



Fake Finger Detection Guide

For Select SecuGen Fingerprint Readers

SG1-0201A-000 (Last updated: 2/14/2022)

Copyright © 2022 SecuGen Corporation. ALL RIGHTS RESERVED. Information in this document is subject to change without notice. The software described in this document is furnished under a license agreement or nondisclosure agreement. The software may be used only in accordance with the terms of the agreement. SecuGen is a registered trademark of SecuGen Corporation. SecuGen, Hamster IV, Hamster Plus, Hamster Pro, Hamster Pro Duo are trademarks or registered trademarks of SecuGen Corporation. All other brands or product names may be trademarks, service marks or registered trademarks of their respective owners.

Contents

Chapter 1. Overview 3

Chapter 2. Controlling Fake Finger Capabilities 5

 2.1. Fake Detection API Functions 5

 2.2. SetFakeDetectionLevel() 5

 2.3. SetGetData() 6

Chapter 1. Overview

Fake Finger Detection, sometimes referred to as Live Finger Detection, is a term used to describe one or more methods that allow a fingerprint biometric sensor to distinguish between fingerprints made from live fingers and those made from fake fingers. SecuGen uses a combination of hardware and software technologies to distinguish between live fingerprints and fake fingerprints.

All SecuGen contact fingerprint devices use a touch chip to detect finger presence. This functionality is enabled by default. When the touch chip is enabled, a finger must be placed on the fingerprint sensor to successfully capture a fingerprint image. This technology prevents the capture of latent fingerprints and fake fingerprints affixed to the platen on the device. Certain models of SecuGen contact fingerprint devices have another layer of fake finger detection capability based on liveness thresholds that can be selected by the programmer. This is accomplished by the Fake Detection Engine. The table below shows a list of SecuGen devices and their capabilities.

SecuGen Device	Touch Chip	Fake Detection Engine
Hamster IV	Yes	No
Hamster Plus	Yes	No
Hamster Pro	Yes	No
Hamster Pro 10	Yes	Yes
Hamster Pro 20	Yes	Yes
Hamster Pro Duo CL	Yes	Yes
Hamster Pro Duo SC/PIV	Yes	Yes

For SecuGen devices that have Fake Detection Engine support, the programmer can set thresholds to adjust the sensitivity of the fake detection engine. When the threshold is set higher, the percentage of fake fingers that are detected will increase. This can also result in a higher incidence of live fingers being incorrectly classified as fake fingers. The table below shows the fake finger threshold settings that are available to the programmer.

FAKE DETECTION LEVEL	Value	Setting
FAKE_DETECTION_LEVEL_0	0	No fake detection
FAKE_DETECTION_LEVEL_1	1	Touch Chip Only
FAKE_DETECTION_LEVEL_2	2	Fake Threshold 1
FAKE_DETECTION_LEVEL_3	3	Fake Threshold 2
FAKE_DETECTION_LEVEL_4	4	Fake Threshold 3
FAKE_DETECTION_LEVEL_5	5	Fake Threshold 4
FAKE_DETECTION_LEVEL_6	6	Fake Threshold 5
FAKE_DETECTION_LEVEL_7	7	Fake Threshold 6
FAKE_DETECTION_LEVEL_8	8	Fake Threshold 7
FAKE_DETECTION_LEVEL_9	9	Fake Threshold 8

Chapter 2. Controlling Fake Finger Capabilities

2.1. Fake Detection API Functions

SecuGen FDx SDK Pro on supported platforms provides the following API calls to allow the programmer to control the fake detection capabilities of the SecuGen fingerprint device:

- **SetFakeDetectionLevel()** - This function is used to simply set the fake detection threshold.
- **SetGetData()** - This function is used when a higher level of fake detection control is required. Commands are provided that allow the programmer to determine if the fake detection engine is enabled, to retrieve fake detection scores, and to set or get the current fake detection threshold.

2.2. SetFakeDetectionLevel()

Call SetFakeDetectionLevel() to set the fake detection level to a value between 0 and 9.

C/C++

```
DWORD WINAPI SetFakeDetectionLevel(int level) ;
```

Example:

```
err = sgfpplib->SetFakeDetectionLevel(FAKE_DETECTION_LEVEL_2);
```

C#

```
Int32 SetFakeDetectionLevel(Int32 level);
```

Example:

```
err = m_FPM.SetFakeDetectionLevel(FAKE_DETECTION_LEVEL_2);
```

Java

```
public long SetFakeDetectionLevel(int level);
```

Example:

```
err = sgfpplib.SetFakeDetectionLevel(FAKE_DETECTION_LEVEL_2)
```

2.3. SetGetData()

Call SetGetData() with the appropriate commands for more granular control of the fake detection capabilities of the SecuGen device you are using.

