山东大学
SHANDONG UNIVERSITY

Project3 circom 实现 poseidon2 哈希算法电路

学院 网络空间安全

专业 网络空间安全

学号 202200460149

班级姓名 网安 22.1 班张弛

2025 年 8 月 5 日

目录

1 实验任务

project 3: 用 circom 实现 poseidon2 哈希算法的电路

要求:

1) poseidon2 哈希算法参数参考参考文档 1 的 Table1，用 (n,t,d)=(256,3,5) 或 (256,2,5)

2）电路的公开输入用 poseidon2 哈希值，隐私输入为哈希原象，哈希算法的输入只考虑一个 block 即可。

3) 用 Groth16 算法生成证明

参考文档:

1. poseidon2 哈希算法 https://eprint.iacr.org/2023/323.pdf

2. circom 说明文档 https://docs.circom.io/

3. circom 电路样例 https://github.com/iden3/circomlib

2 circom 安装

跟随参考网站，将 circom 安装成功了：

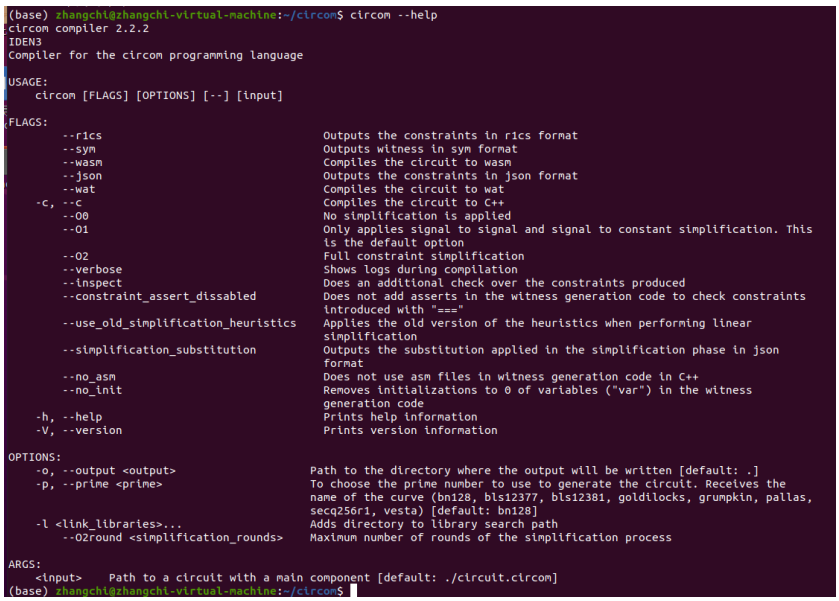

图 1 circom

3 poseidon2 哈希算法

在实现之前我们先来了解什么是 poseidon2 哈希算法：

Poseidon2 是一种专门为零知识证明系统（如 zk-SNARKs / zk-STARKs）设计的高效加密哈希函数，属于加性友好哈希函数（ARX 或 SPN）族。它是在原始 Poseidon 哈希函数的基础上提出的改进版本，目标是提高在零知识电路中的效率，降低约束复杂度。

4 poseidon2 哈希算法 circom 实现

由于 circom 官方库中有 poseidon 相关的实现，我们可以在此基础上进行修改得到 poseidon2 的算法实现：使用参数为 (n,t,d)=(256,3,5)。具体代码如下所示，而库中的 poseidon.circom 见后面的完整代码部分：

```
1 pragma circom 2.1.4;
2
3 include "/home/zhangchi/circomlib/circuits/poseidon.circom";
4
```

```
template PoseidonHash2() {
    signal input in[2];
    signal output out;

    component p = Poseidon(2);

    for (var i = 0; i < 2; i++) {
        p.inputs[i] <== in[i];
    }


    out <== p.out;
}
component main = PoseidonHash2();
```

这里有一个注意的是 include 的文件所在位置由于我用的绝对路径，所以如果路径错误，直接执行是会报错的，后续的一些命令中我也有用到绝对路径，可以相应调整。

这段代码是满足题目要求的：

公开输入用 poseidon2 哈希值 signal output out;

定义一个输出信号 out，用于输出计算结果（即 Poseidon 哈希值）。

所有 output 默认是公开的，会写入到 public.json 中作为验证条件。

隐私输入为哈希原象：signal input in[2];

