

# BaroCRYPT(Cubrid) 가이드

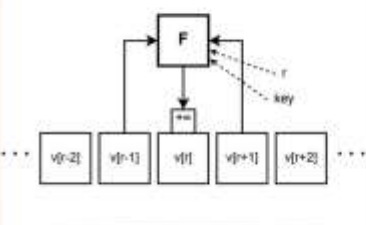
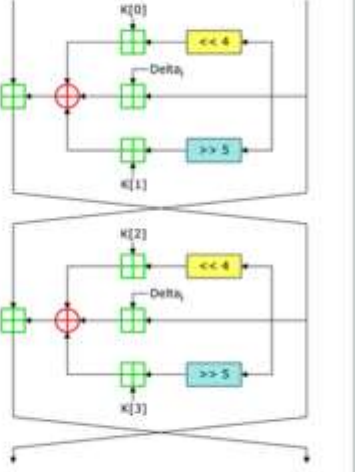
## 목차

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) .....	5
3.1 Stored Function 이란?.....	5
3.2 Java 모듈(barocrypts.class).....	5
4. About BaroCRYPT.....	8

# 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
 <ul style="list-style-type: none"> <li>✓ Designers – Roger Needham David Wheeler .</li> <li>✓ First published – 1994</li> <li>✓ Successors – XXTEA</li> <li>✓ Key sizes – 128 bits</li> <li>✓ Block sizes – 64 bits</li> <li>✓ Structure – Feistel network</li> <li>✓ Rounds – variable; recommended 64 Feistel rounds (32 cycles)</li> </ul>	<pre> void xorshift32_xorshift32_32( int *a, int *b, int *c) {   int t;   int u = *a;   int v = *b;   int w = *c;   t = (u &lt;&lt; 13) ^ (v &lt;&lt; 17);   t = (t &lt;&lt; 5) ^ w;   *a = u ^ t;   *b = v ^ t;   *c = w ^ t; }     </pre> <ul style="list-style-type: none"> <li>✓ 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.</li> <li>✓ 표준 단일 비밀 키 설정에서 TEA의 최상의 구조적 암호 해독은 전체 Code book 보다 적은 2121.5 시간에 21 라운드를 돌파하는 무결성 암호 해독임.</li> </ul>	

## 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)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:$JAVA_HOME/lib/barocrypt.jar
export LD_LIBRARY_PATH=$JAVA_HOME/jre/lib/amd64:$JAVA_HOME/jre/lib/amd64/server:$LD_LIBRARY_PATH
export PATH=$PATH:$JAVA_HOME/bin:$JAVA_HOME/jre/lib/amd64/server
```

#### ② Java version 확인

```
> java -version
java version "1.7.0_131"
OpenJDK Runtime Environment (rhel-2.6.9.0.e15_11-x86_64 u131-b00)
OpenJDK 64-Bit Server VM (build 24.131-b00, mixed mode)
```

SUN 이외의 다른 벤더가 제공하는 Java 가상 머신을 사용하는 경우, Library Path에 Java VM(libjvm.so)이 있는 경로를 추가해 주어야 한다. 이때, libjvm.so 파일의 경로는 OS 플랫폼, 지원 비트마다 다르므로 주의하여 설정한다. 예를 들어 SUN Sparc 머신에서 libjvm.so 파일의 경로는 \$JAVA\_HOME/jre/lib/sparc이다.

### 2.2 BaroCRYPT 연동 API

필드 또는 데이터 암복호화에 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  
암호화한 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  
복호화한 데이터를 반환

## ③ 데이터 암호화 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. BaroCRYPT 연동 API(DB)

#### 3.1 Stored Function이란?

저장 함수(Stored Function)를 사용하면 SQL로 구현하지 못하는 복잡한 프로그램의 로직을 구현할 수 있으며, 사용자가 보다 쉽게 데이터를 조작하게 할 수 있다. 저장 함수는 데이터를 조작하기 위해 실행 명령의 흐름이 있고, 쉽게 조작할 수 있고, 관리할 수 있는 블록 단위라고 할 수 있다.

Cubrid는 Java로 저장 함수를 개발할 수 있도록 지원한다. Java 저장 함수는 Cubrid에서 호스팅한 Java 가상 머신(JVM, Java Virtual Machine)에서 실행된다.

Java 저장 함수는 SQL에서도 호출할 수 있으며, JDBC를 사용하여 쉽게 Java 응용 프로그램에서 호출할 수 있다.

Java 저장 함수를 사용할 때 얻을 수 있는 이점은 다음과 같다.

- 생산성과 사용성  
Java 저장 함수는 한번 만들어 놓으면 계속해서 사용할 수 있다. 사용자가 저장 함수를 SQL에서도 호출하여 사용할 수 있고, JDBC를 사용하여 쉽게 Java 응용 프로그램에서 호출할 수 있다.
- 뛰어난 상호 운용성, 이식성  
Java 저장 함수는 Java 가상 머신을 사용하므로, 시스템에 Java 가상 머신을 사용할 수만 있다면 언제 어디서나 사용할 수 있다.

#### 3.2 Java 모듈(barocrypts.class)

##### 1) cubrid.conf 파일 확인

Cubrid 설치 위치로 이동하여 **cubrid.conf** 파일의 **java\_stored\_procedure**의 설정값이 **yes**로 되어 있는지 확인하고, **no**로 되어 있다면 **yes**로 설정하고 저장 후 닫는다. 기본값은 **no**이다. (cubrid.conf 파일 위치는 CUBRID 설치위치/conf 폴더 안에 있음)

##### 2) loadjava 유틸리티

컴파일된 Java 파일이나 JAR(Java Archive) 파일을 CUBRID로 로드하기 위해서 loadjava 유틸리티를 사용한다. loadjava 유틸리티를 사용하여 Java \*.class 파일이나 \*.jar 파일을 로드하면 해당 파일이 해당 데이터베이스 경로로 이동한다.

