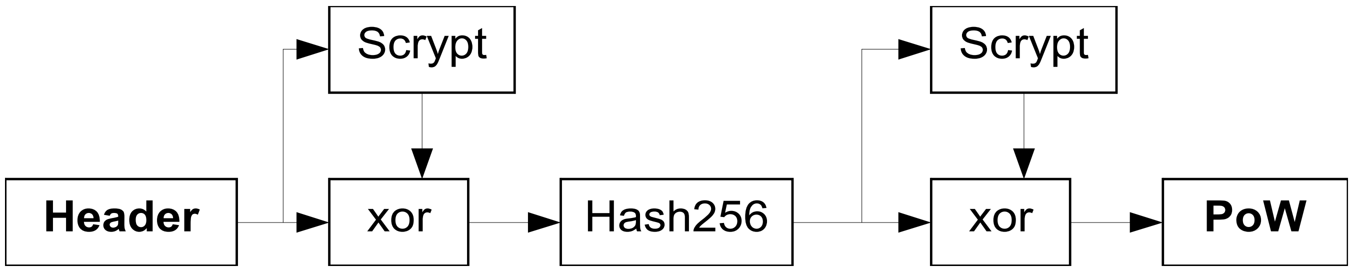
"Hybrid Scrypt-Hash256" Proof Of Work algorithm

For a Block Erupter or similar device, with an hash rate of about 333 Mhash/s, the current Bitcoin mining profitability is, at the time of writing (2013-12-28) about 0.004 BTC/Month and will continue to decrease, as the Bitcoin mining difficulty continues to grow.
It has become impractical to use Block Erupters and similar smaller devices to mine Bitcoins.
So, what do of all these Erupter devices? Wouldn't it be nice to find a reuse for them?

The HybridScryptHash256 Algorithm has been designed with the target of the reuse the large number of Erupters but keeping away the new ASIC devices which are order of magnitude more powerful.

The HybridScryptHash256 works in a three stage way: two Scrypt stage (with variable parameters depending on the current network difficulty) with in between the usual Hash256 stage.

For additional cryptographic security, two additional xor stages have been inserted.

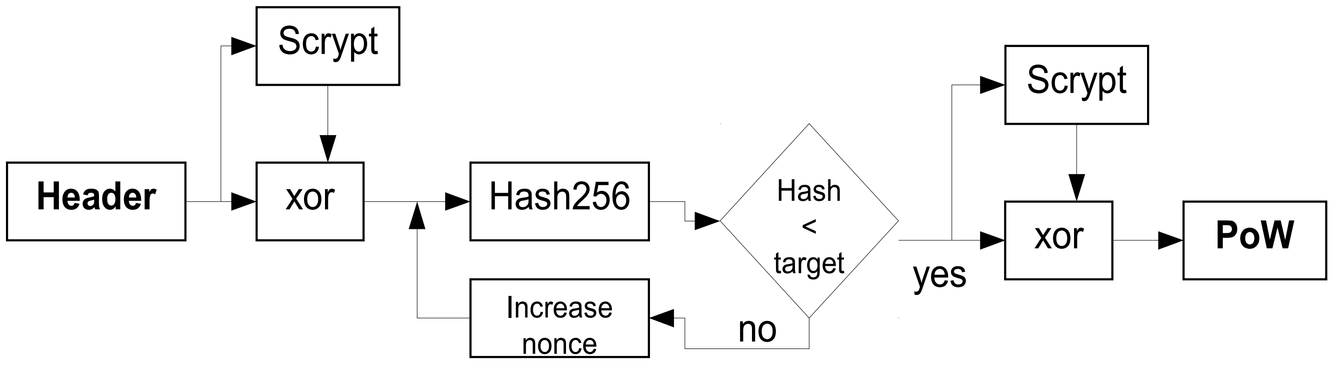


The idea is to allow all the users of erupters and similar small devices to mine an altcoin without having "whales" coming and disrupt mining. With HybridScryptHash256, each HASH256 mining iteration with an erupter is encapsulated between two scrypt stages (whose parameters depend on the current difficulty). In this way, a good amount of CPU power (or maybe even GPU power) is necessary for each HASH256 dedicated hardware. If a miner owns a lot of HASH256 computing power, in order to mine altcoins, he will need also a corresponding CPU power. If the correspondent CPU power is not available, then the HASH256 computing power is automatically limited.