哈希算法的输入只考虑一个 block：

in[2] 输入，对应 Poseidon2 的 t=3（2 个输入 + 1 padding），也就是只处理了一个 block

然后我们编译电路：

```
(base) zhangchi@zhangchi-virtual-machine:~/桌面$ circom poseidon21.circom --r1cs --wasm --sym
template instances: 72
non-linear constraints: 243
linear constraints: 274
public inputs: 0
private inputs: 2
public outputs: 1
wires: 520
labels: 771
Written successfully: ./poseidon21.r1cs
Written successfully: ./poseidon21.sym
Written successfully: ./poseidon21_js/poseidon21.wasm
Everything went okay
```

图 2 编译电路

可以看到编译成功，同时我们可以使用 snarkjs 查看电路信息，通过运行以下命令来打印电路的约束：

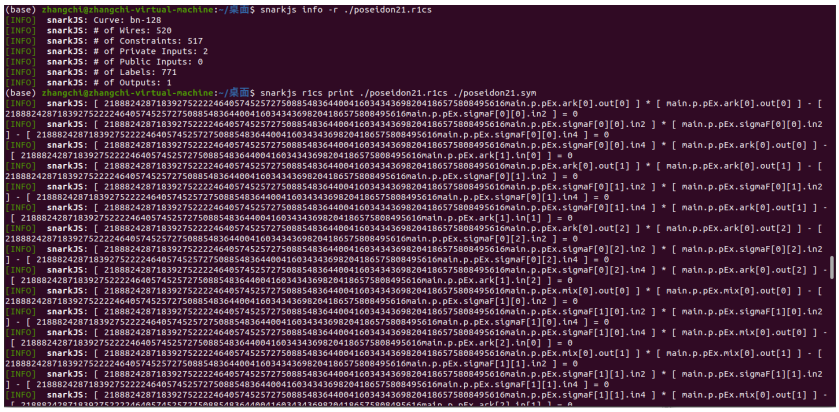

图 3 查看相关信息

接着是见证，在创建证明之前，我们需要计算与电路的所有约束匹配电路的所有信号。为此，我们将使用 circom 生成的 Wasm 模块来协助完成这项工作。

使用生成的 Wasm 二进制文件和三个 JavaScript 文件，我们只需提供一个包含输入的文件，模块将执行电路并计算所有中间信号和输出。输入、中间信号和输出的集合称为见证。我们创建了一个名为 input.json 的文件作为输入，内容如下：

```
{
  "in": ["123", "456"]
}
```

接着我们使用 WebAssembly 计算见证：

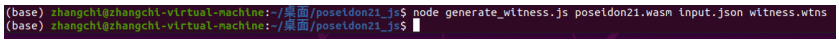

图 4 计算见证

然后我们就可以利用 Groth16 来生成证明：

我们需要生成可信设置，下面是阶段 1：

接着是阶段 2 运行如下命令：

```
snarkjs powersoftau prepare phase2 pot12_0001.ptau pot12_final.ptau -v
snarkjs groth16 setup /home/zhangchi/桌面/poseidon21.r1cs pot12_final.ptau
    poseidon21_0000.zkey
snarkjs zkey contribute poseidon21_0000.zkey poseidon21_0001.zkey --name="1
    st Contributor Name" -v
#导出验证密钥
snarkjs zkey export verificationkey multiplier2_0001.zkey verification_key.
    json
```

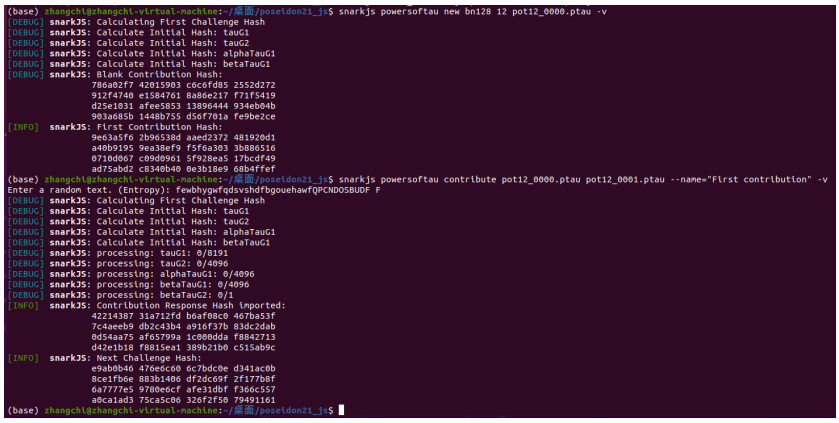

图 5 生成可信设置

一旦计算了见证并且已经执行了可信设置，我们就可以生成一个与电路和见证相关联的 zk-proof：

```
snarkjs groth16 prove poseidon21_0001.zkey witness.wtns proof.json public.
    json
#此命令生成一个 Groth16 证明并输出两个文件：
#proof.json：它包含证明。
#public.json：它包含公共输入和输出的值。

#验证
snarkjs groth16 verify verification_key.json public.json proof.json
```

