最重要的源码如下：

build/zkMeCredentialQuery/params/input.json

{

"issuerClaim":[

"379949150130214723420589610911161895495647789006649785264738141299135414272",

"3583233690122716044519380227940806650830",

"1670236457",

"18"

],

"userID":"379949150130214723420589610911161895495647789006649785264738141299135414272",

"claimSchema":"3583233690122716044519380227940806650830",

"operator":1,

"timestamp":"1670236456",

"value":[

"18",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0",

"0"

]

}

input.json用于生成witness：

node generate\_witness.js <file.wasm> <input.json> <output.wtns>

circuits/zkMeCredentialQuery.circom

pragma circom 2.0.0;

include "lib/query/zkMeCredentialQuery.circom";

component main{public [userID, claimSchema, operator, value, timestamp]} = CredentialQuery(64);

circuits/lib/query/zkMeCredentialQuery.circom

pragma circom 2.0.0**;**

include "../../../node\_modules/circomlib/circuits/mux1.circom"**;**

include "../../../node\_modules/circomlib/circuits/bitify.circom"**;**

include "../../../node\_modules/circomlib/circuits/comparators.circom"**;**

include "query.circom"**;**

template CredentialQuery**(**valueArraySize**)** **{**

/\* issuerClaim signals \*/

signal input issuerClaim**[**4**];**

/\* userID ownership signals \*/

signal input userID**;**

/\* current time \*/

signal input timestamp**;**

/\*\* Query \*/

signal input claimSchema**;**

signal input operator**;**

signal input value**[**valueArraySize**];**

// Verify that claims are issued to designated identities

issuerClaim**[**0**]** **===** userID**;**

// Verify issuerClaim schema

issuerClaim**[**1**]** **===** claimSchema**;**

// Verify the expiration time of the issue:

// timestamp < expirationComp.expiration

component lt **=** LessEqThan**(**252**);**

lt**.in[**0**]** **<==** timestamp**;**

lt**.in[**1**]** **<==** issuerClaim**[**2**];**

component res **=** Mux1**();**

res**.**c**[**0**]** **<==** 1**;**

res**.**c**[**1**]** **<==** lt**.**out**;**

res**.**s **<==** 3**;**

res**.**out **===** 1**;**

// query

component q **=** Query**(**valueArraySize**);**

q**.in** **<==** issuerClaim**[**3**];**

q**.**operator **<==** operator**;**

**for(var** i **=** 0**;** i**<**valueArraySize**;** i**++)** **{**

q**.**value**[**i**]** **<==** value**[**i**];**

**}**

q**.**out **===** 1**;**

**}**

circuits/lib/query/query.circom

pragma circom 2.0.0**;**

include "../../../node\_modules/circomlib/circuits/mux3.circom"**;**

include "../../../node\_modules/circomlib/circuits/bitify.circom"**;**

include "../../../node\_modules/circomlib/circuits/comparators.circom"**;**

include "comparators.circom"**;**

/\*

Operators:

"0" - noop, skip execution. Ignores all `in` and `value` passed to query, out 1

"1" - equals

"2" - less-than

"3" - greater-than

"4" - in

"5" - notin

\*/

template Query **(**valueArraySize**)** **{**

// signals

signal input **in;**

signal input value**[**valueArraySize**];**

signal input operator**;**

signal output out**;**

// operation components

component eq **=** IsEqual**();**

eq**.in[**0**]** **<==** **in;**

eq**.in[**1**]** **<==** value**[**0**];**

// LessThan

component lt **=** LessThan**(**252**);**

lt**.in[**0**]** **<==** **in;**

lt**.in[**1**]** **<==** value**[**0**];**

component gt **=** GreaterThan**(**252**);**

gt**.in[**0**]** **<==** **in;**

gt**.in[**1**]** **<==** value**[**0**];**

// in

component inComp **=** IN**(**valueArraySize**);**

inComp**.in** **<==** **in;**

**for(var** i **=** 0**;** i**<**valueArraySize**;** i**++){**inComp**.**value**[**i**]** **<==** value**[**i**];}**

// mux

component mux **=** Mux3**();**

component n2b **=** Num2Bits**(**3**);**

n2b**.in** **<==** operator**;**

mux**.**s**[**0**]** **<==** n2b**.**out**[**0**];**

mux**.**s**[**1**]** **<==** n2b**.**out**[**1**];**

mux**.**s**[**2**]** **<==** n2b**.**out**[**2**];**

mux**.**c**[**0**]** **<==** 1**;** // noop, skip execution

mux**.**c**[**1**]** **<==** eq**.**out**;**

mux**.**c**[**2**]** **<==** lt**.**out**;**

mux**.**c**[**3**]** **<==** gt**.**out**;**

mux**.**c**[**4**]** **<==** inComp**.**out**;**

mux**.**c**[**5**]** **<==** 1**-**inComp**.**out**;**

mux**.**c**[**6**]** **<==** 0**;** // not in use

mux**.**c**[**7**]** **<==** 0**;** // not in use

// output

out **<==** mux**.**out**;**

**}**

circuits/lib/query/comparators.circom

pragma circom 2.0.0**;**

include "../../../node\_modules/circomlib/circuits/comparators.circom"**;**

// nElements - number of value elements

// Example nElements = 3, '1' v ['12', '1231', '9999'], 1 not in array of values

template IN **(**valueArraySize**){**

signal input **in;**

signal input value**[**valueArraySize**];**

signal output out**;**

component eq**[**valueArraySize**];**

**var** count **=** 0**;**

**for** **(var** i**=**0**;** i**<**valueArraySize**;** i**++)** **{**

eq**[**i**]** **=** IsEqual**();**

eq**[**i**].in[**0**]** **<==** **in;**

eq**[**i**].in[**1**]** **<==** value**[**i**];**

count **+=** eq**[**i**].**out**;**

**}**

// Greater then

component gt **=** GreaterThan**(**252**);**

gt**.in[**0**]** **<==** count**;**

gt**.in[**1**]** **<==** 0**;**

out **<==** gt**.**out**;** // 1 - if in signal in the list, 0 - if it is not

**}**

circuits/mux3.circom

/\*

Copyright 2018 0KIMS association.