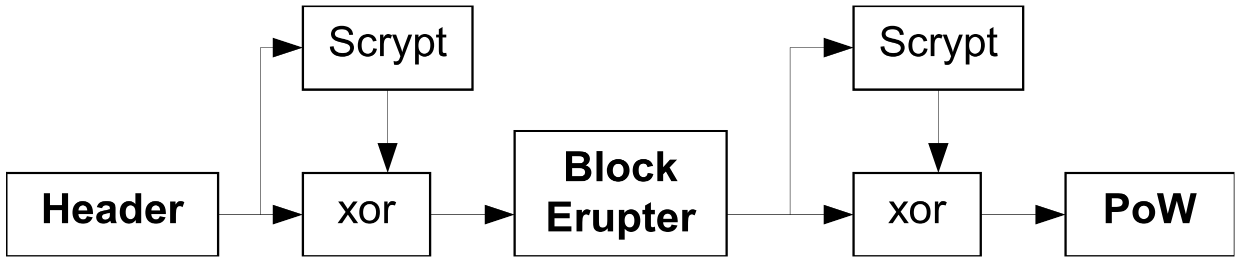
A first implementation of HybridScryptHash256 can be found in MediterraneanCoin (http://www.mediterraneancoin.org). Source code for the implementation of HybridScryptHash256 in MediterraneanCoin can be found here: https://github.com/mrtexaznl/mediterraneancoin/blob/master-0.8/src/scrypt.cpp

Mining with HybridScryptHash256

When mining for a new block, the HybridScryptHash256 becomes the following:



In practice, when using a Block Erupter or similar, the block diagram becomes the following:



Explanation of block diagram and detailed description of the algorithm

The input parameter is H80 (block header), an array of bytes of size 80.

steps for calculation of PoW:

extract nBits (difficulty) from header (byte 72-75)

extract parameters for scrypt stages from data array (see appendix): multiplier, rParam, pParam

H68 = the first 68 bytes of the header (we don't consider, for now, nTime, nBits, nNonce, i.e. the last 12 bytes of the block header)

calculate scrypt(H68, H68, 1024* multiplier, rParam, pParam, S68, 68) and store the result in S68, an array of bytes of size 68

calculate xor of arrays S68 and H68 and store the resulting array in S68xor

concatenate S68xor and the last 12 bytes of H80, the block header; store this in S80, an array of bytes of size 80.

calculate the HASH256 of S80; store the result in S256, an array of byte of size 32.

calculate the bit mask of s256 and store in array MASK, of size 32.
MASK is the smallest number such that (mask & S256) == S256.

calculate scrypt(S256, 32, S256, 32, 1024* multiplier, rParam, pParam, SC256, 32) and store the result in SC256, an array of bytes of size 32.

calculate and of arrays SC256 and MASK and store result in array MSC256.

calculate xor of array MSC256 and S56: the resulting array of bytes of size 32 is the PoW.

Appendix – block header format

the format of the header for MediterraneanCoin is the same as Bitcoin/Litecoin:

```
int nVersion;        // 4 bytes
uint256 hashPrevBlock;  // 32 bytes
uint256 hashMerkleRoot; // 32 bytes
unsigned int nTime;  // 4 bytes
unsigned int nBits;  // 4 bytes
unsigned int nNonce; // 4 bytes
```

The "compact" format of nBits is a representation of a whole
number N using an unsigned 32bit number similar to a floating point format.
The most significant 8 bits are the unsigned exponent of base 256.
This exponent can be thought of as "number of bytes of N".
The lower 23 bits are the mantissa.
Bit number 24 (0x800000) represents the sign of N.
N = (-1^sign) * mantissa * 256^(exponent-3)

Appendix a – data array for scrypt parameters

(only the first three column; the fourth column contains an estimation of the computational difficulty of the scrypt algorithm with those parameters).

int nBits = header[72-75]; // as in Bitcoin block

// take the exponent
```
int nSize = nBits >> 24;
```

// take a set of parameters for the two scrypt stages
```
int pos = 271 - 1 - nSize;
```

