에 설정해야 한다.

```
Linux인 경우 export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/baropam/crypt
HP-UX인 경우 export SHLIB_PATH=$SHLIB_PATH:/home/baropam/crypt
AIX인 경우 export LIBPATH=$LIBPATH:/home/baropam/crypt
```

BaroCRYPT에 대한 header 파일인 barocrypt.h은 다음과 같다.

```
#ifndef BAROCRYPT_INCLUDED
#define BAROCRYPT_INCLUDED

#include <stdlib.h>

#ifdef __cplusplus
extern "C" {
#endif

/**
 * Function: baro_encrypt
 * @data: Data to be encrypted
 * Returns: Encrypted data or %NULL on failure
 *
 * Caller is responsible for freeing the returned buffer.
```

```

*/
void * baro_encrypts(const void * data);

/**
 * Function: baro_decrypt
 * @data: Data to be decrypted
 * Returns: Decrypted data or %NULL on failure
 *
 * Caller is responsible for freeing the returned buffer.
 */
void * baro_decrypts(const void * data);

#ifdef __cplusplus
}
#endif

#endif

```

① baro_encrypts 함수

- NAME
baro_encrypts
- SYNOPSIS
void * baro_encrypts(const void * data)
- DESCRIPTION
데이터를 암호화하는 함수
data: 암호화할 데이터
- RETURN VALUES
암호화한 데이터를 반환

② baro_decrypts 함수

- NAME
baro_decrypts
- SYNOPSIS
void * baro_decrypts(const void * data)
- DESCRIPTION
데이터를 복호화 하는 함수
data: 복호화할 데이터
- RETURN VALUES
복호화한 데이터를 반환

③ 데이터 암호화 사용 예

```
#include <errno.h>
```

```

#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "barocrypt.h"

int main(int argc, char *argv[]) {
    const char *source_data = argv[1];
    const char *encrypt_data;
    const char *decrypt_data;

    encrypt_data = baro_encrypts(source_data );
    decrypt_data = baro_decrypts(encrypt_data);

    printf("Source  data = [%s]\n", source_data );
    printf("encrypt data = [%s]\n", encrypt_data);
    printf("decrypt data = [%s]\n", decrypt_data);

    if (encrypt_data) free((void *)encrypt_data);
    if (decrypt_data) free((void *)decrypt_data);

    return 0;
}

```

2) Java 모듈(barocrypt.jar)

필드 또는 데이터 암호화에 사용되는 대칭 키(64byte)를 프로그램 내부에 고정되어 있으며, Java 모듈 (barocrypt.jar)을 사용하기 위해서는 반드시 barocrypt.jar file이 존재하는 디렉토리 (/home/baropam/crypt)를 포함한 Java 모듈을 Class path에 설정해야 한다.

```
export CLASSPATH=$CLASSPATH:/home/baropam/crypt/barocrypt.jar
```

① baro_encrypts 함수

- NAME
baro_encrypts
- SYNOPSIS
public static String baro_encrypts(String data)
- DESCRIPTION
데이터를 암호화 하는 함수
data: 암호화할 데이터

- RETURN VALUES
암호화된 데이터를 반환

② baro_decrypts 함수

- NAME
baro_decrypts
- SYNOPSIS
public static String baro_decrypts(String data)
- DESCRIPTION
데이터를 복호화 하는 함수
data: 복호화할 데이터
- RETURN VALUES
복호화된 데이터를 반환

③ 데이터 암복호화 사용 예

```
import barocrypt.barocrypt.*;

public static void main(String[] args) {
    try {
        String encrypt_data = baro_encrypts(args[0] );
        String decrypt_data = baro_decrypts(encrypt_data);

        System.out.println("text      = [" + args[0]      + "]);
        System.out.println("encrypt_data = [" + encrypt_data + "]);
        System.out.println("decrypt_data = [" + decrypt_data + "]);
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
    }
}
```

3) PowerBuider 모듈(barocrypt.dll)

필드 또는 데이터 암복호화에 사용되는 대칭 키(64byte)를 프로그램 내부에 고정되어 있다.