C/C++

```
DWORD WINAPI SetGetData(DWORD flag, void* data);

#define GET_FAKE_ENGINE_READY          203    // non-zero if ready
#define GET_FAKE_NUM_OF_THRESHOLD      204    // the number of thresholds available
#define GET_FAKE_THRESHOLD             205    // get current threshold index
#define SET_FAKE_THRESHOLD             206    // set new threshold index
#define GET_FAKE_THRESHOLD_VALUE       207    // get current threshold value in double
#define GET_FAKE_SCORE                 208    // get score
#define GET_FAKE_DEFAULT_THRESHOLD     211    // get default threshold index
```

COMMAND	VALUE	RETURN DATA TYPE
GET_FAKE_ENGINE_READY	203	DWORD*
GET_FAKE_NUM_OF_THRESHOLD	204	DWORD*
GET_FAKE_THRESHOLD	205	DWORD*
SET_FAKE_THRESHOLD	206	DWORD*
GET_FAKE_THRESHOLD_VALUE	207	DOUBLE*
GET_FAKE_SCORE	208	DOUBLE*
GET_FAKE_DEFAULT_THRESHOLD	211	DWORD*

C#

```
Int32 SetGetData(Int32 flag, Byte[] data);

public class FakeFingerThreshold
{
    public const Int32 GET_FAKE_ENGINE_READY          = 203; // non-zero if ready
    public const Int32 GET_FAKE_NUM_OF_THRESHOLD      = 204; // the number of thresholds available
    public const Int32 GET_FAKE_THRESHOLD             = 205; // get current threshold index
    public const Int32 SET_FAKE_THRESHOLD             = 206; // set new threshold index
    public const Int32 GET_FAKE_THRESHOLD_VALUE       = 207; // get current threshold value double
    public const Int32 GET_FAKE_SCORE                 = 208; // get score
    public const Int32 GET_FAKE_DEFAULT_THRESHOLD     = 211; // get default fake threshold index
}
```

COMMAND	VALUE	RETURN DATA TYPE
GET_FAKE_ENGINE_READY	203	Byte[8]
GET_FAKE_NUM_OF_THRESHOLD	204	Byte[8]
GET_FAKE_THRESHOLD	205	Byte[8]
SET_FAKE_THRESHOLD	206	Byte[8]
GET_FAKE_THRESHOLD_VALUE	207	Byte[16]
GET_FAKE_SCORE	208	Byte[16]
GET_FAKE_DEFAULT_THRESHOLD	211	Byte[8]

Java

```

public long SetGetData(int flag, byte[] data)

public abstract class SGSetGetData {
    public static final int GET_FAKE_ENGINE_READY      = 203;// non-zero if ready
    public static final int GET_FAKE_NUM_OF_THRESHOLD = 204;// the number of thresholds available
    public static final int GET_FAKE_THRESHOLD        = 205;// get current threshold index
    public static final int SET_FAKE_THRESHOLD        = 206;// set new threshold index
    public static final int GET_FAKE_THRESHOLD_VALUE   = 207;//get current threshold value (double)
    public static final int GET_FAKE_SCORE            = 208;// get score
    public static final int GET_FAKE_DEFAULT_THRESHOLD= 211;// get default threshold index
}

```

COMMAND	VALUE	RETURN DATA TYPE
GET_FAKE_ENGINE_READY	203	Byte[1]
GET_FAKE_NUM_OF_THRESHOLD	204	Byte[4]
GET_FAKE_THRESHOLD	205	Byte[4]
SET_FAKE_THRESHOLD	206	Byte[4]
GET_FAKE_THRESHOLD_VALUE	207	Byte[8]
GET_FAKE_SCORE	208	Byte[8]
GET_FAKE_DEFAULT_THRESHOLD	211	Byte[4]

2.3.1 Query the Fake Engine Status

Call SetGetData() with the GET_FAKE_ENGINE_READY command to determine if the fake engine is ready. The value 1 will be returned if the fake engine is available. The value 0 will be returned if the engine is not available.

C/C++

```
long err;  
DWORD fakeEngineReady;  
err = sgfplib->SetGetData(GET_FAKE_ENGINE_READY, &fakeEngineReady);
```

C#

```
Int32 iError = 0;  
Byte[] dwBuffer;  
dwBuffer = new byte[8];  
Int32 fakeEngineReady = 0;  
iError = m_FPM.SetGetData(FakeFingerThreshold.GET_FAKE_ENGINE_READY , dwBuffer);  
fakeEngineReady = BitConverter.ToInt32(dwBuffer, 0);
```

Java

```
boolean[] ready = new boolean[1];  
byte[] data = new byte[1];  
long result = sgfpm.SetGetData(SGSetGetData.GET_FAKE_ENGINE_READY,data);  
if (data[0] == 0x01)  
    ready[0] = true;  
else  
    ready[0] = false;
```


2.3.2 Query the Number of Fake Thresholds Supported

Call `SetGetData()` with the `GET_FAKE_NUM_OF_THRESHOLD` command to determine if the fake engine is ready. The number of thresholds supported will be returned. If the attached SecuGen device supports the fake detection engine, a value greater than 1 will be returned.