// here are the scrypt parameters;
// the harder becomes the difficulty, the harder become the scrypt stages, besides the hash256 stage.

```
int multiplier = dataFinal[pos][0];
int rParam = dataFinal[pos][1];
int pParam = dataFinal[pos][2];

static double dataFinal [][4] = {
            { 1 , 8 , 7 , 0.063 },
            { 8 , 7 , 1 , 0.063 },
            { 2 , 4 , 7 , 0.061 },
            { 1 , 4 , 14 , 0.06 },
            { 1 , 14 , 4 , 0.062 },
            { 4 , 2 , 7 , 0.061 },
            { 8 , 1 , 7 , 0.062 },
            { 1 , 5 , 11 , 0.059 },
            { 2 , 7 , 4 , 0.065 },
            { 2 , 14 , 2 , 0.062 },
            { 4 , 7 , 2 , 0.061 },
            { 4 , 1 , 14 , 0.063 },
            { 4 , 14 , 1 , 0.063 },
            { 2 , 2 , 14 , 0.062 },
            { 1 , 10 , 6 , 0.066 },
            { 1 , 12 , 5 , 0.066 },
            { 2 , 3 , 10 , 0.069 },
            { 2 , 5 , 6 , 0.066 },
            { 2 , 6 , 5 , 0.068 },
            { 2 , 10 , 3 , 0.065 },
            { 2 , 15 , 2 , 0.067 },
            { 1 , 4 , 15 , 0.066 },
            { 1 , 15 , 4 , 0.071 },
            { 4 , 1 , 15 , 0.067 },
            { 4 , 3 , 5 , 0.068 },
            { 2 , 2 , 15 , 0.066 },
            { 4 , 5 , 3 , 0.066 },
            { 1 , 6 , 10 , 0.067 },
            { 1 , 7 , 9 , 0.073 },
            { 1 , 9 , 7 , 0.068 },
            { 1 , 5 , 12 , 0.066 },
            { 4 , 15 , 1 , 0.067 },
            { 1 , 8 , 8 , 0.07 },
            { 4 , 8 , 2 , 0.071 },
            { 2 , 4 , 8 , 0.072 },
            { 2 , 8 , 4 , 0.07 },
            { 4 , 2 , 8 , 0.071 },
            { 4 , 4 , 4 , 0.072 },
            { 1 , 6 , 11 , 0.072 },
            { 8 , 4 , 2 , 0.071 },
            { 1 , 11 , 6 , 0.073 },
            { 1 , 13 , 5 , 0.071 },
            { 2 , 11 , 3 , 0.074 },
            { 8 , 1 , 8 , 0.073 },
            { 32 , 1 , 2 , 0.073 },
            { 32 , 2 , 1 , 0.076 },
            { 2 , 3 , 11 , 0.073 },
            { 16 , 1 , 4 , 0.072 },
            { 8 , 8 , 1 , 0.071 },
            { 16 , 2 , 2 , 0.074 },
            { 16 , 4 , 1 , 0.071 },
            { 1 , 5 , 13 , 0.071 },
            { 1 , 10 , 7 , 0.077 },
            { 1 , 14 , 5 , 0.079 },
            { 2 , 7 , 5 , 0.082 },
            { 8 , 2 , 4 , 0.071 },
            { 1 , 9 , 8 , 0.079 },
            { 2 , 3 , 12 , 0.079 },
            { 2 , 4 , 9 , 0.081 },
            { 2 , 5 , 7 , 0.08 },
            { 2 , 9 , 4 , 0.078 },
            { 2 , 12 , 3 , 0.079 },
            { 1 , 5 , 14 , 0.083 },
            { 1 , 6 , 12 , 0.079 },
            { 2 , 6 , 6 , 0.079 },
            { 4 , 2 , 9 , 0.08 },
            { 4 , 6 , 3 , 0.078 },
            { 64 , 1 , 1 , 0.075 },
            { 1 , 8 , 9 , 0.083 },
            { 1 , 12 , 6 , 0.082 },
            { 4 , 9 , 2 , 0.087 },
            { 8 , 1 , 9 , 0.1 },
            { 8 , 3 , 3 , 0.082 },
            { 4 , 3 , 6 , 0.079 },
            { 8 , 9 , 1 , 0.08 },
            { 1 , 7 , 10 , 0.078 },
            { 1 , 5 , 15 , 0.084 },
            { 1 , 15 , 5 , 0.084 },
            { 1 , 11 , 7 , 0.088 },
            { 1 , 6 , 13 , 0.085 },
            { 1 , 7 , 11 , 0.085 },
            { 1 , 8 , 10 , 0.09 },
            { 1 , 13 , 6 , 0.086 },
            { 2 , 3 , 13 , 0.096 },
            { 2 , 13 , 3 , 0.086 },
            { 8 , 5 , 2 , 0.088 },