This file is part of circom (Zero Knowledge Circuit Compiler).

circom is a free software: you can redistribute it and/or modify it

under the terms of the GNU General Public License as published by

the Free Software Foundation, either version 3 of the License, or

(at your option) any later version.

circom is distributed in the hope that it will be useful, but WITHOUT

ANY WARRANTY; without even the implied warranty of MERCHANTABILITY

or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public

License for more details.

You should have received a copy of the GNU General Public License

along with circom. If not, see <https://www.gnu.org/licenses/>.

\*/

pragma circom 2.0.0**;**

template MultiMux3**(**n**)** **{**

signal input c**[**n**][**8**];** // Constants

signal input s**[**3**];** // Selector

signal output out**[**n**];**

signal a210**[**n**];**

signal a21**[**n**];**

signal a20**[**n**];**

signal a2**[**n**];**

signal a10**[**n**];**

signal a1**[**n**];**

signal a0**[**n**];**

signal a**[**n**];**

// 4 constrains for the intermediary variables

signal s10**;**

s10 **<==** s**[**1**]** **\*** s**[**0**];**

**for** **(var** i**=**0**;** i**<**n**;** i**++)** **{**

a210**[**i**]** **<==** **(** c**[**i**][** 7**]-**c**[**i**][** 6**]-**c**[**i**][** 5**]+**c**[**i**][** 4**]** **-** c**[**i**][** 3**]+**c**[**i**][** 2**]+**c**[**i**][** 1**]-**c**[**i**][** 0**]** **)** **\*** s10**;**

a21**[**i**]** **<==** **(** c**[**i**][** 6**]-**c**[**i**][** 4**]-**c**[**i**][** 2**]+**c**[**i**][** 0**]** **)** **\*** s**[**1**];**

a20**[**i**]** **<==** **(** c**[**i**][** 5**]-**c**[**i**][** 4**]-**c**[**i**][** 1**]+**c**[**i**][** 0**]** **)** **\*** s**[**0**];**

a2**[**i**]** **<==** **(** c**[**i**][** 4**]-**c**[**i**][** 0**]** **);**

a10**[**i**]** **<==** **(** c**[**i**][** 3**]-**c**[**i**][** 2**]-**c**[**i**][** 1**]+**c**[**i**][** 0**]** **)** **\*** s10**;**

a1**[**i**]** **<==** **(** c**[**i**][** 2**]-**c**[**i**][** 0**]** **)** **\*** s**[**1**];**

a0**[**i**]** **<==** **(** c**[**i**][** 1**]-**c**[**i**][** 0**]** **)** **\*** s**[**0**];**

a**[**i**]** **<==** **(** c**[**i**][** 0**]** **);**

out**[**i**]** **<==** **(** a210**[**i**]** **+** a21**[**i**]** **+** a20**[**i**]** **+** a2**[**i**]** **)** **\*** s**[**2**]** **+**

**(** a10**[**i**]** **+** a1**[**i**]** **+** a0**[**i**]** **+** a**[**i**]** **);**

**}**

**}**

template Mux3**()** **{**

**var** i**;**

signal input c**[**8**];** // Constants

signal input s**[**3**];** // Selector

signal output out**;**

component mux **=** MultiMux3**(**1**);**

**for** **(**i**=**0**;** i**<**8**;** i**++)** **{**

mux**.**c**[**0**][**i**]** **<==** c**[**i**];**

**}**

**for** **(**i**=**0**;** i**<**3**;** i**++)** **{**

s**[**i**]** **==>** mux**.**s**[**i**];**

**}**

mux**.**out**[**0**]** **==>** out**;**

**}**

本质上，逻辑为：

s10 = s[1] \* s[0];

a210[0] = ( c[0][ 7]-c[0][ 6]-c[0][ 5]+c[0][ 4] - c[0][ 3]+c[0][ 2]+c[0][ 1]-c[0][ 0] ) \* s10;

a21[0] = ( c[0][ 6]-c[0][ 4]-c[0][ 2]+c[0][ 0] ) \* s[1];

a20[0] = ( c[0][ 5]-c[0][ 4]-c[0][ 1]+c[0][ 0] ) \* s[0];

a2[0] = ( c[0][ 4]-c[0][ 0] );

a10[0] = ( c[0][ 3]-c[0][ 2]-c[0][ 1]+c[0][ 0] ) \* s10;

a1[0] = ( c[0][ 2]-c[0][ 0] ) \* s[1];

a0[0] = ( c[0][ 1]-c[0][ 0] ) \* s[0];

a[0] = ( c[0][ 0] );

out[0] = ( a210[0] + a21[0] + a20[0] + a2[0] ) \* s[2] + ( a10[0] + a1[0] + a0[0] + a[0] );

但还是不理解这个算法是什么意思。