WV Modular DRM Security Integration Guide for Common Encryption (CENC)

Android Supplement

Version 6

Revision History

Version	Date	Description	Author
1	3/4/2013	Initial revision	Jeff Tinker, Fred Gylys-Colwell, Edwin Wong, Rahul Frias, John Bruce
2	3/14/2013	Added RSA Certificate Provisioning	Jeff Tinker, Fred Gylys-Colwell
4	4/2/2013	Added Generic Modular DRM	Jeff Tinker, Fred Gylys-Colwell
5	4/3/2013	Updated Testing section	Edwin Wong
6	4/5/2013	Refactored common information into *Widevine Modular DRM Security Integration Guide for CENC.* This document is now the *Android Supplement*.	Jeff Tinker

Table of Contents

Terms and Definitions

Device Id — A null-terminated C-string uniquely identifying the device. 32 character maximum, including NULL termination.

Device Key — 128-bit AES key assigned by Widevine and used to secure entitlements.

Keybox — Widevine structure containing keys and other information used to establish a root of trust on a device. The keybox is either installed during manufacture or in the field. Factory provisioned devices have a higher level of security and may be approved for access to higher quality content.

Provision — Install a Keybox that has been uniquely constructed for a specific device.

Trusted Execution Environment (TEE) — The portion of the device that contains security hardware and prevents access by non secure system resources.

References

Widevine Security Integration Guide for Android-based Devices

Widevine Modular DRM Security Integration Guide for Common Encryption (CENC)

Android DRM API for DASH

DASH - 23009-1 MPD and Segment Formats

DASH - 14496-12 ISO BMFF Amendment

DASH - 23001-7 ISO BMFF Common Encryption

Audience

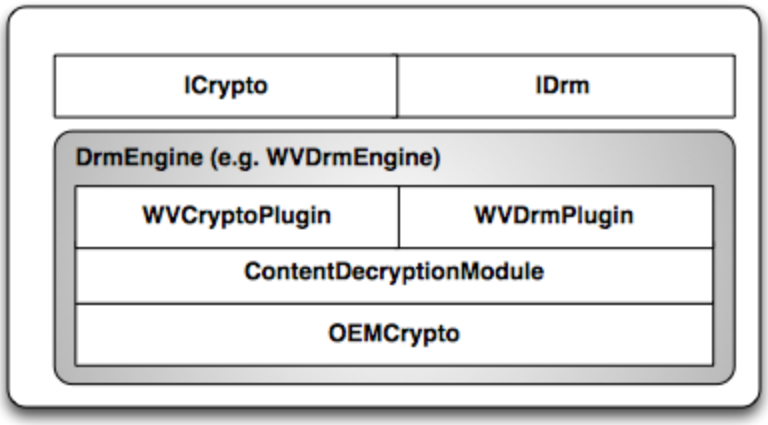
This document is intended for SOC and OEM device manufacturers to integrate with Widevine content protection on android devices.

Purpose

This document defines steps required to build a Widevine DrmEngine component for an android device, and the required functionality of the OEM-provided OEMCrypto library. It contains Android-specific supplemental information for the common document *Widevine Modular DRM Security Integration Guide for Common Encryption (CENC).*

Widevine DrmEngine

The Widevine DrmEngine implements the MediaDrm and Crypto APIs to support content decryption in support of the Android MediaCodec and MediaCrypto APIs. Refer to the "Android DRM API for DASH" document to learn more about how the DrmEngine interacts with these higher level APIs.



The OEMCrypto API defines a hardware abstraction layer to enable the Widevine DrmEngine functionality to be adapted to the underlying hardware feature set.

The remainder of this document defines the OEMCrypto APIs and steps required to build and test the vendor-supplied OEMCrypto implementation library liboemcrypto.so required by the Widevine DrmEngine component on android devices.

Deliverables

The OEMCrypto API implementation should be performed by the vendor. The API is to be implemented in the shared library liboemcrypto.so, which should be placed in /vendor/lib on the device.

Additional Requirements

In the document **Widevine Modular DRM Security Integration Guide for Common Encryption (CENC),** several features are listed as optional. The Session Usage Table is not optional for Android devices. An Android device will not pass GTS testing as a Level 1 device unless it can support the Session Usage Table API.

Unit and Integration Testing

A unit test validates a single piece of functionality, in isolation from the rest of the system. The unit test class typically contains unit tests for all of the methods of a single C++ source file. An integration test combines various components and tests the system as a whole.