            { 1 , 10 , 8 , 0.087 },
            { 2 , 5 , 8 , 0.087 },
            { 2 , 8 , 5 , 0.089 },
            { 2 , 4 , 10 , 0.093 },
            { 4 , 4 , 5 , 0.087 },
            { 8 , 2 , 5 , 0.088 },
            { 1 , 9 , 9 , 0.09 },
            { 2 , 10 , 4 , 0.089 },
            { 4 , 2 , 10 , 0.089 },
            { 4 , 5 , 4 , 0.089 },
            { 4 , 10 , 2 , 0.086 },
            { 8 , 1 , 10 , 0.09 },
            { 8 , 10 , 1 , 0.088 },
            { 16 , 1 , 5 , 0.091 },
            { 1 , 7 , 12 , 0.113 },
            { 2 , 14 , 3 , 0.102 },
            { 1 , 6 , 14 , 0.119 },
            { 1 , 14 , 6 , 0.098 },
            { 4 , 7 , 3 , 0.116 },
            { 16 , 5 , 1 , 0.099 },
            { 4 , 3 , 7 , 0.108 },
            { 1 , 12 , 7 , 0.111 },
            { 2 , 3 , 14 , 0.112 },
            { 2 , 7 , 6 , 0.096 },
            { 1 , 8 , 11 , 0.104 },
            { 4 , 11 , 2 , 0.108 },
            { 1 , 11 , 8 , 0.104 },
            { 2 , 6 , 7 , 0.092 },
            { 2 , 11 , 4 , 0.099 },
            { 2 , 15 , 3 , 0.097 },
            { 1 , 6 , 15 , 0.098 },
            { 1 , 9 , 10 , 0.098 },
            { 1 , 7 , 13 , 0.102 },
            { 1 , 15 , 6 , 0.104 },
            { 2 , 4 , 11 , 0.102 },
            { 2 , 5 , 9 , 0.102 },
            { 4 , 2 , 11 , 0.096 },
            { 1 , 10 , 9 , 0.101 },
            { 2 , 3 , 15 , 0.1 },
            { 2 , 9 , 5 , 0.095 },
            { 8 , 1 , 11 , 0.1 },
            { 1 , 13 , 7 , 0.098 },
            { 2 , 12 , 4 , 0.107 },
            { 4 , 12 , 2 , 0.106 },
            { 8 , 11 , 1 , 0.128 },
            { 8 , 4 , 3 , 0.11 },
            { 2 , 6 , 8 , 0.119 },
            { 2 , 8 , 6 , 0.108 },
            { 2 , 4 , 12 , 0.123 },
            { 4 , 8 , 3 , 0.118 },
            { 1 , 12 , 8 , 0.117 },
            { 4 , 2 , 12 , 0.116 },
            { 8 , 2 , 6 , 0.127 },
            { 8 , 6 , 2 , 0.114 },
            { 16 , 3 , 2 , 0.124 },
            { 4 , 6 , 4 , 0.119 },
            { 8 , 1 , 12 , 0.128 },
            { 1 , 9 , 11 , 0.118 },
            { 2 , 10 , 5 , 0.126 },
            { 8 , 3 , 4 , 0.119 },
            { 1 , 7 , 14 , 0.114 },
            { 1 , 8 , 12 , 0.113 },
            { 4 , 3 , 8 , 0.118 },
            { 4 , 4 , 6 , 0.124 },
            { 16 , 1 , 6 , 0.124 },
            { 1 , 10 , 10 , 0.114 },
            { 1 , 11 , 9 , 0.113 },
            { 2 , 5 , 10 , 0.114 },
            { 16 , 2 , 3 , 0.115 },
            { 16 , 6 , 1 , 0.122 },
            { 2 , 7 , 7 , 0.13 },
            { 32 , 1 , 3 , 0.123 },
            { 1 , 8 , 13 , 0.131 },
            { 2 , 13 , 4 , 0.118 },
            { 4 , 5 , 5 , 0.113 },
            { 4 , 13 , 2 , 0.118 },
            { 32 , 3 , 1 , 0.129 },
            { 1 , 14 , 7 , 0.112 },
            { 1 , 7 , 15 , 0.124 },
            { 8 , 12 , 1 , 0.127 },
            { 1 , 15 , 7 , 0.131 },
            { 1 , 13 , 8 , 0.121 },
            { 4 , 2 , 13 , 0.134 },
            { 8 , 1 , 13 , 0.131 },
            { 1 , 9 , 12 , 0.142 },
            { 4 , 3 , 9 , 0.145 },
            { 4 , 9 , 3 , 0.188 },
            { 2 , 4 , 13 , 0.145 },
            { 1 , 10 , 11 , 0.128 },
            { 1 , 12 , 9 , 0.126 },
            { 2 , 11 , 5 , 0.146 },
            { 8 , 13 , 1 , 0.138 },
            { 1 , 11 , 10 , 0.13 },