① Global External Function 선언

```
FUNCTION string baro_encrypts(string data) LIBRARY "barocrypt.dll" ALIAS FOR "baro_encrypts;ansi"
FUNCTION string baro_decrypts(string data) LIBRARY "b.dll" ALIAS FOR "baro_decrypts;ansi"
```

② 데이터 암호화 사용 예

```
string ls_source_data = "qwerqwerqwer0이종일qwerqwer";  
string ls_encript_data = "";  
string ls_decript_data = "";  
  
ls_encript_data = baro_encrypts(ls_source_data)  
MessageBox("암호화", "encript_data = [" + ls_encript_data + "]")  
  
ls_decript_data = baro_decrypts(ls_encript_data)  
MessageBox("복호화", "decript_data = [" + ls_decript_data + "]")  
  
return
```

3. BaroCRYPT 연동 API(DB)

3.1 External Procedure란?

복잡한 수식계산을 Oracle에서 제공하는 기능으로만 충분하지 않을 경우가 있는데, 이럴 경우 C나 JAVA 같은 언어로 복잡한 기능을 작성한 후 Oracle 에서는 파라미터를 넘겨서 해당 결과를 받으면 수행속도의 개선을 가져올 수 있는데, 간단히 말하자면 C언어나 VB, JAVA 등의 언어를 사용하여 SQL에서 구현하기 어렵거나 복잡한 것을 구현한 뒤 SQL에서 호출해서 사용하는 것을 말한다.

3.2 장점 및 단점

External Procedure을 사용하여 얻는 장점으로는 Java나 C의 재 활용성이 우수하다. 반면 External Procedure을 사용으로 발생하는 단점은 Session이 종료되지 않으면 extProc는 Oracle에서 메모리를 관리하는 영역이 아니라 O/S영역이기 때문에 한번 호출될 때마다 해당 Session이 종료되지 않으면 끝까지 살아남게 되어 지속적인 메모리에 남아 있게 되어 O/S의 메모리 부하가 생길 수밖에 없다.

그래서, O/S의 메모리를 최대한 줄일 수 있는 방법은 다음과 같다.

- ① Session의 수를 제한하여 O/S의 메모리 한계치를 벗어나지 않도록 조정한다.
- ② O/S에서 extProc로 생성된 것 중 오래된 Process를 Kill한다. Process를 Kill하더라도 없으면 재생성 되므로 큰 문제는 발생되지 않는다.
- ③ 애플리케이션에서 불필요하게 External Procedure를 호출하는 Function의 사용을 제거하고, 사용 완료 후 Session을 종료하여 메모리의 부하를 최소화하게 한다.

3.3 Java 모듈(barocrypts.java)

1) Java 모듈 Load

Java 모듈을 Oracle 내에 Java 공간으로 Load하는 방법은 다음과 같이 두 가지 방법이 존재한다.

- ① 컴파일된 Source만 Load

```
> loadjava -user baropam/baropam barocrypts.class
```

- ② 소스 및 컴파일 소스까지 Load

```
> loadjava -user baropam/baropam -resolve -v barocrypts.java
arguments: '-user' 'barocrypt/**' '-resolve' '-v' 'barocrypts.java'
creating: source barocrypts
loading : source barocrypts
created : CREATE$JAVA$LOB$TABLE
resolving: source barocrypts
Classes Loaded: 0
```

```

Resources Loaded: 0
Sources Loaded: 1
Published Interfaces: 0
Classes generated: 0
Classes skipped: 0
Synonyms Created: 0
Errors: 0

> loadjava -user baropam/baropam -force -resolve -verbose -synonym -grant public BaroClient.java
arguments: '-user' 'icam/**' '-force' '-resolve' '-verbose' '-synonym' '-grant' 'public' 'BaroClient.java'
creating: source BaroClient
loading : source BaroClient
granting: execute on source BaroClient to public
granting: execute on class BaroClient to public
resolving: source BaroClient
synonym : BaroClient
Classes Loaded: 0
Resources Loaded: 0
Sources Loaded: 1
Published Interfaces: 0
Classes generated: 0
Classes skipped: 0
Synonyms Created: 1
Errors: 0

