**The SDK Manuel for Blentity –**

**Data protector with FIDO2**

Rev 1.0

28th February 2025

Table of Contents

[1. Introduction 3](#_Toc192093137)

[1.1. The Blentity SDK Security Features 5](#_Toc192093138)

[1.2. Blentity SDK Architecture 7](#_Toc192093139)

[1.3. API Description Document 8](#_Toc192093140)

[2. Use Case and Example 9](#_Toc192093141)

[2.1 OpenPGP CLI Tool 9](#_Toc192093142)

[2.1.1 Export GPG public key 11](#_Toc192093143)

[2.1.2 Encrypt A Secret/Key Data 13](#_Toc192093144)

[2.1.3 Decrypt An Encrypted File 15](#_Toc192093145)

[3. Revision History 18](#_Toc192093146)

# Introduction

With the security features of Blentity and Blentity SDK, some extension software created by tool vendors can easily achieve the following requirements:

* Software Integrity
* Results cannot be accessed without authorization
* Operator authentication
* Report consistency
* Log persistency

Blentity is a multi-functional USB device. It provides the user with storage encryption, FIDO2 password-less authentication, and secure key management services.

From the user’s perspective, it can be used:

* to log in to any web application or operating system that supports FIDO (WebAuthnn) user authentication,
* to encrypt or hide the data within its bundled flash storage (encryption USB drive) and
* to securely keep the keys used by the applications

Blentity is managed by the built-in user tool, Blentity Manager. WiSECURE can also provide licensed customers with SDK (Software Development Kit) to help them create their own enhanced security features based on the functions of Blentity. The essential functions and the potential enhanced applications of Blentity are listed in the following diagram:

A computer hardware software diagram

AI-generated content may be incorrect.

Figure 1‑1 The Blentity Applications

## The Blentity SDK Security Features

The security features provided by Blentity:

* Read-only volume, protected (available after user secret verification or
* authentication) volume, and hidden-and-encrypted flash storage
* Standard FIDO authenticator functionality
* Key generation and maintained inside its secure chipDigital signature generation and verification (by public and private key)

The advanced security features can be accessible through Blentity SDK:

* hidden-and-encrypted storage read and write,
* cryptographic operations (including AES encryption and decryption), configuration of Blentity
* creation of the image of the read-only volume

The security features are depicted in the following diagram:



Figure 1‑2 The Blentuty SDK Features

The read-only volume can carry the read-only software binary without the risk of binary content modification by a malicious third party. Only the authorized party has the right to create the new contents of the read-only volume.

The protected volume within Blentity can be available only after the user or applications pass the PIN (Personal Identification Number) verification or public key cryptography-based challenge/response verification.

The verification mechanism of the protected volume can support the volume opened for the authorized local software entity or even if it is a remote web service.

## Blentity SDK Architecture

The Blentity SDK can be classified into the below classes and the architecture is shown in the Figure 1-3.

A diagram of a application

AI-generated content may be incorrect.

Figure 1‑3 Blentity SDK Architecture

* **Common API**: APIs related to device communication and basic device information.
* **Device management APIs**: APIs related to device management, status, and behavior control.
* **User management APIs**: APIs related to user authentication, including PUK and PIN verification.
* **FW management APIs**: APIs related to device firmware, including updates and manufacturing tools.
* **HSM APIs :** 
  + **Common Crypto Service(CCS) :** provides standard interface for the Samurai Key HSM, including crypto service(encrypt, sign, digest, …), persistency service (object store), object management (import / export key ) and access control (create session, login, …).
  + **PKCS #11(P11) :** Provide the P11 standard interface to access security objects.
* **Storage**: APIs related to the LUN operations, including format, reading, writing, settings, and information.
* **HSM Application APIs :**
  + **PGP**: APIs related to PGP, including file encryption/decryption and public/private key management.
  + **FileAegis**: the APIs related to FileAegis applications.
  + **KACLS**: Provides libkacls for users to access Google Workspace KACLS-related APIs, including certificate management and key packaging functions.
* **FIDO**: APIs that allow API users to send custom-packaged FIDO commands to the device and package list/delete RK operations as APIs.

## API Description Document

Use any web browser to open the *Blentity\_SDK\_doc\html\index.html* document for detailed descriptions of each API.

# Use Case and Example

## OpenPGP CLI Tool

The OpenPGP example demonstrates the WiSECURE OpenPGP APIs. In this example, users can learn how to use the relevant APIs.

