wolfCrypt JCE Provider and JNI Wrapper Manual

2023-03-22

Contents

1 イントロダクション

JCE (Java Cryptography Extension) フレームワークは、カスタムの暗号化サービス プロバイダーのインストールをサポートします。この仕組みにより、Java セキュリティ API によって使用される基本的な暗号化機能のサブセットを実装できます。

このドキュメントでは、wolfCrypt JCE プロバイダーの詳細と使用方法について説明します。wolfCrypt JCE プロバイダー (wolfJCE) は、ネイティブの wolfCrypt 暗号化ライブラリーをラップして、Java Security API との互換性を確保します。こちらの Github リポジトリを参照してください。

wolfcrypt-jni パッケージには、JCE プロバイダーに加えて、wolfCrypt JNI ラッパーの両方が含まれています。JNI ラッパーは、必要に応じて単独で使用できます。# システム要件

1.1 Java / JDK

wolfJCE では、ホストシステムに Java をインストールする必要があります。Oracle JDK や OpenJDK など、ユーザーや開発者が利用できる JDK バリアントがいくつかあります。wolfJCE は現在、OpenJDK、Oracle JDK、および Android でテストされています。OpenJDK と Android では、JCE プロバイダーがコード署名されている必要はありませんが、Oracle JDK では必要です。コード署名の詳細については、第 7 章を参照してください。

参考までに、wolfJCE がテストされた OpenJDK の特定のバージョンは次のとおりです：

```
$ java -version
Openjdk version “ 1.8.0_91 ”
OpenJDK Runtime Environment (build 1.8.0_91-8u91-b14-3ubuntu1~15.10.1~b14)
OpenJDK 64-Bit Server VM (build 25.91-b14, mixed mode)
```

また、Oracle JDK 1.8.0_121 および Android 24 でもテストされています。

1.2 JUnit

単体テストを実行するには、JUnit が開発システムにインストールされている必要があります。JUnit は、プロジェクトの Web サイト (www.junit.org) からダウンロードできます

Unix/Linux/OSX システムに JUnit をインストールするには: 1. junit.org/junit4/ から “**junit-4.13.jar**” と “**hamcrest-all-1.3.jar**” をダウンロードします。執筆時点では、前述の.jar ファイルは次のリンクからダウンロードできます:

リンク: junit-4.13.jar リンク: hamcrest-all-1.3.jar

2. これらの JAR ファイルをシステムに配置し、その場所を指すように JUNIT_HOME を設定します：

```
$ export JUNIT_HOME=/path/to/jar/files
```

1.3 make と ant

“make” と “ant” は、それぞれネイティブ C コードと Java コードのコンパイルに使用されます。

これらが開発マシンにインストールされていることを確認してください。

1.4 wolfSSL / wolfCrypt ライブラリ

ネイティブ wolfCrypt ライブラリのラッパーとして、wolfSSL をホスト プラットフォームにインストールし、インクルードおよびライブラリ検索パスに配置する必要があります。wolfJCE は、wolfSSL/wolfCrypt ネイティブ ライブラリの FIPS または非 FIPS バージョンに対してコンパイルできます。

1.4.1 wolfSSL / wolfCrypt のコンパイル

wolfJCE で使用するために Unix/Linux 環境で wolfSSL をコンパイルおよびインストールするには、wolfSSL マニュアルのビルド手順に従ってください。wolfSSL をコンパイルする最も一般的な方法は、Autoconf システムを使用することです。

wolfSSL (wolfssl-x.x.x)、wolfSSL FIPS リリース (wolfssl-x.x.x-commercial-fips)、または wolfSSL FIPS Ready リリースをインストールできます。いずれの場合も、./configure スクリプト実行時に`--enable-keygen` オプションが必要です。

wolfSSL 標準ビルド:

```
$ cd wolfssl-x.x.x
$ ./configure --enable-keygen
$ make check
$ sudo make install
```

wolfSSL FIPSv1 ビルド:

```
$ cd wolfssl-x.x.x-commercial-fips
$ ./configure --enable-fips --enable-keygen
$ make check
$ sudo make install
```

wolfSSL FIPSv2 ビルド:

```
$ cd wolfssl-x.x.x-commercial-fips
$ ./configure --enable-fips=v2 --enable-keygen
$ make check
$ sudo make install
```

wolfSSL FIPS Ready ビルド:

```
$ cd wolfssl-x.x.x-commercial-fips
$ ./configure --enable-fips=ready --enable-keygen
$ make check
$ sudo make install
```

これにより、システムのデフォルトのインストールロケーションに wolfSSL ライブラリがインストールされます。多くのプラットフォームでは、これは次の場所になっています：

```
/usr/local/lib
/usr/local/include
```