```

2) 암호화 Stored Function 생성 및 확인

Oracle에 sqlplus로 접속한 뒤 TO_ENCRYPTS(암호화 함수), TO_DECRYPTS(복호화 함수)를 다음과 같이 생성 및 확인한다.

```

> sqlplus baropam/baropam

SQL*Plus: Release 11.2.0.1.0 Production on 화 9월 26 10:56:01 2017

Copyright (c) 1982, 2009, Oracle. All rights reserved.

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> CREATE OR REPLACE FUNCTION TO_ENCRYPTS (input VARCHAR2) RETURN VARCHAR2 IS
 2     LANGUAGE JAVA
 3     NAME 'barocrypts.barocrypt_encrypts(java.lang.String) return java.lang.String';
 4 /

Function created.

SQL> CREATE OR REPLACE FUNCTION TO_DECRYPTS (input VARCHAR2) RETURN VARCHAR2 IS
 2     LANGUAGE JAVA
 3     NAME 'barocrypts.barocrypt_decrypts(java.lang.String) return java.lang.String';
 4 /

Function created.

```

```

SQL> commit;

Commit complete.

SQL> SELECT OBJECT_NAME, OBJECT_TYPE, STATUS FROM USER_OBJECTS WHERE
OBJECT_TYPE LIKE 'JAVA%';

OBJECT_NAME
-----
OBJECT_TYPE          STATUS
-----
barocrypts
JAVA CLASS          VALID

barocrypts
JAVA SOURCE         VALID

```

3) 암호화 함수(TO_ENCRYPTS, TO_DECRYPTS) 테스트

```

> sqlplus baropam/baropam

SQL*Plus: Release 11.2.0.1.0 Production on 화 9월 12 11:34:24 2017

Copyright (c) 1982, 2009, Oracle. All rights reserved.

다음에 접속됨:
Oracle Database 11g Release 11.2.0.1.0 - 64bit Production

SQL> SELECT TO_ENCRYPTS('qwerqwerqwer이종일qwerqwer') FROM DUAL;

TO_ENCRYPTS('QWERQWERQWER이종일QWERQWER')
-----
BDx8KvL4xf0dHUf7LJI/edUWGwaJGGYtzYKhc5VvcHdnBArS

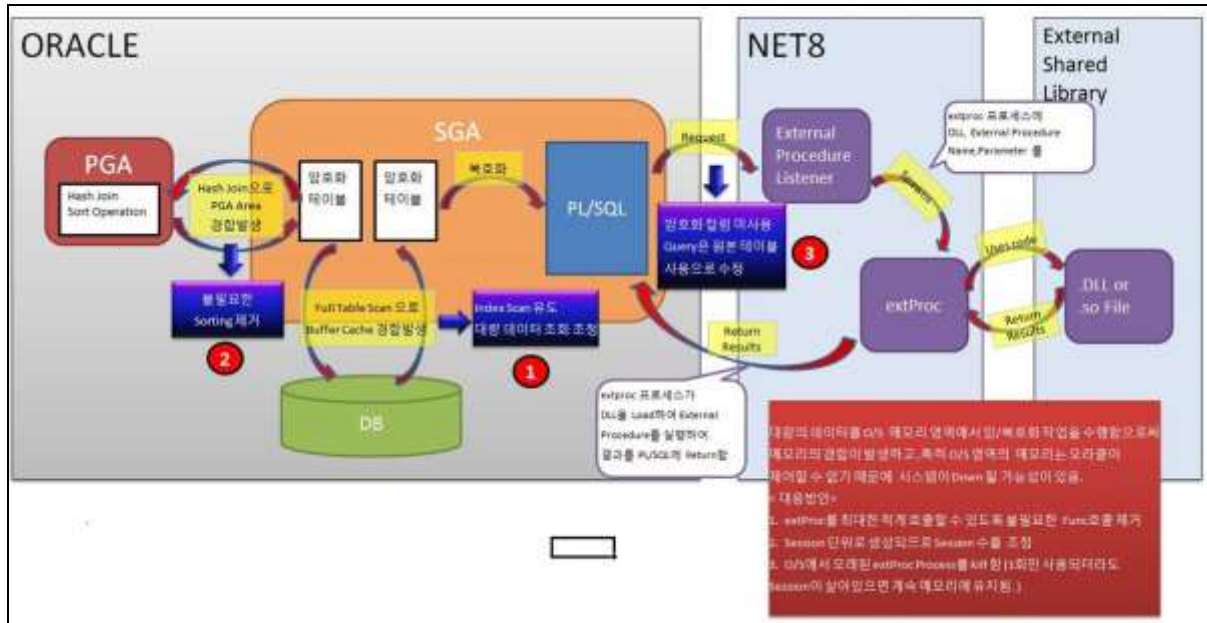
SQL> SELECT TO_DECRYPTS('BDx8KvL4xf0dHUf7LJI/edUWGwaJGGYtzYKhc5VvcHdnBArS') FROM DUAL;

TO_DECRYPTS('BDX8KVL4XF0DHUF7LJL/EDUWGWAJGGYTZYKHC5VCHDNBARS')
-----
qwerqwerqwer이종일qwerqwer

