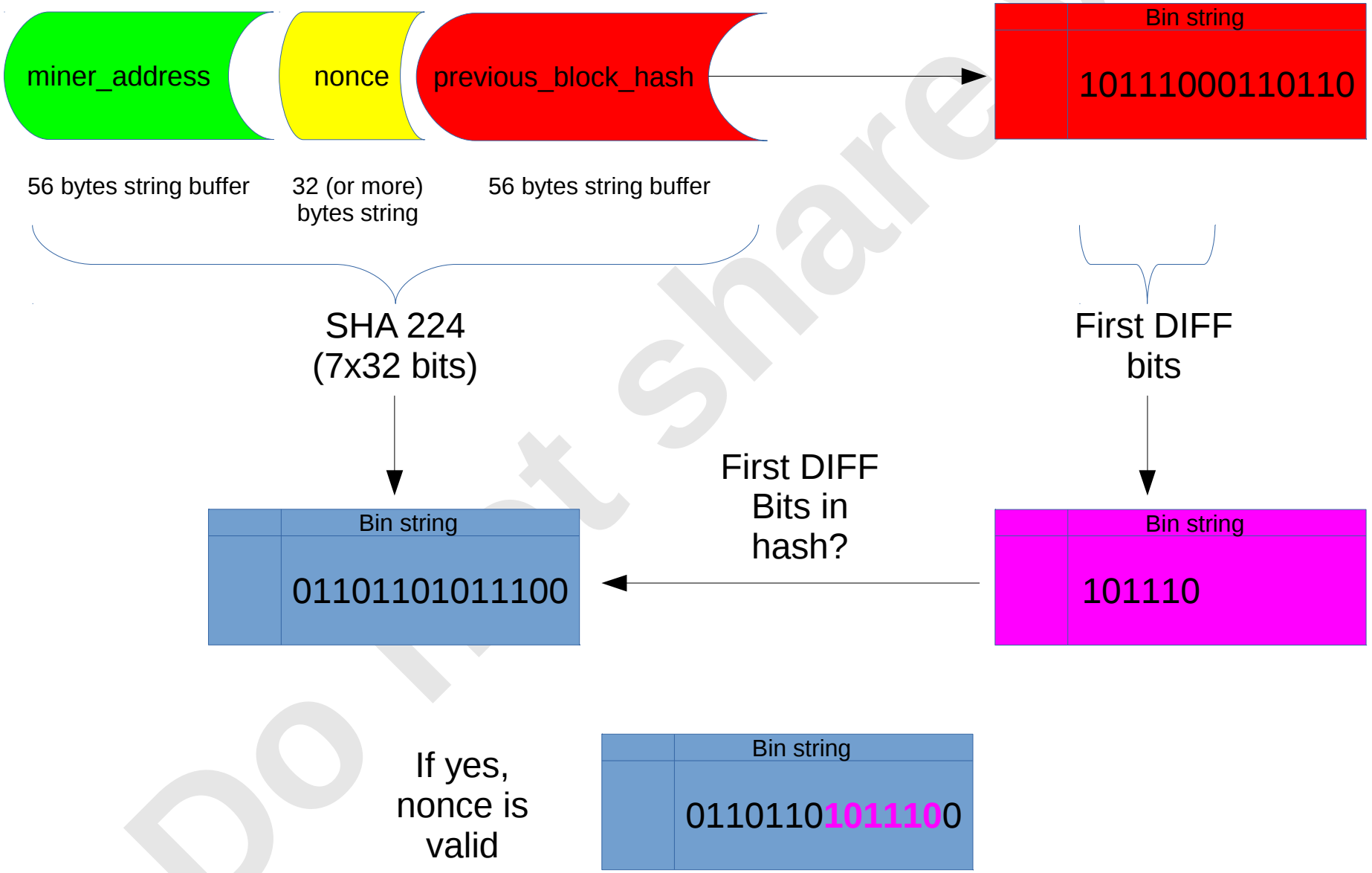