```
$ loadjava [option] database-name java-class-file
```

**database-name:** Java 파일을 로드하려고 하는 데이터베이스 이름

**java-class-file:** 로드하려는 Java 클래스 파일 이름 또는 jar 파일 이름

**option:** **-y**는 이름이 같은 클래스 파일이 존재하면 자동으로 덮어쓰기 한다. 기본값은 **no**이다. 만약 **-y** 옵션을 명시하지 않고 로드할 때 이름이 같은 클래스 파일이 존재하면 덮어쓰기를 할 것인지 묻는다.

##### 3) Java 모듈 Load

Java 모듈을 Cubrid 내에 Java 공간으로 Load하는 방법은 다음과 같다.

```
> loadjava -y sbpdb barocrypts.class
```

#### 4) 암호화 Stored Function 생성 및 확인

SQL 창에 즉 Cubrid 매니저 질의모드에 들어가서 생성해 주면 된다. 생성 후 commit 해야 한다.

```
CREATE FUNCTION 함수명(파라미터명 파라미터타입)
RETURN 리턴타입
AS LANGUAGE 언어명
NAME '클래스명.함수명(파라미터타입) return 리턴타입';
```

SQL 창에 즉 Cubrid 매니저 질의모드 TO\_ENCRYPTS(암호화 함수), TO\_DECRYPTS(복호화 함수)를 다음과 같이 생성 및 확인한다.

```
CREATE OR REPLACE FUNCTION TO_ENCRYPTS (data String)
RETURN String
AS LANGUAGE JAVA
NAME 'barocrypts.barocrypt_encrypts(java.lang.String) return java.lang.String';

CREATE OR REPLACE FUNCTION TO_DECRYPTS (data String)
RETURN String
AS LANGUAGE JAVA
NAME 'barocrypts.barocrypt_decrypts(java.lang.String) return java.lang.String';

commit;

SELECT * FROM db_stored_procedure WHERE sp_type = 'FUNCTION' AND LANG = 'JAVA' ;
```

sp_name	sp_type	return_type	arg_count	lang	target	owner
'TO_ENCRYPTS'	'FUNCTION'	'STRING'	1	'JAVA'		
'barocrypts.barocrypt_encrypts(java.lang.String) return java.lang.String'						'DBA'
'TO_DECRYPTS'	'FUNCTION'	'STRING'	1	'JAVA'		
'barocrypts.barocrypt_decrypts(java.lang.String) return java.lang.String'						'DBA'

등록된 Java 저장 함수 정보는 **db\_stored\_procedure** 시스템 가상 클래스와 **db\_stored\_procedure\_args** 시스템 가상 클래스에서 확인할 수 있다. **db\_stored\_procedure** 시스템 가상 클래스에서는 저장 함수의 이름과 타입, 반환 타입, 인자의 수, Java 클래스에 대한 명세, Java 저장 함수의 소유자에 대한 정보를 확인할 수 있다. **db\_stored\_procedure\_args** 시스템 가상 클래스에서는 저장 함수에서 사용하는 인자에 대한 정보를 확인할 수 있다.

#### 5) 암호화 함수(TO\_ENCRYPTS, TO\_DECRYPTS) 테스트

```
SELECT TO_ENCRYPTS('qwerqwerqwer이종일qwerqwer') FROM db_root;
```

```
TO_ENCRYPTS('QWERQWERQWER이종일QWERQWER')
```

```
BDx8KvL4xf0dHUf7LJI/edUWGwaJGGYtzYKhc5VvcHdnBArS
```

```
SELECT TO_DECRYPTS('BDx8KvL4xf0dHUf7LJl/edUWGwaJGGYtzYKhc5VvcHdnBARS') FROM db_root;  
  
TO_DECRYPTS('BDX8KVL4XF0DHUF7LJL/EDUWGWAJGGYTZYKHC5VCHDNBARS')  
-----  
qwerqwerqwer 이종일qwerqwer
```

## 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](mailto:mc529@nurit.co.kr)  
주소: 서울시 강서구 마곡중앙2로 15, 913호(마곡동, 마곡테크노타워2)