HlyGrail or Holy Grail of Bitcoin

This document is the proof of ownership of the algorithm to use a bitcoin transaction as a verifiable escrow for any digital material AND the code to decrypt the file. The base concept is to use a bitcoin transaction that contains a bitcoin consumed input address, and an OP_RETURN with data that is a hash code which verifies digital data using the bitcoin address as the key for the digital data during encryption. The date of the transaction is the date of creation and signature for the digital data. The consumed input bitcoin address is the key used for encryption/decryption of the digital data which can be verified with the OP_RETURN before decryption. The encrypted data could be the digital data itself or an intellectual property file containing the real digital data location or locations, hash of that file or files, ownership and decrypting information such as type of encryption etc. (A consumed input is chosen for the key to make it easier to find the OP_RETURN especially if you do not know the transaction. But technically inputs and outputs could be the key or keys depending on the application.)

This concept can be used as a simple escrow verification of the digital data or can be as complex as the verification of a new distributed programming language. As example the address could decrypt an opcode for a programming language algorithm and the OP_RETURN could contain a hash for the verified actual code prior to processing. The transaction or address could be an index to the encrypted code file that is verified by the OP_RETURN hash prior to decoding and running the algorithm. By verifying the code with the hash, no one could change the actual code to inject a virus. Only verified opcodes would be used in the program and the finished program could be verified the same way. Many copies of the encrypted code can be run on many servers but still be verified as the original code from the hash before decrypting with the address, allowing massive parallel processing.

In the simplest terms, the digital data could be unencrypted and the OP_RETURN contains the checksum for the file. The address or transaction would be the lookup for the file on a known website. This is similar to Proof of Existence. Using the address as the key is a tighter binding providing more security of the digital data and can be verified prior to decryption and using the data as code. The key, checksum and lookup are all stored in one transaction and can be chained for program purposes.

portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON INFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

HlyGrail is written as hex encoded to 486c79477261696c and CAN be used as a prefix in the OP_RETURN to designate use of this algorithm. The data of the OP_RETURN should be the hash of any file: audio, video, pictures, documents, intellectual property or programs that are encoded with any encryption using the consumed input bitcoin address as key for the encryption, the OP_RETURN to prove the file is original and the transaction date for recording date.

The original digital data (like this file) can be provided in plain text and verified by a checksum. The plain text checksum can be placed in an intellectual property file. The intellectual property file can be encrypted, or the original digital data can be encrypted with the consumed input address as the key for encryption and the OP_RETURN containing the hash of the encrypted file. The digital data and or the intellectual property file can be distributed by any method. If using multiple output addresses, they could hold the address/key for where the file is located with their spent OP_RETURN being their hash or instructions. Other outputs could lead to additional information or a chain such as a computer program.

There are many options for usage of this algorithm and no one is using it at the time of this writing. Many use the idea that an OP_RETURN holds a hash of a document but none are using the consumed address as the key for encryption and pointer for data. The transaction can be used as a pointer to the intellectual property file.

The use of the address as the key is critical because it connects proof and time of a document to the encryption of the document in bitcoin and linking to the document. This allows for easy proof of ownership but most importantly is the ability to encode something that can be shown in plain text or encrypted that can be proved before decrypting to avoid viruses or other modification of the data. The major benefit is it allows the storage of virtually unlimited data coded FROM the blockchain but not needed to be coded IN the blockchain.

Examples:

THIS document (hlygrail0.pdf) is a prime example of how to use the algorithm. THIS document is stored in plain text available for anyone. There is an intellectual property file (hlygrail0.txt) for THIS file that is encoded with bitcoin 18hsnexG7KGUBtKgYnCS2Zyg59V5CsBPAt using rijndael-256 mode NCFB. The very simple concept is the address is the key and the OP_RETURN holds the verification SHA-256 hash of the intellectual file (hlygrail0.txt). The intellectual file contains the location or locations of THIS document (HlyGrail or Holy Grail of Bitcoin) including the SHA-256 checksum for THIS document. There is no way to create the intellectual property file containing the hash of THIS document until THIS document is created. Also there is no way to create the hash for the intellectual property file until it is created and encrypted using the bitcoin address and then stored in the OP_RETURN. The OP_RETURN transaction date proves existence date and the associated encoding /

decoding bitcoin address for the OP_RETURN proves ownership in the intellectual property file because knowledge and OWNERSHIP of the bitcoin address was needed to encrypt the intellectual property file AND store the hash in the OP_RETURN.

