Adafruithashlib Library Documentation

Release 1.0

Brent Rubell

Jan 14, 2020

Contents

This module implements a common interface to many different secure hash and message digest algorithms. Included are the FIPS secure hash algorithms SHA1, SHA224, SHA256, SHA384, and SHA512 (defined in FIPS 180-2) as well as RSA's MD5 algorithm (defined in Internet RFC 1321).

SHA1 and MD5 algorithms are not supported by the CircuitPython module.

This library is based on the work performed in the micropython-lib hashlib module by Paul Sokolovsky

CHAPTER 1

Dependencies

This driver depends on:

- Adafruit CircuitPython

Please ensure all dependencies are available on the CircuitPython filesystem. This is easily achieved by downloading the Adafruit library and driver bundle.

CHAPTER 2

Installing from PyPI

On supported GNU/Linux systems like the Raspberry Pi, you can install the driver locally from PyPI. To install for current user:

```
pip3 install adafruit-circuitpython-hashlib
```

To install system-wide (this may be required in some cases):

```
sudo pip3 install adafruit-circuitpython-hashlib
```

To install in a virtual environment in your current project:

```
mkdir project-name && cd project-name
python3 -m venv .env
source .env/bin/activate
pip3 install adafruit-circuitpython-hashlib
```

CHAPTER 3

Usage Example

```
import adafruit_hashlib as hashlib
m = hashlib.sha256()
m.update(b"CircuitPython")
print("Msg Hex Digest: {}\nMsg Digest Size: {}\nMsg Block Size: {}".format(
        m.hexdigest(), m.digest_size, m.block_size))
```

CHAPTER 4

Contributing

Contributions are welcome! Please read our Code of Conduct before contributing to help this project stay welcoming.

CHAPTER 5

Documentation

For information on building library documentation, please check out this guide.

CHAPTER 6

Table of Contents

6.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/hashlib_simpletest.py

```
# pylint: disable=no-member, line-too-long
import adafruit_hashlib as hashlib

# Bytes-to-encode
byte_string = b"CircuitPython"

# Create a SHA-1 message
print("--SHA1--")
m = hashlib.sha1()
# Update the hash object with byte_string
m.update(byte_string)
# Obtain the digest, digest size, and block size
print(
    "Msg Digest: {}\nMsg Digest Size: {}\nMsg Block Size: {}".format(
        m.hexdigest(), m.digest_size, m.block_size))
# Validate the digest against CPython3 hashlib-sha1
assert (
    m.hexdigest() == "62c6e222ccd72f21b8ce0c61f42860d6c70954c0"
), "Digest does not match expected string."


# Create a SHA-224 message
print("--SHA224--")
m = hashlib.sha224()
# Update the hash object with byte_string
m.update(byte_string)
# Obtain the digest, digest size, and block size
```

(continues on next page)

(continued from previous page)

```
print(
    "Msg Digest: {}\nMsg Digest Size: {}\nMsg Block Size: {}".format(
        m.hexdigest(), m.digest_size, m.block_size))
# Validate the digest against CPython hashlib-sha224
assert (
    m.hexdigest() == "744535a10879be6b18bbcdd135032891346f530a7845d580f7869f36"
), "Digest does not match expected string."

# SHA-256
print("--SHA256--")
m = hashlib.sha256()
# Update the hash object with byte_string
m.update(byte_string)
# Obtain the digest, digest size, and block size
print("Msg Digest: {}\nMsg Digest Size: {}\nMsg Block Size: {}".format(
    m.hexdigest(), m.digest_size, m.block_size))
# Validate the digest against CPython hashlib-sha256
assert (
    m.hexdigest() == "3ce8334ca39e66afb9c37d571da4caad68ab4a8bcbd6d584f75e4268e36c0954
↪"
), "Digest does not match expected string."

# SHA-384
print("--SHA384--")
m = hashlib.sha384()
# Update the hash object with byte_string
m.update(byte_string)
# Obtain the digest, digest size, and block size
print("Msg Digest: {}\nMsg Digest Size: {}\nMsg Block Size: {}".format(
    m.hexdigest(), m.digest_size, m.block_size))
# Validate the digest against CPython hashlib-sha384
assert (
    m.hexdigest() ==
↪"7a12f0815f5511b8ba52c67922d1ae86dfd9bfcc4e0799ad89a9f01fc526c8f074ddb5948c06db9893536f2e65c7621b
↪"
), "Digest does not match expected string."

# SHA-512
print("--SHA512--")
m = hashlib.sha512()
# Update the hash object with byte_string
m.update(byte_string)
# Obtain the digest, digest size, and block size
print("Msg Digest: {}\nMsg Digest Size: {}\nMsg Block Size: {}".format(
    m.hexdigest(), m.digest_size, m.block_size))
# Validate the digest against CPython hashlib-sha512
assert (
    m.hexdigest() ==
↪"20a88a9b04aa490e457f8980e57331bc85c4d6ca30735a9e502f817e74011a9ece07078e53adf70c232ac91f6c79d4cd6
↪"
), "Digest does not match expected string."
```

6.2 adafruit_hashlib

Secure hashes and message digests

- Author(s): Paul Sokolovsky, Brent Rubell

6.2.1 Implementation Notes

Hardware:

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: https://github.com/adafruit/circuitpython/releases

adafruit_hashlib.**algorithms_available**
Returns a list containing the names of the hash algorithms that are available in this module.

adafruit_hashlib.**new**(*algo*, *data=b''*)
Creates a new hashlib object. :param str algo: Name of the desired algorithm. :param str data: First parameter.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index

A

N