I. JAVASCRIPT MINING CODE ANAYLSIS

채굴 연산을 위해서 기본적으로 마이닝 풀에 연결이 되어야 한다. 마이닝 풀로부터 받은 Job 및 Verify에 대해서 연산을 하고 결과 값을 마이닝 풀에 보내준다. 해당 결과 값에 대해서 정상적으로 계산에 성공했다면 보상을 받는 형태가 된다. 따라서 이러한 과정에 대해 실시간 성이 중요하고, 연산을 다른 채굴자보다 빠르게 수행하여 제출을 해야한다.

현재 자바스크립트로 작성되어있는 채굴 코드에서 기본적으로 두 방식에서 웹소켓을 이용한 프록시 형태로 마이닝 풀로 연결이 된다. 이후 Thread를 사용하는 방식 및 채굴 알고리즘으로 두가지로 분류가 가능하다.

A. Using Web Worker

Web Worker를 사용하는 형태가띄는 대표적인 웹 마이닝 코드로 Coinhive가 있다. Web Worker를 이용하여 채굴을 수행하되, Web Worker상에서 사용되는 해시 알고리즘은 Web Assembly로 컴파일 된 CryptoNight 알고리즘을 이용한다. 이러한 방식의 코드는 CryptoNoter라는 오픈소스로도 공개되어 있어, 다양한 변종 형태가 존재한다.

코드의 구조를 그림으로 정리하면 Figure 1와 같다.

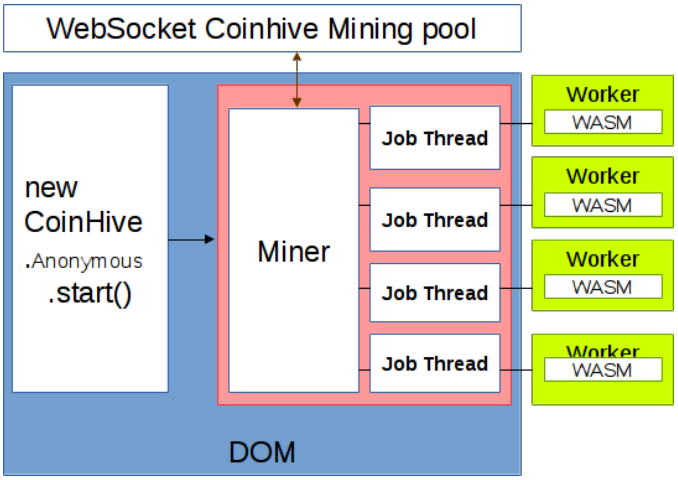

Fig. 1. Coinhive's Code Object Relationship Diagram

초기화 코드를 통해서 생성되는 오브젝트는 Miner이다. Miner에서는 웹 소켓와의 통신과 Job Thread 오브젝트를 관리하는데 목적을 갖고 있다. 초기 웹 소켓 연결에서 코드 게시자의 SiteKey에 대해 인증을 수행하고, 정상적으로 인증이 수행된 후엔 마이닝 풀로부터 작업을 받아 Job Thread에 분배를 한다.

Job Thread가 생성되는 갯수는 CPU의 코어의 갯수에 따라간다. Miner를 통해서 생성된 Job Thread에는 Miner에서 전달하는 작업과 실질적인 연산을 수행하는 Web Worker간에 중재 역할을 한다. 각 Job Thread 오브젝트에서는 하나의 Web Worker를 갖는다. setJob 및 verify 메소드를 통해서 Web Worker에 작업을 전달하고, onReceiveMsg를 통해서 연산한 결과를 Web Worker로부터 받아 Minger에 전달을 한다.

```
CryptonightWASMWrapper.prototype.onMessage =
    (function(msg) {
  var job = msg.data;
  if (job.verify_id) {
      this.verify(job);
      return
  }
  if (!this.currentJob ||
      this.currentJob.job_id !== job.job_id)
      {
      this.setJob(job)
  }
  if (job.throttle) {
      this.throttleWait = 1 / (1 -
          job.throttle) - 1;
      this.throttledStart = this.now();
      this.throttledHashes = 0;
      this.workThrottled()
  } else {
      this.work()
  }
});
```

Listing 1. onMeessage Method on Web Worker code

Web Worker는 실질적으로 연산을 하게 되는 부분으로 Web Worker내에서 Web Assembly로 컴파일된 CryptoNight 알고리즘이 동작을 하며 onMessage를 통해서 작업을 전달 받고 전달 받은 작업에 맞게 verify나 work 메소드를 호출하게 된다. 각 메소드 내에서 연산을 한 후 결과 값을 postMessage 함수를 통해서 Job Thread의 onReceiveMsg를 호출한다.