A number of unit and integration tests are provided for vendors to verify the basic functions of their implementation. The tests utilize Google C++ Testing Framework, which can be found in the Android tree under external/gtest. It can also be downloaded from Google C++ Testing Framework.

Because of the rapid development cycle, initial delivery source code will be sent in a tar file. You should already have an android tree. Unpack like this:

```
lunch <YOUR TARGET>
cd $ANDROID_BUILD_TOP
tar -C $ANDROID_BUILD_TOP -xzf $tarfile
```

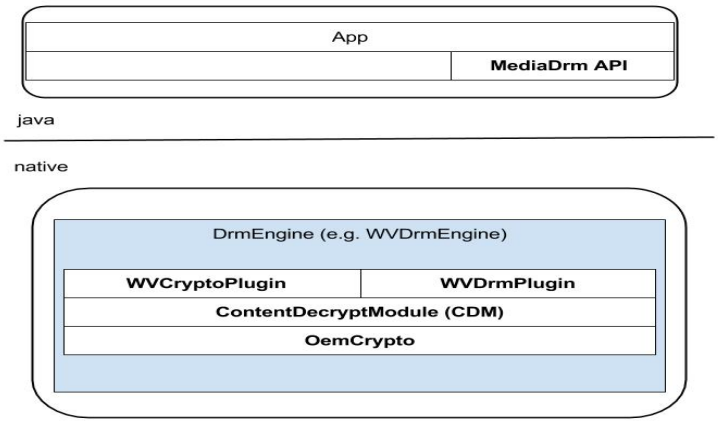
Setting Up the Build Environment

Before building any tests, please setup the build environment in the local branch

```
. build/envsetup.sh
lunch <Your TARGET>
```

Targeted Components

The tests provided verify the following emboldened components:



Testing OEMCrypto Library

The reference implementation of OEMCrypto library is in the directory $ANDROID_BUILD_TOP/vendor/widevine/libwvdrmengine/oemcrypto/mock

Out of the box, you should be able to compile existing unit tests. First build the gtest library:

```
cd $ANDROID_BUILD_TOP/external/gtest
mm
```

Build the reference implementation of oemcrypto.so:

```
cd $ANDROID_BUILD_TOP/vendor/widevine/libwvdrmengine/oemcrypto/mock
mm
```

Build the unit tests for oemcrypto.so:

```
cd $ANDROID_BUILD_TOP/vendor/widevine/libwvdrmengine/oemcrypto/test
mm
```

Run the existing unit tests:

```
cd $ANDROID_BUILD_TOP
adb root
adb remount
adb push $OUT/system/bin/oemcrypto_test /system/bin
adb push $OUT/system/vendor/lib/liboemcrypto.so /system/vendor/lib
adb shell /system/bin/oemcrypto_test
```

You should see output ending in:

```
[----------] Global test environment tear-down
[==========] 17 tests from 1 test case ran. (4104 ms total)
[  PASSED  ] 17 tests.
  YOU HAVE 135 DISABLED TESTS
```

Most of the tests are disabled by default. That is because they install a test keybox, and they

delete the usage table, which might ruin a production device. If you are using the reference implementation, or you don't mind having a test keybox installed on your device. You can run

```
adb shell /system/bin/oemcrypto_test   --gtest_also_run_disabled_tests
```

There are some tests that check timing, so it will take more than a minute or two to run. You should see output ending in:

```
…
[----------] Global test environment tear-down
[==========] 152 tests from 9 test cases ran. (114727 ms total)
[  PASSED  ] 152 tests.
```

If that works, you should be ready to start!

Remove the reference implementation liboemcrypto.so from the device.

In your own directory, build the shared library liboemcrypto.so. It should use the include file OEMCryptoCENC.h, which is found here:

```
LOCAL_C_INCLUDES := \
   vendor/widevine/libwvdrmengine/oemcrypto/include
```

Then rebuild and run the oemcrypto tests. Notice that some of the tests use a known test keybox. If your implementation does not implement OEMCryptoWrapKeyBox and OEMCrypto_InstallKeybox, then you will not be able to run these tests. WARNING: the unit tests do call OEMCrypto_InstallKeybox. **THIS MAY OVERWRITE ANY PRODUCTION KEYBOX ON THE DEVICE. DO NOT RUN THE UNIT TESTS ON A PRODUCTION DEVICE!** If you are OK with this, then pass the command line argument "--gtest_also_run_disabled_tests" when running oemcrypt_test.