Simple use: Proves the owner of the file created the file and the address linking to the file.
1. The address is the key and the hash is the verification. The OP_RETURN can contain a code that indicates a file and the hash. (The code could be a look up for a file or an index of files. You can hold over 4 billion unique indexes in eight bytes of hexadecimal code. The remaining 32 bytes would be the hash or checksum of the intellectual property.)
2. On a website, the transaction, address or code points to a file on the site or possibly online storage like dropbox that is encoded with the address. The creator also states the type of encryption and the type of hash used for the file on the website or the encryption is standard to that specific application.
3. The intended user can verify the hash of the file before decrypting the file with the address especially if the digital data is code for a computer program.

Two Address Pointer: Assume that encryption and hash functions are known.
1. One consumed INPUT with OP_RETURN that verifies Hash of the encrypted file.
2. One OUTPUT that has an output OP_RETURN which contains a link to the data. (Currently only one OP_RETURN is allowed or you could have two OP_RETURNS from the original address.) The data might be a website link which contains the encrypted file.

As an online distributed program, an ADDRESS or transaction could point to the encrypted code file for an algorithm in one OP_RETURN, with certified code verification in the other OP_RETURN. As example the ADDRESS might divide two numbers. You get the file location of the encoded algorithm from one OP_RETURN then verify it with the hash of the other OP_RETURN before decoding, passing in two numbers and running the divide algorithm. Next you point to another ADDRESS that contains another algorithm like computing velocity from the answer from the division ADDRESS. The neat part is hundreds of people could use the same address for division at the same time and the code file could be on many different computers still using the same hash code to verify the software. (This actually could be done with one OP_RETURN for the verification if the location of the code is known or loaded on the local machine, or if the chain output has an OP_RETURN to a website.)

Three Address Pointer: Assume encryption and hash functions are known. The address links could be on a website and/or indexes could be stored in the front of the hash checksum.
1. One consumed INPUT with OP_RETURN to verify Hash of the encrypted file.
2. One OUTPUT that has an OP_RETURN which contains a link to data.
3. One OUTPUT that continues the SPEND chain to another address.

This process allows the data to be encrypted and verified and linked to another instruction with encrypted and verified data. As example you could encrypt Genesis 1 of the Bible and verify the hash and point to the unchangeable Word for that verse, then the third address would point to Genesis 2 which would point to verification and the encrypted verse and a pointer to the next verse etc.

This could be done with two pointers by knowing the file address or having the encoded file on the local machine, or if the chain output has an OP_RETURN to a website. The OP_RETURN pointer would be the hash for verification of the Word and the spend pointer to the new address would be the next verse. (The link to the file could be the transaction instead of the address which is also the lookup for the bitcoin and the OP_RETURN with the hash.)

Expand Blockchain Storage Offline:
You could even store previous blocks of the blockchain in the blockchain with many advantages. One would only need to keep a list of the transactions with OP_RETURN containing previously encrypted block's hash code. Periodically the software would add new transactions containing encrypted blocks to the previous list. Maybe only a few thousands transactions would need to be kept in the actual full node with references to many locations containing the actual blocks which could always be verified by the hash code contained in the transaction for that block. Many lookup process could be developed to speed the lookup of transactions stored in off chain validated blocks. Anyone could make a verifiable chain of the blockchain stored offline.

This document should stand as an international patent and copyright without the need of governments or agencies, lawyers and extortion payments. It will allow anyone anyplace in the world to equally prove their intellectual property regardless of stature or financial situation. This will not automatically give ownership to someone that registers previous unrecorded intellectual property, but it will establish the DATE and DETAILS of YOUR claim. (Warning: If you do take someone else's intellectual property such as a picture, encode it into an intellectual property file tied to a bitcoin address using this algorithm, all you accomplish is PROVING you stole or attempted to steal the property.)

Visit the website for more information or follow the address spends for this document and maybe you will find more hash codes to files of interest.

Owner: Roger Johnsrud, Genus Enterprises LLLP Date 1/10/2016
Website http://hlygrail.com

Use the following ADDRESS to find the transaction that contains OP_RETURN as output:
ADDRESS: 18hsnexG7KGUBtKgYnCS2Zyg59V5CsBPAt – Key for decoding hlygrail0.txt
OP_RETURN: SHA-256 hash verifies the encoded Intellectual Property file hlygrail0.txt

HlyGrail 486c79477261696c is the prefix for the OP_RETURN

Decode the intellectual property file hlygrail0.txt with rijndael-256 mode NCFB using ADDRESS

The decoded intellectual property file hlygrail0.txt contains the SHA-256 checksum of THIS file (hlygrail0.pdf)

HlyGrail Patent / Copyright Number 18hsnexG7KGUBtKgYnCS2Zyg59V5CsBPAt