参数配置与电路设计：

1.参数配置：

输出长度 n = 256 bits

状态大小 t = 3（包含两个输入元素和一个容量元素）

S-box 指数 d = 5

2.电路设计：

隐私输入为哈希原象

公开输入为预期的哈希值

电路验证计算出的哈希值与公开输入的哈希值一致

使用 Groth16 算法生成证明

3.脚本文件：

Generate_input.js:生成电路输入示例

Build.sh:自动化编译电路、生成信任设置和证明的脚本

代码整体概述：

Poseidon 使用海绵结构,包括吸收(absorbing)和挤压(squeezing)阶段，代码中主要处理轮函数（round function），包括完整轮（full rounds）和部分轮（partial rounds）。轮函数由 S-box 层（非线性变换）和线性层（混合层）组成。使用 Circomlib 库（gates.ci rcom 和 bitify.ci rcom）来构建电路基本组件，如逻辑门和位操作。

1.关键部分：

常量定义（如 N、T、D）指定了哈希函数的参数

轮常量（round constants）用于轮函数的随机化

模运算模板（如 AddMod、MulMod）支持有限域上的算术

SBox 模板实现了 S-box 变换（x^D mod p）

MixLayer 模板（未完成）计划实现线性混合层

2.常量定义（Constants Definition）

```
const N = 256;        // 输出长度
const T = 3;          // 状态大小 (rate + capacity)
const RATE = T - 1;   // 输入率
const D = 5;          // S-box 指数
const ROUNDS_F = 8;   // 完全轮数
const ROUNDS_P = 4;   // 部分轮数
const TOTAL_ROUNDS = ROUNDS_F + ROUNDS_P;
```

3.轮常量（Round Constants）

```
function roundConstants(i) {
 return [
     [0x0000000000000001n,          0x0000000000000002n,
0x0000000000000003n],
     // ... 其他轮常量
 ][i];
}
```

轮常量是每轮添加到状态中的随机常数，用于打破对称性和增加随机性。这里使用大整数（BigInt）表示。

4.部分轮活跃索引（Partial Round Active Index）

partialRoundActiveIndex 函数指定了部分轮中激活 S-box 的元素索引：

```
circom
function partialRoundActiveIndex(round) {
    return [1, 2, 1, 2][round];
}
```

在部分轮（partial rounds）中，只对状态数组的部分元素应用 S-box 变换（而非全部），以提高效率。这里，轮索引 round 对应一个预定义数组，例如第 0 轮激活索引 1，第 1 轮激活索引 2。

5. 模运算辅助函数（Modular Arithmetic Templates）

处理有限域上的模运算，使用素数 p 作为域大小：

ModConstant 模板：实现模运算

```
circom
template ModConstant(n) {
    signal input in;
    signal output out;
    out <== in % n;
}
```

AddMod 模板：有限域加法

```
circom
template AddMod(p) {
```

```
    signal input a;
    signal input b;
    signal output out;
    component mod = ModConstant(p);
    mod.in <== a + b;
    out <== mod.out;
}
```

MulMod 模板：有限域乘法

```
circom
template MulMod(p) {
    signal input a;
    signal input b;
    signal output out;
    component mod = ModConstant(p);
    mod.in <== a * b;
    out <== mod.out;
}
```

ModConstant 直接计算输入模 n，而 AddMod 和 MulMod 通过组合 ModConstant 来实现。

6. 幂运算和 S-box 模板（PowMod and SBox Templates）

S-box 是 Poseidon 的核心非线性组件，使用 PowMod 实现：

```
circom
```

```
template PowMod(p, e) {
    signal input in;
    signal output out;
    if (e == 0) {
        out <== 1;
    } else {
        signal temp;
        temp <== in;
        for (var i = 1; i < e; i++) {
            component mul = MulMod(p);
            mul.a <== temp;
            mul.b <== in;
            temp <== mul.out;
        }
        out <== temp;
    }
}
template SBox(p) {
    signal input in;
    signal output out;
    component pow = PowMod(p, D);
    pow.in <== in;
```

```
    out <== pow.out;
}
```

PowMod 模板计算输入 in 的 e 次幂模 p。它使用循环和 MulMod 来实现幂运算

7. 线性变换（MixLayer Template）

```
template MixLayer(p) {
    signal input in[T];
    signal output out[T];
    // 简化的混合矩阵，实际应使用文档中定义的矩阵
    // 这里使用单位矩阵的变体作为示例
```

MixLayer 计划实现线性混合层，用于扩散状态元素。它输入大小为 T 的数组，输出变换后的数组。

参考文献：

《Poseidon2: A Faster Version of the Poseidon Hash Function》