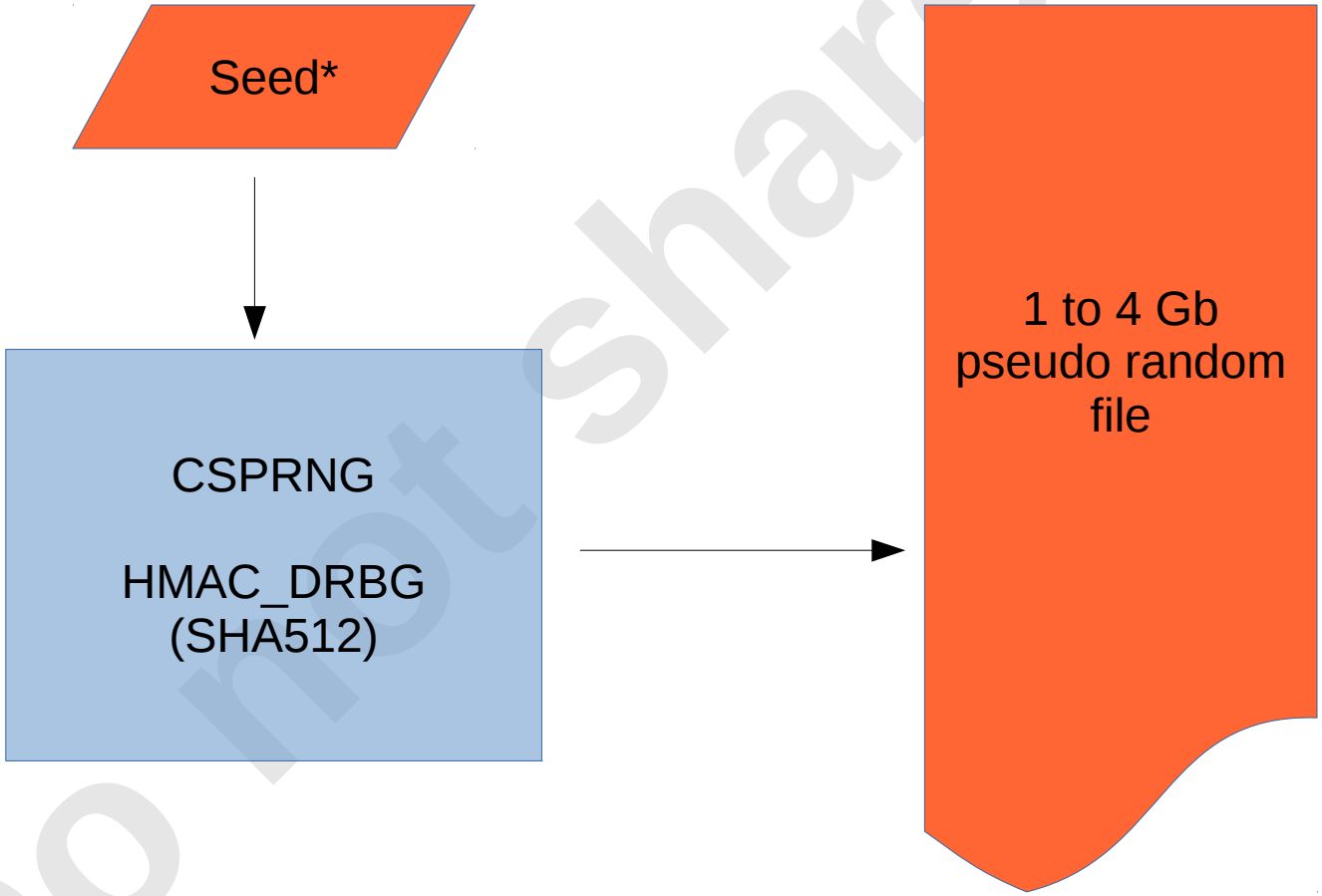
Bismuth Legacy Mining Algorithm