```

3.4 C 모듈(libbarocrypts.so)

C 모듈은 Java 모듈과 달리 External Procedure를 사용하는 것이 번거롭지만 External Procedure의 동작 순서는 다음과 같다.



- ① 사용자가 SQL에서 External Procedure에서 작성한 Function을 DB에 요청을 한다.
- ② Shared SQL Area에서 해당 문장을 Parsing하면서 External Procedure를 알고 NET8 Listener에서 사용자 SQL이 External Procedure를 호출했으니 해석해 달라고 요청한다.
- ③ Listener는 다시 extProc 프로세스를 생성하면서 O/S에 있는 External Procedure가 있는 DLL, Procedure Name, Parameter를 넘겨준다.
- ④ extProc를 O/S에 있는 DLL 파일을 찾아서 O/S의 메모리에 Load하여 요청받은 Function의 결과를 처리한다.
- ⑤ 처리된 결과를 SQL에 return해 준다.
- ⑥ Session이 종료되면 extproc도 자동으로 종료된다.

1) C 모듈 library 생성

C 모듈을 Oracle 내에 암호화 Library(libcrypt_encrypts-암호화 Library, libcrypt_decrypts-복호화 Library)를 다음과 같이 각각 생성한다.

```
> sqlplus baropam/baropam
SQL*Plus: Release 11.2.0.1.0 Production on 화 9월 12 14:13:40 2017
Copyright (c) 1982, 2009, Oracle. All rights reserved.
Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SQL> CREATE OR REPLACE LIBRARY libcrypt_encrypts AS '/home/baropam/crypt/libbarocrypts.so' ;
2 /
Library created.
```

```
SQL> CREATE OR REPLACE LIBRARY libcrypt_decrypts AS '/home/baropam/crypt/libbarocrypts.so' ;
2 /
```

Library created.

```
SQL> commit;
```

Commit complete.

참고) Library 생성 중 "ORA-01031: insufficient privileges" 오류가 발생하는 경우 library를 생성하는 계정 (baropam)에 library 생성 권한이 없어서 발생한다. 이런 경우 Oracle sysdba로 접속한 다음 library를 생성하는 계정에 library 생성권한을 부여해야 한다.

```
SQL> create library to baropam ;
```

2) 암호화 Stored Function 생성 및 확인

Oracle에 sqlplus로 접속한 뒤 TO_ENCRYPTS(암호화 함수), TO_DECRYPTS(복호화 함수)를 다음과 같이 생성 및 확인한다.

```
project:icam /usr/baropam> sqlplus baropam/baropam
```

```
SQL*Plus: Release 11.2.0.1.0 Production on 화 9월 12 14:16:06 2017
```

```
Copyright (c) 1982, 2009, Oracle. All rights reserved.
```

```
Connected to:
```

```
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
```

```
SQL> CREATE OR REPLACE FUNCTION TO_ENCRYPTS (input VARCHAR2) return VARCHAR2
as external
language C
library libcrypt_encrypts
name "baro_encrypts"
parameters (input STRING);
2 3 4 5 6 7 /
```

Function created.

```
SQL> CREATE OR REPLACE FUNCTION TO_DECRYPTS (input VARCHAR2) return VARCHAR2
as external
language C
library libcrypt_decrypts
name "baro_decrypts"
parameters (input STRING);
2 3 4 5 6 7 /
```

Function created.

```
SQL> commit;
```

Commit complete.