            { 2 , 5 , 11 , 0.128 },
            { 1 , 8 , 14 , 0.142 },
            { 2 , 4 , 14 , 0.127 },
            { 2 , 14 , 4 , 0.131 },
            { 1 , 14 , 8 , 0.153 },
            { 2 , 6 , 9 , 0.129 },
            { 2 , 8 , 7 , 0.147 },
            { 2 , 9 , 6 , 0.14 },
            { 4 , 4 , 7 , 0.136 },
            { 8 , 7 , 2 , 0.138 },
            { 8 , 2 , 7 , 0.163 },
            { 2 , 7 , 8 , 0.137 },
            { 8 , 1 , 14 , 0.138 },
            { 4 , 7 , 4 , 0.136 },
            { 4 , 14 , 2 , 0.138 },
            { 16 , 1 , 7 , 0.158 },
            { 1 , 9 , 13 , 0.142 },
            { 4 , 2 , 14 , 0.133 },
            { 1 , 13 , 9 , 0.14 },
            { 4 , 10 , 3 , 0.141 },
            { 1 , 8 , 15 , 0.147 },
            { 2 , 15 , 4 , 0.148 },
            { 1 , 10 , 12 , 0.138 },
            { 2 , 5 , 12 , 0.141 },
            { 2 , 10 , 6 , 0.142 },
            { 4 , 6 , 5 , 0.134 },
            { 8 , 14 , 1 , 0.14 },
            { 1 , 12 , 10 , 0.135 },
            { 1 , 15 , 8 , 0.14 },
            { 2 , 12 , 5 , 0.144 },
            { 2 , 6 , 10 , 0.145 },
            { 1 , 11 , 11 , 0.141 },
            { 16 , 7 , 1 , 0.131 },
            { 2 , 4 , 15 , 0.143 },
            { 4 , 2 , 15 , 0.137 },
            { 4 , 3 , 10 , 0.144 },
            { 4 , 5 , 6 , 0.14 },
            { 8 , 1 , 15 , 0.144 },
            { 8 , 5 , 3 , 0.142 },
            { 4 , 15 , 2 , 0.142 },
            { 2 , 7 , 9 , 0.141 },
            { 1 , 9 , 14 , 0.164 },
            { 2 , 9 , 7 , 0.175 },
            { 1 , 14 , 9 , 0.165 },
            { 2 , 8 , 8 , 0.181 },
            { 2 , 13 , 5 , 0.148 },
            { 4 , 4 , 8 , 0.168 },
            { 8 , 4 , 4 , 0.155 },
            { 1 , 10 , 13 , 0.142 },
            { 1 , 13 , 10 , 0.144 },
            { 2 , 5 , 13 , 0.14 },
            { 8 , 15 , 1 , 0.133 },
            { 8 , 2 , 8 , 0.152 },
            { 8 , 3 , 5 , 0.146 },
            { 1 , 12 , 11 , 0.144 },
            { 2 , 11 , 6 , 0.157 },
            { 4 , 8 , 4 , 0.141 },
            { 4 , 3 , 11 , 0.154 },
            { 8 , 8 , 2 , 0.142 },
            { 16 , 4 , 2 , 0.164 },
            { 1 , 11 , 12 , 0.148 },
            { 16 , 1 , 8 , 0.159 },
            { 16 , 2 , 4 , 0.146 },
            { 4 , 11 , 3 , 0.146 },
            { 2 , 6 , 11 , 0.141 },
            { 1 , 9 , 15 , 0.147 },
            { 32 , 2 , 2 , 0.141 },
            { 1 , 15 , 9 , 0.148 },
            { 16 , 8 , 1 , 0.144 },
            { 32 , 1 , 4 , 0.161 },
            { 64 , 2 , 1 , 0.163 },
            { 32 , 4 , 1 , 0.145 },
            { 1 , 10 , 14 , 0.154 },
            { 4 , 7 , 5 , 0.163 },
            { 1 , 14 , 10 , 0.17 },
            { 2 , 5 , 14 , 0.162 },
            { 4 , 5 , 7 , 0.159 },
            { 64 , 1 , 2 , 0.16 },
            { 2 , 7 , 10 , 0.158 },
            { 2 , 14 , 5 , 0.154 },
            { 4 , 9 , 4 , 0.161 },
            { 1 , 11 , 13 , 0.155 },
            { 1 , 12 , 12 , 0.162 },
            { 2 , 8 , 9 , 0.155 },
            { 2 , 10 , 7 , 0.153 },
            { 4 , 4 , 9 , 0.166 },
            { 2 , 6 , 12 , 0.157 },
            { 2 , 12 , 6 , 0.167 },
            { 4 , 12 , 3 , 0.165 },
            { 2 , 9 , 8 , 0.166 },
            { 1 , 13 , 11 , 0.159 },
            { 4 , 3 , 12 , 0.171 }

   };
```