一个有效的证明不仅证明我们知道一组满足电路的信号，而且证明我们使用的公共输入和输出与文件中描述的信号相匹配。

最终结果如图所示，可以看到验证成功：

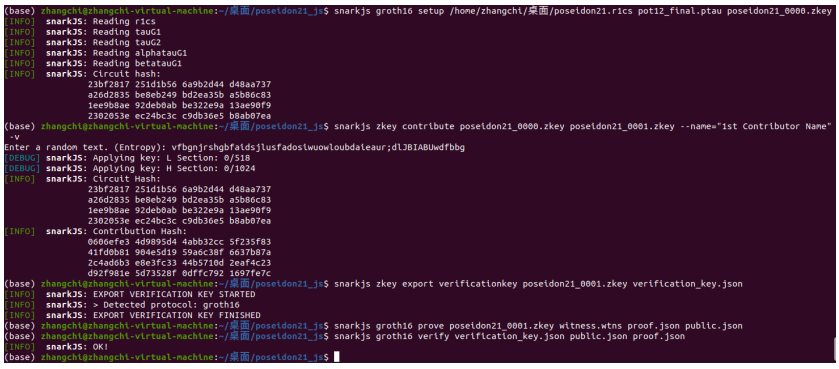

图 6 验证成功

5 完整代码

```
pragma circom 2.0.0;

include "./poseidon_constants.circom";

template Sigma() {
    signal input in;
    signal output out;

    signal in2;
    signal in4;

    in2 <== in*in;
    in4 <== in2*in2;

    out <== in4*in;
}

template Ark(t, C, r) {
    signal input in[t];
    signal output out[t];

    for (var i=0; i<t; i++) {
        out[i] <== in[i] + C[i + r];
    }
}

template Mix(t, M) {
    signal input in[t];
    signal output out[t];

    var lc;
    for (var i=0; i<t; i++) {
        lc = 0;
        for (var j=0; j<t; j++) {
            lc += M[j][i]*in[j];
        }
        out[i] <== lc;
    }
}

template MixLast(t, M, s) {
    signal input in[t];
```

```
    signal output out;

    var lc = 0;
    for (var j=0; j<t; j++) {
        lc += M[j][s]*in[j];
    }
    out <== lc;
}

template MixS(t, S, r) {
    signal input in[t];
    signal output out[t];


    var lc = 0;
    for (var i=0; i<t; i++) {
        lc += S[(t*2-1)*r+i]*in[i];
    }
    out[0] <== lc;
    for (var i=1; i<t; i++) {
        out[i] <== in[i] +  in[0] * S[(t*2-1)*r + t + i -1];
    }
}

template PoseidonEx(nInputs, nOuts) {
    signal input inputs[nInputs];
    signal input initialState;
    signal output out[nOuts];

    // Using recommended parameters from whitepaper https://eprint.iacr.org
        /2019/458.pdf (table 2, table 8)
    // Generated by https://extgit.iaik.tugraz.at/krypto/hadeshash/-/blob/
        master/code/calc_round_numbers.py
    // And rounded up to nearest integer that divides by t
    var N_ROUNDS_P[16] = [56, 57, 56, 60, 60, 63, 64, 63, 60, 66, 60, 65,
        70, 60, 64, 68];
    var t = nInputs + 1;
    var nRoundsF = 8;
    var nRoundsP = N_ROUNDS_P[t - 2];
    var C[t*nRoundsF + nRoundsP] = POSEIDON_C(t);
    var S[  N_ROUNDS_P[t-2]  *  (t*2-1)  ]  = POSEIDON_S(t);
    var M[t][t] = POSEIDON_M(t);
    var P[t][t] = POSEIDON_P(t);
```

```
83
84      component ark[nRoundsF];
85      component sigmaF[nRoundsF][t];
86      component sigmaP[nRoundsP];
87      component mix[nRoundsF-1];
88      component mixS[nRoundsP];
89      component mixLast[nOuts];
90
91
92      ark[0] = Ark(t, C, 0);
93      for (var j=0; j<t; j++) {
94          if (j>0) {
95              ark[0].in[j] <== inputs[j-1];
96          } else {
97              ark[0].in[j] <== initialState;
98          }
99      }
100
101     for (var r = 0; r < nRoundsF\2-1; r++) {
102         for (var j=0; j<t; j++) {
103             sigmaF[r][j] = Sigma();
104             if(r==0) {
105                 sigmaF[r][j].in <== ark[0].out[j];
106             } else {
107                 sigmaF[r][j].in <== mix[r-1].out[j];
108             }
109         }
110
111         ark[r+1] = Ark(t, C, (r+1)*t);
