ANATOMY OF A miniLock KEY PAIR

REQUIRES USE OF NaCl, SCRYPT, ZXCVBN, & BLAKE2s LIBRARIES IN LANGUAGE OF YOUR CHOOSING

E-Mail Address

Example: miliLock+test1@mailinator.com

Passphrase (with sufficient entropy, >100bits)

Example: o8rS%kA33qHCh^mbbXn6d$eoq9dbAvZc

Example Public Key: 6ZLq23QX4NabaoF2VjXwcVY6gTtNPN6rss4duLs6pEqbb

VERIFY ENTROPY OF PASSPHRASE

In the official release, this is accomplished using the ZXCVBN library (named so for a crappy password (hint: look on your keyboard)). The C# port of miniLock uses Michael Ford's port (ZXCVBN-CS). The desired result is AT LEAST 100 BITS of entropy. Failing to provide a good passphrase, the official release relies on a dictionary of 58110 words and randomly chooses 7 of them to suggest to the user a better alternative (approximately 111 bits according to miniLock's author).

BASED ON VERSION 1.0.2 OF NADIM KOBEISSI'S miniLock OFFICIAL RELEASE ON GitHub

@SparkDustJoe OCT 2014 DOC VERSION 1.1

HASHING THE PASSPHRASE & SECRET (OR "LONG TERM") KEY

Upon acceptance of a passphrase, it is passed through a Blake2s hash (no key, no salt), and then through an SCRYPT transformation, the parameters of which are as follows: Number of Iterations (CPU Cost) = 2^17 (or 131072), Parallelism (Threads) = 1, Block Size = 8, Output Length = 32, Salt = UTF8 Encoded E-Mail string

The 32 byte output is the Secret Key. This transformation takes the longest during key generation, and it is so on purpose: to prevent speedy-brute-force attacks.

Note: Ephemeral Key Pairs skip this step and rely on securely-generated, random, 32-byte arrays.

PUBLIC KEY

To generate a Public Key from the Secret Key (sometimes referred to as the "Long Term Secret" or Private Key), run a Curve25519 "Get Public Key" call in your NaCl library of choice. This process executes very quickly and will produce a 32 byte output. This is the Public Key.

CHECK SUM & PUBLIC KEY ID

All miniLock Public Keys that are not stored as 32bytes in Base64, are encoded as 33 bytes of Base58 (which was chosen for maximum key printability and portability). The 33rd byte is a Blake2s hash of the first 32 bytes (output length 1). To verify if a public key is "valid" (not mangled in transmission), decode the Base58 string, and compare a Blake2s hash of the first 32 bytes (output length 1) against the 33rd byte.