LiteCastle Grimoire

Karl Zander – uvajda@protonmail.com

Abstract.

LiteCastle is an authenticated file encryption program, obfuscating the file's message and detecting if the message has been tampered with. The file is secured by a passphrase protected secret key file and only the person knowing the passphrase may unlock the key file to perform signing or decryption. Encryption is achieved with the ZanderFish3 CBC mode block cipher and Q'loQ-RSA (3072 bit prime size).

1. Introduction

LiteCastle is a program written purely in C with cryptographic primitives designed by myself. LiteCastle hopes to bring professional levels of security to files that are encrypted using the program. The program is cross platform, working on Linux, MacOS, FreeBSD, OpenBSD, NetBSD and Solaris.

In writing DarkCastle, I wanted to create something smaller, more compact, fewer options but great strength so I chose the ZanderFish3 cipher to be the cipher of choice for LiteCastle. Authentication is achieved through the Ganja HMAC algorithm, key derivation through Manja and key wrapping provided by the Amagus stream cipher. Public key encryption and signing authentication is achieved with the Q'loQ-RSA algorithm.

2. Design goals

The application must be able to provide secure communication between two people

The application must offer only one cipher and only one key length, thus being Lite'er than DarkCastle.

The cipher chosen for the program must be state of the art.

The number of key derivation iterations must be high.

The application must employ best security practices when encrypting, decrypting or authenticating data.

The application should execute, encrypt and authenticate quickly.

Each message much be individually key wrapped with a random key and passkey.

At the very least the application must consist of a cipher, message authentication code and key derivation function.

The application must process data in a memory efficient way.

The application must perform comparably to popular encryption applications.

3. Installation and usage

- Installation prerequisites

OpenSSL is required for the public key components

GCC/Clang, make and git must be already installed

On Debian systems you may run `apt install build-essential git libssl-dev` to install the prerequisites

On MacOS you may have to soft link your include directories

- Installation

To install LiteCastle run the following commands:

1. git clone https://github.com/pvial00/LiteCastle
2. cd LiteCastle/src
3. make
4. mv lcastle /usr/local/bin (This works on Mac systems, on Linux you may have to `sudo mv lcastle /usr/local/bin`

- Usage

Simply running `lcastle` will print the algorithms available and parameter format.

To encrypt a file with LiteCastle, do the following:

`lcastle -e <input file> <output file> <public keyfile> <secret keyfile>`

To decrypt a file with LiteCastle, do the following:

`lcastle -d <inputfile> <outputfile> <secret keyfile> <public keyfile>`

4. Key derivation

The Manja algorithm was selected to be the key derivation function for LiteCastle. Manja takes a maximum of 256 characters/bytes as it's input and thus LiteCastle has a 256 character limit for passwords.

LiteCastle uses 100,000 Manja iterations and a salt of "LiteCastleZFish".

5. Key wrapping algorithm and key generation

- Key generation

Key generation is achieved using the amagus_random() PRNG which sources it's initial entropy for a random 1024 bit key and 128 bit nonce from /dev/urandom. The cipher used in the PRNG is Amagus.

Key wrapper algorithm – Amagus

The key wrapper operates by first generating itself a random nonce using amagus_random() and a random key of appropriate length using the same generator. The cipher is then fed the key supplied by the KDF and the random nonce and encrypts key. The key is then XOR'ed with the KDF key and is written to file.

To decrypt the process is run in reverse.

6. Authentication

Authentication is achieved through use of the Ganja hash function. ganja_hmac() is called to generate 256 bit hashes which are used as message authentication codes for messages. The nonce/iv, key wrapped key, key wrap nonce and message are all run through ganja_hmac() to produce a MAC for the message that is attached to the front of the file.

Sender authentication is achieved with the Q'loQ public key encryption algorithm. Each message must be signed using the sender's secret key. Verification is done with the receiver's public key.

7. Passphrase security

As of version 1.3, a passphrase (max length 256 characters) is now required to encrypt the secret key file upon generation. This passphrase is also asked when signing or decrypting a message. The public key file is not encrypted.

8. Cryptanalysis

TBD

9. Statistical Properties

LiteCastle was subjected to the Diehard battery of tests (dieharder) and NIST statistical testing. LiteCastle passed these tests.