2 コンパイル

このセクションの手順を実行する前に、上記の第 2 章 の依存モジュールがインストールされていることを確認してください。

最初に、Linux または OSX のどちらを使用しているかに応じて、システムに適切な “makefile” をコピーします。

Linux を使用している場合:

```
$ cd wolfcrypt-jni
$ cp makefile.linux makefile
```

Mac OSX にインストールする場合:

```
$ cd wolfcrypt-jni
$ cp makefile.macosx makefile
```

次に、“make” を使用してネイティブ (C ソース) コードをコンパイルします:

```
$ cd wolfcrypt-jni
$ make
```

Java ソースのコンパイルには “ant” を使用します。JNI または JCE (JNI を含む) パッケージをデバッグ モードまたはリリース モードでコンパイルするための ant ターゲットがいくつかあります。ターゲットを指定せずに “ant” を実行すると、使用オプションが表示されます：

```
$ ant
...
build:
[echo] wolfCrypt JNI and JCE
[echo]
    ------------------------------------------------------------------------

[echo] USAGE:
[echo] Run one of the following targets with ant:
[echo] build-jni-debug | builds debug JAR with only wolfCrypt JNI classes
[echo] build-jni-release | builds release JAR with only wolfCrypt JNI classes
[echo] build-jce-debug | builds debug JAR with JNI and JCE classes
[echo] build-jce-release | builds release JAR with JNI and JCE classes
[echo]
    ------------------------------------------------------------------------
```

必要に応じてビルドターゲットを指定してください。たとえば、リリース モードで wolfJCE プロバイダーをビルドする場合は、次を実行します：

```
$ ant build-jce-release
```

また、JUnit テストを実行するには、次のコマンドを実行します。これにより、実行されたビルド (JNI と JCE) に一致するテストのみがコンパイルされ、それらのテストも実行されます。

```
$ ant test
```

Java JAR とネイティブ ライブラリの両方を消去するには:

```
$ ant clean
$ make clean
```


2.1 API マニュアル（Javadoc）

ant を実行すると、wolfcrypt-jni/docs ディレクトリの下に一連の Javadoc が生成されます。ルートドキュメントを表示するには、Web ブラウザで次のファイルを開きます：

```
wolfcrypt-jni/docs/index.html
```

3 インストール

wolfJCE をインストールして使用するには、次の 2 つの方法があります：

3.1 実行時インストール

実行時に wolfJCE をインストールして使用するには、まず “**libwolfcryptjni.so**” がシステムのライブラリ検索パスにあることを確認してください。Linux では、このパスを次のようにして変更できます：

```
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/path/to/add
```

次に、wolfCrypt JNI / wolfJCE JAR ファイル (**wolfcrypt-jni.jar**) を Java クラスパスに配置します。これを行うには、システムのクラスパス設定を調整するか、コンパイル時と実行時に次のようにします：

```
$ javac -classpath <path/to/jar> ...
$ java -classpath <path/to/jar> ...
```

最後に、Java アプリケーションで、プロバイダークラスをインポートし、Security.addProvider() を呼び出して、実行時にプロバイダーを追加します：

```
import com.wolfssl.provider.jce.WolfCryptProvider;
public class TestClass {
   public static void main(String args[]) {
      ...
      Security.addProvider(new WolfCryptProvider());
      ...
   }
}
```

検証のためにインストールされているすべてのプロバイダーのリストを出力するには、次のようにします：

```
Provider[] providers = Security.getProviders()
for (Provider prov:providers) {
   System.out.println(prov);
}
```

3.2 OS / システムレベルでのインストール

システム レベルで wolfJCE プロバイダーをインストールするには、JAR を OS の正しい Java インストールディレクトリにコピーし、共有ライブラリがライブラリ検索パスにあることを確認します。

wolfJCE JAR ファイル (**wolfcrypt-jni.jar**) と共有ライブラリ (**libwolfcryptjni.so**) を次のディレクトリに追加します：

```
$JAVA_HOME/jre/lib/ext directory
```

たとえば、OpenJDK を使用する Ubuntu では、次のようになります：

```
/usr/lib/jvm/java-8-openjdk-amd64/jre/lib/ext
```

さらに、次のようなエントリを java.security ファイルに追加します：

```
security.provider.N=com.wolfssl.provider.jce.WolfCryptProvider
```

ava.security ファイルは次の場所にあります：

```
$JAVA_HOME/jre/lib/security/java.security
```

“N” を、ファイル内の他のプロバイダーと比較して WolfCryptProvider に持たせたい優先順位に置き換えます。

4 パッケージ構成

wolfJCE は、“wolfcrypt-jni” JNI ラッパー ライブラリにバンドルされています。wolfJCE は wolfCrypt の基礎となる JNI バインディングに依存するため、wolfcrypt-jni と同じネイティブ ライブラリ ファイルと JAR ファイルにコンパイルされます。

