Cryptography

cryptography

cryptography is an actively developed library that provides cryptographic recipes and primitives. It supports Python 2.6-2.7, Python 3.3+, and PyPy.

cryptography is divided into two layers of recipes and hazardous materials (hazmat). The recipes layer provides a simple API for proper symmetric encryption and the hazmat layer provides low-level cryptographic primitives.

Installation

```
$ pip install cryptography
```

Example

Example code using high level symmetric encryption recipe:

```
from cryptography.fernet import Fernet
key = Fernet.generate_key()
cipher_suite = Fernet(key)
cipher_text = cipher_suite.encrypt(b"A really secret message. Not for prying eyes.")
plain_text = cipher_suite.decrypt(cipher_text)
```

GPGME bindings

The GPGME Python bindings provide Pythonic access to GPG Made Easy, a C API for the entire GNU Privacy Guard suite of projects, including GPG, libgcrypt, and gpgsm (the S/MIME engine). It supports Python 2.6, 2.7, 3.4, and above. Depends on the SWIG C interface for Python as well as the GnuPG software and libraries.

A more comprehensive GPGME Python Bindings HOWTO is available with the source, and an HTML version is available at http://files.au.adversary.org. Python 3 sample scripts from the examples in the HOWTO are also provided with the source and are accessible at gnupg.org.

Available under the same terms as the rest of the GnuPG Project: GPLv2 and LGPLv2.1, both with the "or any later version" clause.

Installation

Included by default when compiling GPGME if the configure script locates a supported python version (which it will if it's in $PATH during configuration).

Example

```
import gpg

# Encryption to public key specified in rkey.
a_key = input("Enter the fingerprint or key ID to encrypt to: ")
filename = input("Enter the filename to encrypt: ")
with open(filename, "rb") as afile:
    text = afile.read()
c = gpg.core.Context(armor=True)
rkey = list(c.keylist(pattern=a_key, secret=False))
ciphertext, result, sign_result = c.encrypt(text, recipients=rkey,
                                            always_trust=True,
                                            add_encrypt_to=True)
with open("{0}.asc".format(filename), "wb") as bfile:
    bfile.write(ciphertext)
# Decryption with corresponding secret key
# invokes gpg-agent and pinentry.
with open("{0}.asc".format(filename), "rb") as cfile:
    plaintext, result, verify_result = gpg.Context().decrypt(cfile)
with open("new-{0}".format(filename), "wb") as dfile:
    dfile.write(plaintext)
# Matching the data.
# Also running a diff on filename and the new filename should match.
if text == plaintext:
    print("Hang on ... did you say *all* of GnuPG?  Yep.")
else:
    pass
```

PyCrypto

PyCrypto is another library, which provides secure hash functions and various encryption algorithms. It supports Python version 2.1 through 3.3.

Installation

```
$ pip install pycrypto
```

Example

```
from Crypto.Cipher import AES
# Encryption
encryption_suite = AES.new('This is a key123', AES.MODE_CBC, 'This is an IV456')
cipher_text = encryption_suite.encrypt("A really secret message. Not for prying eyes.")

# Decryption
```

```
        decryption_suite = AES.new('This is a key123', AES.MODE_CBC, 'This is an IV456')
        plain_text = decryption_suite.decrypt(cipher_text)
```