# Appendix

https://github.com/projectblink

February 15, 2023 (Updated)

**Algorithm 1:** ZK IHR Circuit

```
/* Public signals                                          */
signal input: node_ihr
signal input: ihr_hash
/* Private signals                                         */
signal input: salt
signal input: required_ihr
/* Output signal                                           */
signal output: if_pass
/* Range proof check                                       */
signal buffer
signal range_check
if node_ihr > required_ihr - buffer && node_ihr < required_ihr +
  buffer then
    range_check = true
end
/* Verify hash                                             */
signal hash
signal hash_check
/* RIPEMD160 to calculate the hash                         */
hash = RIPEMD160 (salt, required_ihr)
if hash == ihr_hash then
    hash_check = true
end
if range_check && hash_check then
    if_pass = true
else
    if_pass = false
end
/* Bandwidth circuit ≡ IHR circuit                         */
```

Merkle Chain

**Algorithm 2:** class MerkleChain

```
pre: the snip is added to the data
post: the data is added to the chain
begin
    add_node(snip)
    d ← snip
    if head = null then
        head,tail ← add_data(d)
    else
        tail ← add_data(d)
    end
end
```

**Algorithm 3:** class add_data(d)

```
pre: the value is added to the vector
post: the vector is generated to a merkle tree and added to the chain
begin
    New Vector data
    data ← d
    if size(data) == max_block_size then
        generate_root(data)
    end
end
```

**Algorithm 4:** generate_root()

```
pre: the vector data is added as the leaves
post: merkel tree and its root is generated
begin
    New Vector temp_data
    temp_data ← data
    while temp_data > 1 do
        for i = 0 i < size(temp_data) i+2 do
            Left ← temp_data[i]
            Right ← (i+1 == size(temp_data)) ? temp_data[i] :
              temp_data[i+1]
            combined = Left + Right
            new_temp_data ← hash(combined)
        end
        temp_data ← new_temp_data
    end
    node_root ← temp_data[0]
end
```

**Algorithm 5:** main()

```
initialized: chain is a object of class MerkleChain and string data
begin
    while true do
        Output "enter data (q to quit)" Get data
        if data = q then
            Break
        else
            addnode(data)
        end
    end
end
```

**Algorithm 6:** Node Weights

Algo

**Algorithm 7:** Snip Construction

Algo

**Algorithm 8:** Hash Proofing

Algo

Tokens should be traded for bitcoins inorder for nodes to fix its market price which will assist in facilitating its transactions, per block stake requirement, non-accepted token producer commission, etc as every procedure follows up with denominations in bitcoin. For regulators they can select a specific any token id, can also possibly be their fiat currency as a L1 token on bitcoin for taxing oppurtunities on profits (capital gains).