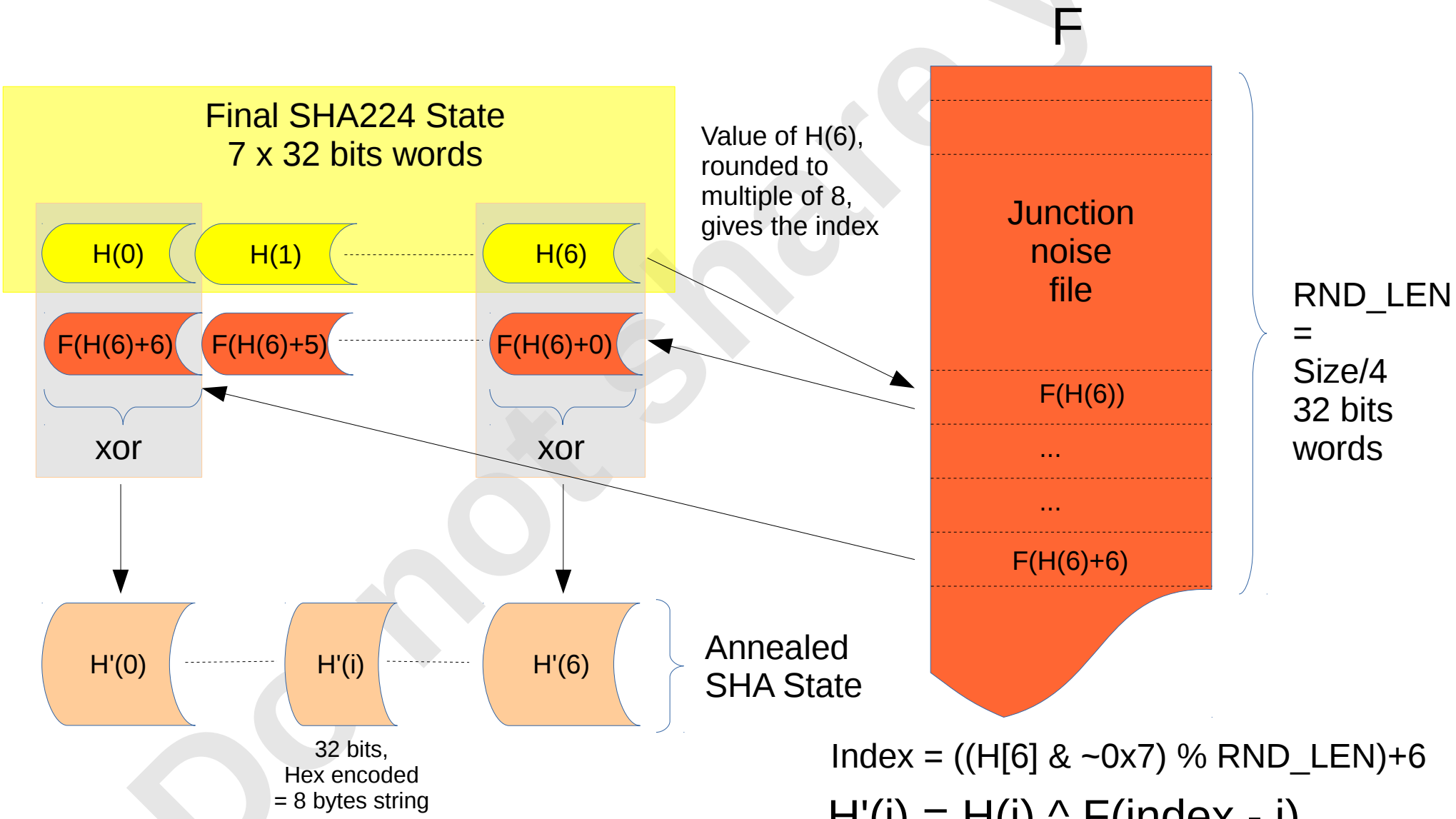


Bismuth Heavy3 Algorithm
Junction noise file



* Seed and file size can change with time.