JNI ラッパーのみを使用したいユーザーの場合、JCE プロバイダー クラスを含まないバージョンの “wolfcrypt-jni.jar” をコンパイルすることができます。

wolfJCE / wolfCrypt JNI パッケージ構造:

```
wolfcrypt-jni /
AUTHORS
build.xml                           ant ビルドスクリプト
COPYING
docs /                              Javadoc
jni /                               ネイティブC JNI バインディングソースファイル
lib /                               コンパイル成果物（ライブラリ）の出力先
LICENSING
Makefile generic                    Makefile
Makefile.linux                      Linux用  Makefile
Makefile.osx                        OSX用 Makefile
README_JCE.md
README.md
src /
    main/java/                      Java ソースファイル
    test/java/                      テストソースファイル
```

wolfJCE プロバイダーのソース コードは “src/main/java/com/wolfssl/provider/jce” ディレクトリにあり、“**com.wolfssl.provider.jce**” Java パッケージの一部です。

wolfCrypt JNI ラッパーは “src/main/java/com/wolfssl/wolfcrypt” ディレクトリにあり、“**com.wolfssl.wolfcrypt**” Java パッケージの一部です。このパッケージは wolfJCE クラスによって使用されるため、JCE のユーザーはこのパッケージを直接使用する必要はありません。

wolfCrypt-JNI と wolfJCE がコンパイルされると、出力 JAR とネイティブ共有ライブラリが “./lib” ディレクトリに配置されます。これらには、JCE ビルドがコンパイルされると、wolfCrypt JNI ラッパーと wolfJCE プロバイダーの両方が含まれることに注意してください。

```
lib/
    libwolfcryptjni.so
    wolfcrypt-jni.jar
```

5 サポートしているアルゴリズムとクラス

wolfJCE は現在、次のアルゴリズムとクラスをサポートしています:

```
MessageDigest Class
    MD5
    SHA-1
    SHA-256
    SHA-384
    SHA-512

SecureRandom Class
    HashDRBG

Cipher Class
    AES/CBC/NoPadding
    DESede/CBC/NoPadding
    RSA/ECB/PKCS1Padding

Mac Class
    HmacMD5
    HmacSHA1
    HmacSHA256
    HmacSHA384
    HmacSHA512

Signature Class
    MD5withRSA
    SHA1withRSA
    SHA256withRSA
    SHA384withRSA
    SHA512withRSA
    SHA1withECDSA
    SHA256withECDSA
    SHA384withECDSA
    SHA512withECDSA

KeyAgreement Class
    DiffieHellman
    DH
    ECDH

KeyPairGenerator Class
    EC
    DH
```

6 JAR コードの署名

Oracle JDK/JVM では、Oracle が発行したコード署名証明書によって JCE プロバイダーが署名されている必要があります。wolfcrypt-jni パッケージの ant ビルド スクリプトは、コンパイル前にパッケージ ディレクトリのルートにカスタム プロパティ ファイルを配置することにより、生成された “wolfcrypt-jni.jar” ファイルのコード署名をサポートします。

自動コード署名を有効にするには、“codeSigning.properties” というファイルを作成し、” wolfcrypt-jni” パッケージのルートに配置します。これは、以下を含むプロパティ ファイルです:

```
sign.alias=<signing alias in keystore>
sign.keystore=<path to signing keystore>
sign.storepass=<keystore password>
sign.tsaurl=<timestamp server url>
```

“ant” または “ant test” の実行時にこのファイルが存在する場合、提供されたキーストアとエイリアスを使用して “wolfcrypt-jni.jar” に署名します。

6.1 事前署名 JAR ファイルの利用

wolfSSL 社 は、Oracle JDK で wolfJCE を認証できるようにする、Oracle からの独自のコード署名証明書のセットを持っています。wolfJCE のリリースごとに、wolfSSL は、次の場所にある “wolfcrypt-jni.jar” の署名済みバージョンをいくつかリリースしています：

wolfcrypt-jni-X.X.X/lib/signed/debug/wolfcrypt-jni.jar wolfcrypt-jni-X.X.X/lib/signed/release/wolfcrypt-jni.jar

この事前に署名された JAR は、Java ソース ファイルを再コンパイルすることなく、JUnit テストで使用できます。この JAR ファイルに対して JUnit テストを実行するには:

```
$ cd wolfcrypt-jni-X.X.X
$ cp ./lib/signed/release/wolfcrypt-jni.jar ./lib
$ ant test
```


7 使用方法

使用方法については、上記の 第 6 章 で指定されているクラスの Oracle/OpenJDK Javadoc に従ってください。java.security ファイルで同じアルゴリズムを提供する他のプロバイダーよりも優先順位が低く設定されている場合は、“wolfJCE” プロバイダーを明示的に要求する必要があることに注意してください。たとえば、SHA-1 の MessageDigest クラスで wolfJCE プロバイダーを使用するには、次のように MessageDigest オブジェクトを作成します

```
MessageDigest md = MessageDigest.getInstance(“SHA-1”, “wolfJCE”);
```

ご質問やご意見がございましたら、support@wolfssl.com までメールでお寄せください。