112         for (var j=0; j<t; j++) {
113             ark[r+1].in[j] <== sigmaF[r][j].out;
114         }
115
116         mix[r] = Mix(t,M);
117         for (var j=0; j<t; j++) {
118             mix[r].in[j] <== ark[r+1].out[j];
119         }
120
121     }
122
123     for (var j=0; j<t; j++) {
124         sigmaF[nRoundsF\2-1][j] = Sigma();
125         sigmaF[nRoundsF\2-1][j].in <== mix[nRoundsF\2-2].out[j];
```

```
126     }
127
128     ark[nRoundsF\2] = Ark(t, C, (nRoundsF\2)*t );
129     for (var j=0; j<t; j++) {
130         ark[nRoundsF\2].in[j] <== sigmaF[nRoundsF\2-1][j].out;
131     }
132
133     mix[nRoundsF\2-1] = Mix(t,P);
134     for (var j=0; j<t; j++) {
135         mix[nRoundsF\2-1].in[j] <== ark[nRoundsF\2].out[j];
136     }
137
138
139     for (var r = 0; r < nRoundsP; r++) {
140         sigmaP[r] = Sigma();
141         if (r==0) {
142             sigmaP[r].in <== mix[nRoundsF\2-1].out[0];
143         } else {
144             sigmaP[r].in <== mixS[r-1].out[0];
145         }
146
147         mixS[r] = MixS(t, S, r);
148         for (var j=0; j<t; j++) {
149             if (j==0) {
150                 mixS[r].in[j] <== sigmaP[r].out + C[(nRoundsF\2+1)*t + r];
151             } else {
152                 if (r==0) {
153                     mixS[r].in[j] <== mix[nRoundsF\2-1].out[j];
154                 } else {
155                     mixS[r].in[j] <== mixS[r-1].out[j];
156                 }
157             }
158         }
159     }
160
161     for (var r = 0; r < nRoundsF\2-1; r++) {
162         for (var j=0; j<t; j++) {
163             sigmaF[nRoundsF\2 + r][j] = Sigma();
164             if (r==0) {
165                 sigmaF[nRoundsF\2 + r][j].in <== mixS[nRoundsP-1].out[j];
166             } else {
167                 sigmaF[nRoundsF\2 + r][j].in <== mix[nRoundsF\2+r-1].out[j];
168             }
```

```
        }

        ark[ nRoundsF\2 + r + 1] = Ark(t, C,  (nRoundsF\2+1)*t + nRoundsP +
            r*t );
        for (var j=0; j<t; j++) {
            ark[nRoundsF\2 + r + 1].in[j] <== sigmaF[nRoundsF\2 + r][j].out;
        }

        mix[nRoundsF\2 + r] = Mix(t,M);
        for (var j=0; j<t; j++) {
            mix[nRoundsF\2 + r].in[j] <== ark[nRoundsF\2 + r + 1].out[j];
        }

    }

    for (var j=0; j<t; j++) {
        sigmaF[nRoundsF-1][j] = Sigma();
        sigmaF[nRoundsF-1][j].in <== mix[nRoundsF-2].out[j];
    }

    for (var i=0; i<nOuts; i++) {
        mixLast[i] = MixLast(t,M,i);
        for (var j=0; j<t; j++) {
            mixLast[i].in[j] <== sigmaF[nRoundsF-1][j].out;
        }
        out[i] <== mixLast[i].out;
    }

}

template Poseidon(nInputs) {
    signal input inputs[nInputs];
    signal output out;

    component pEx = PoseidonEx(nInputs, 1);
    pEx.initialState <== 0;
    for (var i=0; i<nInputs; i++) {
        pEx.inputs[i] <== inputs[i];
    }
    out <== pEx.out[0];
}
```