Bismuth Heavy3 Algorithm
SHA Annealing

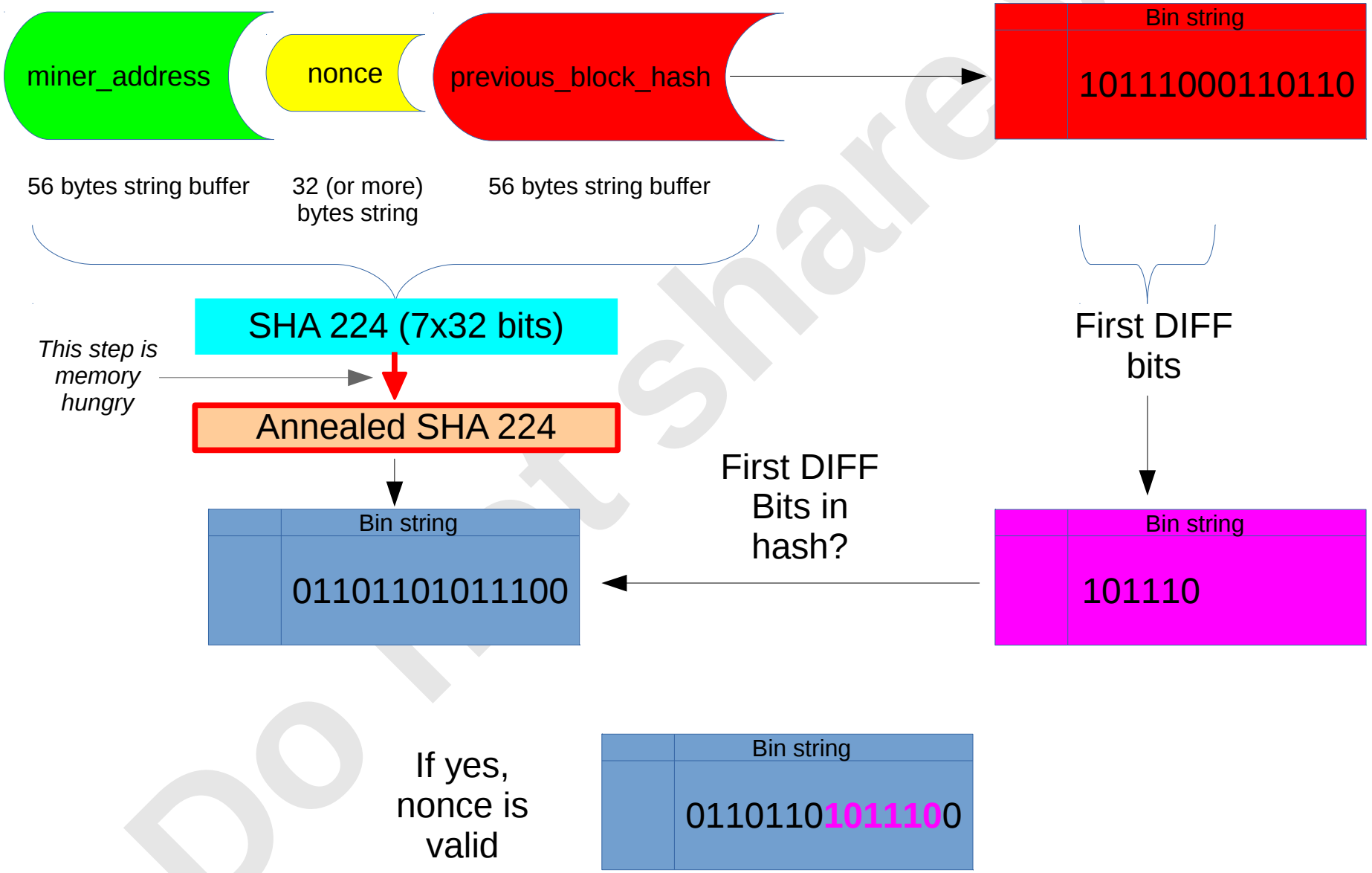


Index = ((H[6] & ~0x7) % RND_LEN)+6

$$H'(i) = H(i) \wedge F(index - i)$$

for i from 0 to 6

Bismuth Heavy3 Algorithm



Bismuth Heavy3 Algorithm OpenCL Code

Between the sha state computation and the diff matching, just call anneal3(MAP, state)
With MAP being the pointer to the rnd.bin file content.

```
inline void anneal3(const __global uint* map, uint* state){
    int index = ((state[6] & ~0x7) % RND_LEN) + 6;
    #pragma unroll
    for (int i = 0; i < 7; ++i) {
        state[i] ^= map[index - i];
    }
}
```

Bismuth Heavy3 Algorithm Test vector

```
__kernel void annealing_kernel(  __global const uint* MAP) {
    printf("__kernel annealing_kernel\n");
    uint state[8];
    // Test vector 0x7a0f384876aca3871adbde8622a87f8b971ede0ed8ee10425e3958a1
    state[0] = 0x7a0f3848;
    state[1] = 0x76aca387;
    state[2] = 0x1adbde86;
    state[3] = 0x22a87f8b;
    state[4] = 0x971ede0e;
    state[5] = 0xd8ee1042;
    state[6] = 0x5e3958a1;
    printf("Start State\n");
    for (int i = 0; i < 7; ++i) {
        printf(" %08x", state[i]);
    }
    printf("\n");
    anneal3(MAP, state);
    printf("End State\n");
    for (int i = 0; i < 7; ++i) {
        printf(" %08x", state[i]);
    }
    // Expected anneal3 : ede9b329 5cdf502d 04100269 aac2533e 43c1815e e1f8f4f6 dc5e0e87
}
```