

# BaroCRYPT Guide(Cubrid)

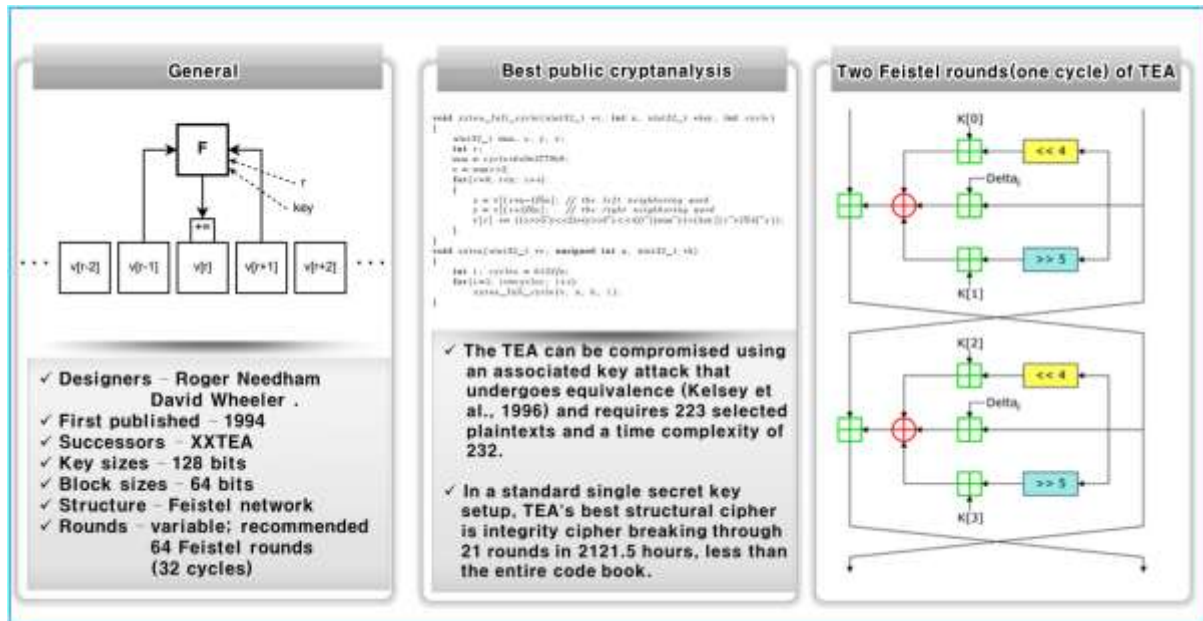
## Index

Index .....	<b>0</b>
1. BaroCRYPT .....	<b>1</b>
1.1 BaroCRYPT Overview.....	1
1.2 BaroCRYPT Features/Benefits .....	1
2. BaroCRYPT Integration API .....	<b>3</b>
2.1 Preparations before using the interlocking API .....	3
2.2 BaroCRYPT Integrate API .....	3
3. BaroCRYPT Integrate API(DB).....	<b>5</b>
3.1 What is a Stored Function?.....	5
3.2 Java module (barocrypts.class).....	5
4. About BaroCRYPT.....	<b>8</b>

# 1. BaroCRYPT

## 1.1 BaroCRYPT Overview

The BaroCRYPT solution is a lightweight and fastest encryption algorithm based on the XXTEA (Extended Extended Tiny Encryption Algorithm), a compact and easy-to-implement block encryption algorithm using the Feistel cipher.



## 1.2 BaroCRYPT Features/Benefits

Based on the XXTEA (aka Corrected Block TEA) encryption algorithm, the BaroCRYPT solution is an optimal solution capable of quickly encrypting and decrypting data even under extreme constraints such as legacy hardware systems (embedded) with a minimum amount of usable RAM. Is as follows.

- It is a small and easy-to-implement block encryption algorithm based on the Feistel cipher, which is small in size, fast and easy to implement.
- It is a small-sized algorithm based on the Feistel cipher, and has high encryption strength compared to its size.
- Although the size of the algorithm is small, it is the fastest and safest algorithm in existence.
- Compared to other block encryption algorithms, it is easy to implement, easy to apply to environments with large hardware specification constraints, and freely used.
- It is a block encryption algorithm that encrypts 64 bits (8 bytes) and uses a 128 bit (16 byte) key.
- Corrected Block TEA (XXTEA) is a block cipher algorithm originally designed to correct the weakness of Block TEA
- Provides free customizing and convenience of interlocking development with various application programs. (API integration in Java and C languages)
- TO\_ENCRYPTS (encryption) and TO\_DECRYPTS (decryption) functions are provided for easy use in

SQL statements.

※ What is a Feistel Cipher?

It is a repetitive block cipher in which the ciphertext is encrypted from the plaintext while repeating the same substitution and substitution. It is a cipher similar to the Data Encryption Standard (DES). The other halves do an exclusive OR (XOR) and then swap each other. Do this process in the same pattern for each permutation, but do not exchange each other in the last permutation. The subkey used during encryption is reversed during decryption.

## 2. BaroCRYPT Integration API

### 2.1 Preparations before using the interlocking API

Since the BaroCRYPT module is written based on Java (barocrypt.jar) and C (libbarocrypt.so), the latest JDK 6.x or higher must be installed, and the environment settings for using the Java module are as follows.

