# DarkCastle

## (Authenticated File Encryption Program)

*by Karl Zander*

*DarkCastle is an authenticated file encryption program designed to allow two parties to exchange a file message using the public key algorithm QloQ and the symmetric algorithms ZANDERFISH3 in CBC mode, ZANDERFISH4 in CBC or the QAPLA and NUQNEH stream ciphers. It works on Unix and Unix-like operating systems such as, Linux, FreeBSD, NetBSD, OpenBSD, MacOS and Solaris. This document outlines the design of the program.*

## Flow

File encryption (ZANDERFISH4-CBC) → KDF (QX) → HMAC (QX) → Key Exchange (QloQ) → Signing (QloQ)

File encryption (ZANDERFISH3-CBC) → KDF (QX) → HMAC (QX) → Key Exchange (QloQ) → Signing (QloQ)

or

File encryption (NUQNEH) → KDF (QX) → HMAC (QX) → Key Exchange (QloQ) → Signing (QloQ)

File encryption (QAPLA) → KDF (QX) → HMAC (QX) → Key Exchange (QloQ) → Signing (QloQ)

## Design

The design of DarkCastle is meant to be feature full without being too complex to follow. Each message has a random key generated through the operating system URANDOM PRNG and is encrypted for key exchange using QloQ. The file contents are then encrypted using the random key. The random key is run through the QX KDF function to produce a key suitable for the QX HMAC function. HMAC is run and the MAC is appended to the file. Lastly, the message is hashed and the hash used in QloQ public signing showing that the message comes from the originator.

DarkCastle features four symmetric ciphers of ARX (Add-Rotate-XOR) constructions: ZanderFish3 a block cipher in CBC mode, ZanderFish4 in CBC mode, NuqneH and Qapla stream ciphers.

A public key algorithm is used for key encapsulation, QloQ. It utilizies two moduli composed of two primes (3072 bit size). Each encryption, decryption or signing operation takes two exponentiations rather than the RSA single exponentiation.

## Operation

QloQ public key pairs must be generated by each party in order to allow them to communicate securely. The .sk file must be kept secret at all times. The .pk file is meant to be exchanged between the two parties and may be kept public at all times.

Key Generation: castle-keygen username

DarkCastle is executed using the following commands. Suppose Alice wants to send a message to Bob. Alice uses Bob's public key pair and her secret key pair to encrypt the message. Upon decryption, Bob uses Alice's public key pair and his secret key pair to decrypt the message.

Encryption: castle zanderfish4 -e msg msg.enc bob.pk alice.sk
Decryption: castle zanderfish4 -d msg.enc msg.dec alice.pk bob.sk

Encryption: castle zanderfish3 -e msg msg.enc bob.pk alice.sk
Decryption: castle zanderfish3 -d msg.enc msg.dec alice.pk bob.sk

or

Encryption: castle nuqneh -e msg msg.enc bob.pk alice.sk
Decryption: castle nuqneh -d msg.enc msg.dec alice.pk bob.sk

Encryption: castle qapla -e msg msg.enc bob.pk alice.sk
Decryption: castle qapla -d msg.enc msg.dec alice.pk bob.sk

## Ciphers

Symmetric Ciphers

ZanderFish4 – 64 rounds at 256 bits (256 bits of round key per round, 256 extra bits of round key applied last round)
ZanderFish3 – 56 rounds at 256 bits (256 bits of round key per round, 256 extra bits of round key applied last round)
NuqneH – 24 rounds at 256 bits (512 bit state)
Qapla – 20 rounds at 256 bits (512 bit state)

Asymmetric Cipher

QloQ – RSA inspired variant

Hash Algorithm (HMAC)

QX – 512 bit state 256 bit output