Testing ContentDecryptionModule

Prerequisites: You will need libgtest.a, libgtest_main.a, libgmock.a, libgmock_main.a and liboemcrypto.so to build and run the tests.

Build the Google C++ Testing Framework static libraries(libgtest.a, libgtest_main.a).

```
cd $ANDROID_BUILD_TOP/external/gtest
mm
```

Build the gmock static libraries(libgmock.a, libgmock_main.a).

```
cd $ANDROID_BUILD_TOP/vendor/widevine/libwvdrmengine/test/gmock
mm
```

Build or supply your liboemcrypto.so. The following example builds the mock liboemcrypto.so.

```
cd $ANDROID_BUILD_TOP/vendor/widevine/libwvdrmengine/oemcrypto/mock
mm
```

Build the tests from vendor/widevine/libwvdrmengine/cdm/test.

```
cd $ANDROID_BUILD_TOP/vendor/widevine/libwvdrmengine/cdm/test
mm
```

Two of the tests are used to verify the CDM functions, they are request_license_test and cdm_engine_test.

The request_license_test uses wvcd::WvContentDecryptionModule interface. The tests include generating and sending a license request to the license server and verifying the response coming back from the server. It also performs query key status and query status tests.

The cdm_engine_test is a unit test that calls the cdm_engine directly to generate a license request and sends it to the license server, then verifies the response coming back from the server.

If your system image has not changed, you can push the newly built tests and libraries to the device using `adb sync`. You may have to run `adb root` and `adb remount` once after powering up the device.

Or you can push individual test and run it as shown below:

```
adb root
adb remount
adb push $OUT/system/lib/libwvdrmengine.so /system/vendor/lib/mediadrm

adb push $OUT/system/bin/request_license_test /system/bin
adb shell /system/bin/request_license_test

adb push $OUT/system/bin/cdm_engine_test /system/bin
adb shell /system/bin/cdm_engine_test
```

Testing Java Drm API and Plugins

The plugin tests include tests for the Java Drm API for DASH, WVCryptoPlugin, WVDrmPlugin and WVDrmPluginFactory.

Prerequisites: You will need libgtest.a, libgtest_main.a, libgmock.a and libgmock_main.a to build and run the tests.

Please refer to Testing ContentDecryptionModule for building the gtest and gmock libraries first.

These are isolated unit tests for the top level components of the DRM engine, they do not exercise the OEMCrypto API.

Build libwvdrmdrmplugin_test from vendor/widevine/libwvdrmengine/mediadrm/test.

```
cd $ANDROID_BUILD_TOP/vendor/widevine/libwvdrmengine/mediadrm/test
mm
adb push $OUT/system/bin/libwvdrmdrmplugin_test /system/bin
adb shell /system/bin/libwvdrmdrmplugin_test
```

Build libwvdrmmediacrypto_test from vendor/widevine/libwvdrmengine/mediacrypto/test.

```
cd $ANDROID_BUILD_TOP/vendor/widevine/libwvdrmengine/mediacrypto/test
mm
adb push $OUT/system/bin/libwvdrmmediacrypto_test /system/bin
adb shell /system/bin/libwvdrmmediacrypto_test
```

Build libwvdrmengine_test from vendor/widevine/libwvdrmengine/test/unit.

```
cd $ANDROID_BUILD_TOP/vendor/widevine/libwvdrmengine/test/unit
mm
adb push $OUT/system/bin/libwvdrmengine_test /system/bin
adb shell LD_LIBRARY_PATH=/system/vendor/lib/mediadrm/
/system/bin/libwvdrmengine_test
```

Build the Java Drm API for DASH test from vendor/widevine/libwvdrmengine/test/java. This is an end-to-end test that uses the MediaDrm APIs to obtain a key request, send it to the Google Play license server and load the response into the CDM, which will cause keys to be loaded into the TEE via the OEMCrypto APIs.

```
cd $ANDROID_BUILD_TOP/vendor/widevine/libwvdrmengine/test/java
mm

adb install MediaDrmAPITest.apk
```

To run this test, find the MediaDrmAPITest icon in the applications on the device and launch it.

Note that there is no UI yet, you will just see a blank screen, but you can check the logcat output and note that keys are loaded.