**Software Platform:**

* + **OS :** Windows 11/10
  + **Development tool:**
    - **Windows PowerShell**
    - **Microsoft Visual Studio Community 2022 (64bits)**
    - **Gpg4Win :** [**Gpg4win - Secure email and file encryption with GnuPG for Windows**](https://www.gpg4win.de/index.html)

After building the project code with Visual Studio, users can find the compiled executable file, ***libbl\_example.exe***, in ***YourWorkDir\libbl\_example\x64\Debug*** or ***YourWorkDir\libbl\_example\x64\Release***

Use the following use case to demonstrate the encryption/decryption feature of the OpenPGP APIs:

* **OpenPGP CLI command Help :** 
  + Open the Windows PowerShell in the ***Debug or Release*** directory.
  + Type the following OpenPGP CLI command  to see how to use the CLI tool:

***.\libbl\_example.exe –help***

**A screenshot of a computer

AI-generated content may be incorrect.**

## Export GPG public key

In this case, you can learn how to export the GPG public key.

A yellow and blue keys

AI-generated content may be incorrect.

Figure 2‑1 Generate GPG Key Pair

Follow the steps below to generate a private/public key pair and export the public key to the ***mypuk.asc*** file.

* Open the **Cert Manager** in the **Blentity Manager** and create a key pair.

A screenshot of a computer

AI-generated content may be incorrect.

* Enter your PIN code.
* Open the **Windows PowerShell** in the project ***Debug or Release*** directory and type the following command:

***.\libbl\_example.exe --export mypuk***

A screenshot of a computer

AI-generated content may be incorrect.

## Encrypt A Secret/Key Data

In this case, you can learn how to use the previously exported public key to encrypt secret/key data.

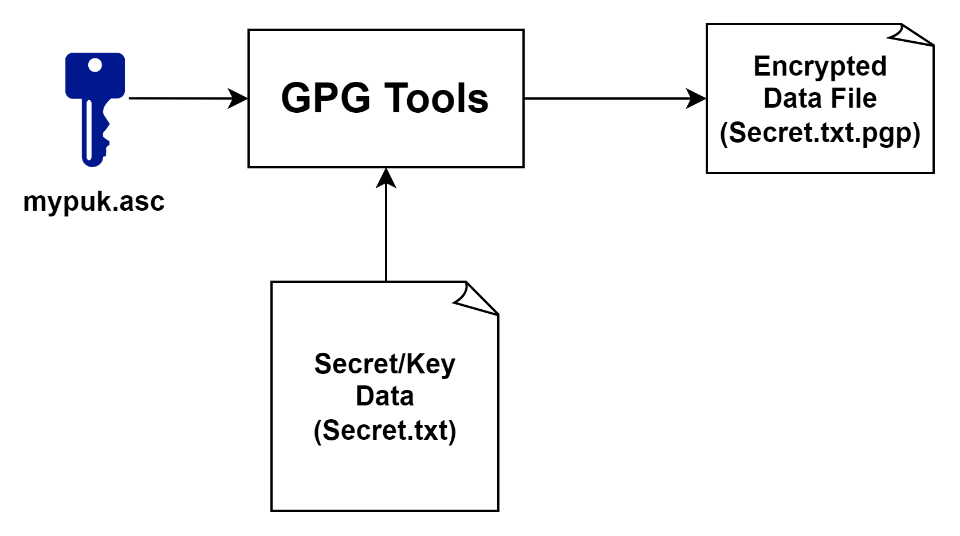


Figure 2‑2 Encrypt A Secret Data

* Open the **Windows PowerShell** in the project ***Debug or Release*** directory and type the following command:

***.\libbl\_example.exe -E .\mypuk.asc .\Secret.txt***

A screenshot of a computer

AI-generated content may be incorrect.

* After executing the above command, you will find an encrypted data file named “***Secret.txt.gpg***” in your directory.

A screenshot of a computer

AI-generated content may be incorrect.

## Decrypt An Encrypted File

In this case, you can learn how to decrypt an encrypted data file using your PIN code.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 2‑3 Decrypte An Encrypted File

* Open the **Windows PowerShell** under the project ***Debug or Release*** directory and type below command:

***.\libbl\_example.exe -D .\Secret.txt.gpg -p your\_pin\_code***

A screenshot of a computer

AI-generated content may be incorrect.

* After executing the above command, you will find an decrypted data file named “***Secret.txt***” in your directory.

A screenshot of a computer

AI-generated content may be incorrect.

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Author** | **Description** |
| 1.0 | 2025/03/04 | Peter | Initial Version |
|  |  |  |  |