여기서 postMessage 함수 및 addEventHandler('message', callback)을 이용하여 Job thread 오브젝트와 Web Worker간에 데이터를 주고 받는데 Job Thread가 존재하는 DOM Scope와 Worker의 Scope가 별개이기 때문이다.

채굴 알고리즘 Web Assembly로 컴파일하여 사용하는 이유로 Javascript로 작성된 코드가 실행되는 것보다 Web Assembly에서 작성된 코드가 약 84배 빠른 속도를 보여주기 때문이다[1].

Cryptonight 알고리즘[2]은 proof-of-work 알고리즘중 하나이다. 이는 느린 메모리에 대한 Random Access에 의존하고 있고, 이 때문에 높은 대역폭을 갖고 있는 GDDR5 메모리를 사용하는 GPU 채굴보다는 L3 Cache를 이용하는 CPU에 대해서 최적화가 되어 있다.

CryptoNight를 소개한 이유로 현재 Browser-based mining에서 널리 사용되고 있기 때문이다. 2014년에 있었던 MineChruch에서는 litecoin에서 채용한 scrypt알고리즘 및 feathercoin에서 채용한 neo scrypt알고리즘 등이 사용되었었고, Using Web Worker에서 소개한 방식와 비슷하다. MineChruch에서 사용한 코드는 WebArchive.org[3]에서 확인할 수 있다. 따라서 코인별로 채택한 알고리즘에 따라 Web Worker에서 동작할 알고리즘을 선택할 수 있다.

2018년 2월 21일부터 크롬, 파이어 폭스 및 Anti-virus 회사의 Ad blocker에서 Web Assembly가 적용된 Javascript mining code를 탐지 및 차단을 시작했다고 알려졌다[4].

B. Using Web Crypto

Web Crypto를 이용하는 방식은 앞선 방식에서 알고리즘 부분만 바뀔 수 있지만 현재 JSECoin에서 독자적으로 사용하

[1]https://medium.com/@BenedekGagyi/the-simplest-way-to-get-started-with-webassembly-1f92f6f90d24
[2]https://en.bitcoin.it/wiki/CryptoNight
[3]https://web.archive.org/web/20160330224656/https://minecrunch.co/web/miner.js
[4]https://bitcointalk.org/index.php?topic=3019903.0

고 있는 방식이기에 따로 소개한다. 앞선 방식과 다르게 Web Worker를 사용하지 않고 기본적으로 브라우저에서 제공하는 기본적인 Threading에 의존한다.

```
function cryptoSha256(str, nonce) {
   var buffer = textEncoderUTF8(str);
   return crypto.subtle.digest("SHA-256",
       buffer).then(function(hash) {
      return hex(hash) + "," + nonce
   })
}
```

Listing 2. Native sha256 function in JSECoin Code

채굴시 사용하는 알고리즘은 Web assembly로 작성된 알고리즘이 아닌 Web Crypto에 있는 sha256을 이용한다. 브라우저에서 작성된 Native한 함수를 사용하기 때문에 Web Assembly를 통해서 얻는 빠른 속도의 장점을 취한다는 것이다. 물론 JSECoin에서 사용하는 코드에서 Web Crypto를 지원하지 않을 경우에도 동작이 가능하도록 fallbackSHA256 함수가 작성되어 있지만, 채굴시 해당 해시 값을 선점해야 한다는 점에서 이득을 보지 못한다. 또한 좀더 최적화된 방식으로 개선하고 싶어도 브라우저에 의존적이라는 점에서 코인 개발자가 취할 수 있는 방법이 존재하지 않는다.

코인을 채굴하는 것과 별개로, JSECoin에서는 VPN을 통한 채굴 및 몰래 삽입한 채굴을 방지하기 위한 코드가 있어 사용자의 동의를 받지 않고 불법적으로 채굴을 수행하는 것을 막고자 노력한 것으로 보인다.