Tibero에 tsq로 접속한 뒤 TO_ENCRYPTS(암호화 함수), TO_DECRYPTS(복호화 함수)를 다음과 같이 생성 및 확인한다.

```
SQL> CREATE OR REPLACE FUNCTION TO_ENCRYPTS (input VARCHAR2) return VARCHAR2
  as language C
  library libcrypt_encrypts
  name "baro_encrypts"
  parameters (input STRING);
  2  3  4  5  6  7 /
```

Function created.

```
SQL> CREATE OR REPLACE FUNCTION TO_DECRYPTS (input VARCHAR2) return VARCHAR2
  as language C
  library libcrypt_decrypts
  name "baro_decrypts"
  parameters (input STRING);
  2  3  4  5  6  7 /
```

Function created.

SQL> commit;

Commit complete.

3) listener 구성

C 모듈을 사용하기 전에 원하는 Shared Library에 대한 경로를 포함하도록 listener를 구성해야 한다. EXTPROC_DLLS 환경 변수를 설정하여 수행하는데, Oracle 계정에 로그인 후 다음과 같이 listener.ora 파일에 설정한다.

```
SID_LIST_LISTENER =
(SID_LIST =
(SID_DESC =
(SID_NAME = PLSExtProc)
(ORACLE_HOME = /oracle/app/oracle/product/11.2.0)
(PROGRAM = extproc)
(ENVS="EXTPROC_DLLS=ANY")
)
(SID_DESC =
(SID_NAME = ODB)
(ORACLE_HOME = /oracle/app/oracle/product/11.2.0)
)
)

LISTENER =
(DESCRIPTION_LIST =
(DESCRIPTION =
(ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1521))
(ADDRESS = (PROTOCOL = TCP)(HOST = project)(PORT = 1521))
)
)
```

```
)
ADR_BASE_LISTENER = /oracle/app/oracle
```

4) tnsnames.ora 구성

C 모듈을 사용하기 전에 listener.ora 파일의 설정한 Key, SID와 동일한 값으로 Oracle 계정에 로그인 후 EXTPROC_CONNECTION_DATA를 다음과 같이 tnsnames.ora 파일에 설정한다.

```
ODB =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = project)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = ODB)
    )
  )
)

EXTPROC_CONNECTION_DATA =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = IPC)(HOST = project)(KEY = EXTPROC1521))
    (CONNECT_DATA = (SID = PLSExtProc))
  )
)
```

tnsnames.ora 파일을 설정한 후 반드시 Oracle 계정에 로그인 후 listener를 재기동 해야 한다.

```
> lsnrctl start | stop | status
```

5) 암호화 함수(TO_ENCRYPTS, TO_DECRYPTS) 테스트

```
> sqlplus baropam/baropam

SQL*Plus: Release 11.2.0.1.0 Production on 화 9월 12 14:16:06 2017

Copyright (c) 1982, 2009, Oracle. All rights reserved.

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> SELECT TO_ENCRYPTS('qwerqwerqwer이종일qwerqwer') FROM DUAL;

TO_ENCRYPTS('QWERQWERQWER이종일QWERQWER')
-----
LkrLD7uCXBnPZreic9NgsHgsDjWQG1QQL0w9UHndzy8=

SQL> SELECT TO_DECRYPTS('LkrLD7uCXBnPZreic9NgsHgsDjWQG1QQL0w9UHndzy8=') FROM DUAL;

TO_DECRYPTS('LKRLD7UCXBNPZREIC9NGSHGSDJWQG1QQL0W9UHNDZY8=')
-----
qwerqwerqwer이종일qwerqwer
```

4. About BaroCRYPT



Version 1.0 - Official Release - 2016.12.1
Copyright © Nurit corp. All rights reserved.
<http://www.nurit.co.kr>

상호: 주식회사 누리아이티
등록번호: 258-87-00901
대표이사: 이종일
대표전화: 02-2665-0119(기술지원/영업문의)
이메일: mc529@nurit.co.kr
주소: 서울시 강서구 마곡중앙2로 15, 913호(마곡동, 마곡테크노타워2)