C/C++

```
long err;  
DWORD numThresholds;  
err = sgfplib->SetGetData(GET_FAKE_NUM_OF_THRESHOLD, &numThresholds);
```

C#

```
Int32 iError = 0;  
Byte[] dwBuffer;  
dwBuffer = new byte[8];  
Int32 nThresholds = 0;  
iError = m_FPM.SetGetData(FakeFingerThreshold.GET_FAKE_NUM_OF_THRESHOLD, dwBuffer);  
nThresholds = BitConverter.ToInt32(dwBuffer, 0);
```

Java

```
int[] numThresholds = new int[1];  
byte[] data = new byte[4];  
long result = SetGetData(SGSetGetData.GET_FAKE_NUM_OF_THRESHOLD, data);  
numThresholds[0] = DataConversion.toInt(data);
```

2.3.3 Query the Fake Threshold Currently Set in the Device Driver

Call SetGetData() with the GET_FAKE_THRESHOLD command to get the fake threshold currently set in the device driver.

C/C++

```
long err;  
DWORD threshold = FAKE_DETECTION_LEVEL_2;  
err = sgfpplib->SetGetData(GET_FAKE_THRESHOLD, &threshold);
```

C#

```
Int32 iError = 0;  
Byte[] dwBuffer;  
dwBuffer = new byte[8];  
Int32 threshold = 0;  
iError = m_FPM.SetGetData(FakeFingerThreshold.GET_FAKE_THRESHOLD , dwBuffer);  
threshold = BitConverter.ToInt32(dwBuffer, 0);
```

Java

```
int[] threshold = new int[1];  
byte[] data = new byte[4];  
long result = SetGetData(SGSetGetData.GET_FAKE_THRESHOLD, data);  
threshold[0] = DataConversion.toInt(data);
```

2.3.4 Set the Fake Threshold

You may call SetGetData() with the SET_FAKE_THRESHOLD command to set the fake threshold. However, it is preferable to use SetFakeDetectionLevel().

C/C++

```
long err;
DWORD threshold = FAKE_DETECTION_LEVEL_3;
err = sgfpplib->SetGetData(SET_FAKE_THRESHOLD, &threshold);
```

C#

```
Int32 iError = 0;
Byte[] dwBuffer;
dwBuffer = new byte[8];
Int32 threshold = FAKE_DETECTION_LEVEL_3;
dwBuffer = BitConverter.ToByteArray(threshold);
iError = m_FPM.SetGetData(FakeFingerThreshold.SET_FAKE_THRESHOLD , dwBuffer);
```

Java

```
int[] threshold = new int[1];
threshold[0] = FAKE_DETECTION_LEVEL_3;
byte[] thresholdarray = DataConversion.toByta(threshold);
long result = SetGetData(SGSetGetData.SET_FAKE_THRESHOLD,thresholdarray);
```

2.3.5 Query the Value Associated with a Fake Detection Level

Call SetGetData() with the GET_FAKE_THRESHOLD_VALUE command to get the fake threshold value associated a given fake detection level (FAKE_DETECTION_LEVEL_2 through FAKE_DETECTION_LEVEL_9).

C/C++

```
double threshold;
err = sgfplib->SetGetData(GET_FAKE_THRESHOLD_VALUE, &threshold);
```

C#

```
Byte[] dblBuffer;
dblBuffer = new Byte[16];
iError = m_FPM.SetGetData(FakeFingerThreshold.GET_FAKE_THRESHOLD_VALUE, dblBuffer);
Double ThresholdLevel = (Double)BitConverter.ToDouble(dblBuffer, 0);
```

Java

```
int[] numThresholds) = new int[1];
byte[] data = new byte[4];
long result = SetGetData(SGSetGetData.GET_FAKE_NUM_OF_THRESHOLD,data);
numThresholds[0] = DataConversion.toInt(data);
```

2.3.6 Get the Fake Score of the Last Fingerprint Capture

Call SetGetData() with the GET_FAKE_SCORE command to get the fake score of the last fingerprint image that was captured.

C/C++

```
double score;  
int result2 = sgfpplib->SetGetData(GET_FAKE_SCORE, &score);
```

C#

```
Byte[] dblBuffer;  
Double score;  
dblBuffer = new Byte[16];  
iError = m_FPM.SetGetData(FakeFingerThreshold.GET_FAKE_SCORE, dblBuffer);  
score = BitConverter.ToDouble(dblBuffer, 0);
```

Java

```
double[] score) = new double[1]  
byte[] data = new byte[8];  
long result = SetGetData(SGSetGetData.GET_FAKE_SCORE, data);  
score[0] = DataConversion.toDouble(data);
```