

Encryption

You can use encryption on any file type supported by pyndb.

You must have an up-to-date version of the cryptography module to use encryption features. Run "pip -U install cryptography" to update.

pyndb allows you to encrypt your databases using Fernet.

Encrypting a new database

Simply specify a password when creating a database to encrypt it.

```
from pyndb import PYNDatabase
new_db = PYNDatabase('new_db.pyndb', password='your-password-here')
```

If you would like, you can also specify a salt and/or a number of iterations.

```
from pyndb import PYNDatabase
new_db = PYNDatabase('new_db.pyndb', password='a really long password', salt=b'A bytes object',
iterations=390000)
```

Due to limitations of PBKDF2, a salt and number of iterations must be provided. By default, pyndb uses `pyndb_default` as the salt, and 390000 as the number of iterations.

Encrypting an existing database

if you load a database with the wrong password, it will try to load it without one. This is so that unencrypted databases can be loaded even if a password is specified. The password will also be set to `None` so that the database is not accidentally encrypted. Therefore, to encrypt an existing database, you can do as follows:

```
from pyndb import PYNDatabase
old_db = PYNDatabase('old_db.pyndb')
old_db.password = 'a very long password'
old_db.save()
```

Removing encryption

To remove encryption, you can load the encrypted database with the password and then set it to None.

```
from pyndb import PYNDatabase
db = PYNDatabase('db.pyndb', password='your-password-here')
db.password = None
db.save()
```

Revision #1

Created 28 May 2022 17:20:22 by jvadair

Updated 28 May 2022 17:44:44 by jvadair