ANATOMY OF A miniLock FILE

REQUIRES USE OF NaCl, JSON, AND BLAKE2s LIBRARIES IN LANGUAGE OF YOUR CHOOSING

ENCRYPTED FILE DATA STRUCTURE (AT REST)

[8byteASCII"miniLock"][4byteHeaderLength(littleEndian)] [JSON"stringified"Header][Ciphertext"chunks" ConcatenatedTogetherInBinaryFormat]

OUTERMOST JSON HEADER OBJECT

{"version":1,"ephemeral":"[Base64EphemeralPublicKey]", "decryptInfo":{"[Base64RecipientNonce]":"[Base64Encoded EncryptedInnerHeader]"}}

The object “decryptInfo” is like a c# Dictionary<string,string> object, containing (at least one) key/value pair of a Recipient Nonce (unique to each recipient), and the Encrypted Inner Header. Only the Shared Key of the Ephemeral Public Key and the Recipient Private Key (with the Recipient Nonce) can unlock an Inner Header (and in theory, only one of them, so you have to try them all). If no Inner Headers can be decrypted, that private key is not an intended recipient.

Note: Neither the Recipients nor the Sender are visible yet!

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

@SparkDustJoe OCT 2014 DOC VERSION 1.1

INNER HEADER JSON OBJECT (DECRYPTED)

{"senderID":"[Base58EncodedPublicID]","recipientID": "[Base58EncodedPublicID]","fileInfo":"[Base64Encoded EncryptedFileInfoHeader]"}

If the recipientID doesn’t match the public key of the recipient attempting to decrypt the file, this is considered an ERROR (Abort). The File Info Header can only be unlocked using the Shared Key of the Sender Public Key and the Recipient Private Key (with the Recipient Nonce).

FILE INFO HEADER JSON OBJECT (DECRYPTED)

{"fileKey":"[Base64Random32bytes]","fileNonce": "[Base64Random16bytes]","fileHash":"[Base64Encoded Blake2s32byteHashOfEntireCiphertextBlock]"}

The fileKey is used to decrypt the Ciphertext “Chunks”, along with the 16byte fileNonce concatenated with an 8byte “chunk number” (little endian), with the last chunk’s most-significant-bit set to 1. Note: use Blake2s, not Blake2b!

CIPHERTEXT CHUNK (AT REST)

[4bytePlaintextLength(LittleEndian)][BinaryChunkByteStream]

To obtain the actual length of a ciphertext chunk when stored in the miniLock file, add 16 to the Plaintext Length. The decrypted length should match the indicated value. This is a product of the XSalsa20 Poly1305 process. The first chunk is always 256 plaintext bytes and contains the original filename, padded with nulls (so all recipients get the same filename).

Note: A plaintext chunk cannot exceed 1024 *1024 (or 2^20, or 1048576, or 1 MEGA) bytes!