#### ① Java environment environment settings(.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 check

```
> 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)
```

When using a Java virtual machine provided by a vendor other than SUN, the path where the Java VM (libjvm.so) is located must be added to the Library Path. At this time, the path of the libjvm.so file is different for each OS platform and support bit, so set it carefully. For example, on a SUN Sparc machine, the path to the libjvm.so file is \$JAVA\_HOME/jre/lib/sparc.

### 2.2 BaroCRYPT Integrate API

The symmetric key (64 bytes) used for field or data encryption/decryption is fixed inside the program. To use the Java module (barocrypt.jar), the Java module including the directory where the barocrypt.jar file exists (/home/baropam/crypt) must be set in the class path.

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

#### ① baro\_encrypts function

- NAME

baro\_encrypts

- SYNOPSIS

```
public static String baro_encrypts(String data)
```

- DESCRIPTION

A function that encrypts data.  
data: data to encrypt

- RETURN VALUES

return encrypted data

## ② baro\_decrypts function

### - NAME

baro\_decrypts

### - SYNOPSIS

```
public static String baro_decrypts(String data)
```

### - DESCRIPTION

A function to decrypt data.  
data: data to decrypt

### - RETURN VALUES

Return the decrypted data

## ③ Example of using data encryption/decryption

```
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 Integrate API(DB)

### 3.1 What is a Stored Function?

Using Stored Functions, you can implement complex program logic that cannot be implemented with SQL, and allow users to manipulate data more easily. A stored function can be said to be a block unit that has a flow of execution commands to manipulate data, can be easily manipulated, and can be managed.

Cubrid supports the development of stored functions in Java. Java stored functions are executed in the Java Virtual Machine (JVM) hosted by Cubrid.

Java stored functions can also be called from SQL. It can be easily called from Java applications using JDBC.

The benefits of using Java stored functions include.

- Productivity and usability  
Once Java stored functions are created, they can be used over and over again. The user can call the stored function in SQL and use it. It can be easily called from Java applications using JDBC.
- Excellent interoperability and portability  
Java stored functions use the Java Virtual Machine, so you can use them anywhere and anytime as long as your system has a Java Virtual Machine available.

### 3.2 Java module (barocrypts.class)

#### 1) Check the cubrid.conf file

Move to the Cubrid installation location and check if the setting value of **java\_stored\_procedure** in the **cubrid.conf** file is **yes**, and if it is **no**, set it to **yes**, save and close. Default is no. (The location of the cubrid.conf file is in the CUBRID installation location/conf folder)

#### 2) loadjava utility

The loadjava utility is used to load compiled Java files or JAR (Java Archive) files into CUBRID. When you load a Java \*.class file or \*.jar file using the loadjava utility, the file is moved to the appropriate database path.

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

**database-name:** Database name from which to load the Java file

**java-class-file:** Java class file name or jar file name to load

**option:** **-y** automatically overwrites a class file with the same name if it exists. Default is **no**.  
When loading without specifying the **-y** option, if a class file with the same name exists, whether to overwrite it or not is asked.

### 3) Java module Load

The method to load a Java module into the Java space within Cubrid is as follows.

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

### 4) Creation and confirmation of encryption stored function

You can create it by entering the query mode of the Cubrid manager in the SQL window. You must commit after creation.

```
CREATE FUNCTION function name(parameter name parameter type)
RETURN return type
AS LANGUAGE language name
NAME 'class name.function name(parameter type) return return type';
```

Create and check TO\_ENCRYPTS (encryption function) and TO\_DECRYPTS (decryption function) in the SQL window, that is, Cubrid manager query mode as follows.

```
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'

Registered Java stored function information can be checked in the **db\_stored\_procedure** system virtual class and **db\_stored\_procedure\_args** system virtual class. In the **db\_stored\_procedure** system virtual class, you can check the name and type of the stored function, the return type, the number of arguments, the Java class specification, and the owner of the Java stored function. In the **db\_stored\_procedure\_args** system virtual class, information about the arguments used in stored functions can be checked.

### 5) Encryption function (TO\_ENCRYPTS, TO\_DECRYPTS) test

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

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

```
BDx8KvL4xf0dHuf7LJI/edUMGwaJGGYtzYKhc5VvcHdnBARs
```

```
SELECT TO_DECRYPTS('BDx8KvL4xf0dHuf7LJI/edUMGwaJGGYtzYKhc5VvcHdnBARs') FROM db_root;
```

```
TO_DECRYPTS('BDX8KVL4XF0DHUF7LJL/EDUMGWAJGGYTZYKHC5VCHDNBARS')
```

```
qwerqwerqwer이종일qwerqwer
```



## 4. About BaroCRYPT



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

Company: Nurit Co., Ltd.  
Registration Number: 258-87-00901  
CEO: Jongil Lee  
Tel: +8210-2771-4076(Technical support, sales inquiry)  
email: [mc529@nurit.co.kr](mailto:mc529@nurit.co.kr)  
Address: #913, 15, Magokjungang 2-ro, Gangseo-gu, Seoul (Magok-dong, Magok Techno Tower 2)