**Design and Implementation of Lookup Table in Dark Matter PRF**

**Description:**

This document explains the method using which lookup table has been implemented in the PRF Dark Matter project. The goal of this table is to cater as a directory search of precomputed values to make “computation at run-time” faster. The implementation has two parts:

1. Creating a lookup table
2. Using the lookup table

The function that the implementation replaces is a matrix vector multiplication in Z3. The output of phase 2 is a 256 bit in Z3 and 81 X 256 bits matrix in Z3. The following steps are followed to perform matrix vector multiplication using a lookup table.

1. **Preprocessing Stage:** 
   1. The input vector of size 256 bit is a 0/1 value in Z3. The input is packed into 16 words of 16-bit integer.
   2. Generate a lookup table of size 16 X (2^16). This table contains multiplication of each of the 16-bit input with every possible integer value between 0 and (2^16 - 1). This will ensure that each of the input has been precomputed.
   3. Reformat 81 X 256 vector into 16 X 16 X 81 matrix.
2. **Lookup Table:**
   * 1. For each of the combination of matrix and vector, lookup the corresponding rows and columns of lookup table to find out the value of their multiplication.
     2. Output is an 81-bit value in Z3.

Description of Rmat: Rmat16 contains 16 vectors containing 16 vectors of PackedZ3<81>. In other words, Rmat16 contains 16 X (16 X 81) elements.

The following document explains the **variables** used in the code.

**matrix\_pointer** acts as a matrix selector selecting one of the sixteen 16 X 81 matrix

**offset pointer** has a dual purpose, it has been defined as int16\_t purposely

**matrix \_internal\_pointer** traverses through the 16 X 81 matrix column wise. The pointer goes decreasing from high(15) to low(0). The reason for this orientation is to be able to match with bitwise multiplication with the bits of int16\_t starting from back.

**lookup\_table** is a **16 X 65536** matrix containing the possibility of all multiplication of Rmat16 with numbers from 0 to 65535.

**Pseudocode:**

**LOOP** for 16 iteration, traversing each row of lookup table

**RESIZE** each row of lookup table to accommodate 65536 elements

**INITIALIZE** counter that traverses each of 65536 columns of lookup

table

**LOOP** until the pointer reaches end of the row of the lookup table

**INITIALIZE** a pointer that goes through each bit of

packedModZ3 vector

**LOOP** until the packedZ3 pointer reaches 81

**INITIALIZE** sum of the multiplication of elements as

zero

**LOOP** a pointer that traverses each column of the

16 X 81 matrix and decrease with each iteration until

it reaches zero

**MULTIPLY** elements of matrix with the vector bit

by bit

**END LOOP**

**COMPUTE** mod 3 addition of multiplication of matrix

elements and vector elements

**END LOOP**

**COPY** the computed value as an element of the lookup table

**END LOOP**

**END LOOP**

**END LOOP**

For a more detailed and technical view, algorithm for this operation is given below

Algorithm createLookupTable

**Input:** 16 count of 16 X 81 random matrix in Z3

**Output:** 16 X 65536 matrix of Z3 elements.

**BEGIN**

**matrix\_pointer ← 0**

**while matrix\_pointer < 16 step**

**Set each vector\_offset \belonging matrix\_pointer to size(65536)**

**While vector\_offset < 65536 step 1**

**Packed\_z3\_counter ← 0**

**Sum ← 0**

**inner\_matrix\_pointer ← 15**

**While inner\_matrix\_pointer > 0 step -1**

**matrix \_element:= lookup\_table[matrix\_pointer][inner\_matrix\_pointer**

**Input\_bit := (vector\_offset>>inner\_matrix\_pointer) & 1**

**Sum += matrix\_element \* input\_bit**

**End while**

**Sum := sum % 3**

**End for**

**end do**

**Lookup\_table[matrix\_pointer][vector\_offset] = sum**

**END**

**Implemented Code**

Creation of lookup table:

void create\_lookup\_table(std::vector<std::vector<PackedZ3<81> > >& Rmat16,

                        std::vector<std::vector<PackedZ3<81> > >& lookup\_table)*//create 16 count of (16 X 81) matrix*

{

   PackedZ3<81> temp\_result\_vec;

   int packed\_z3\_counter; *//go through the packedMod3 bit by bit*

int matrix\_internal\_pointer;*//once matrix is selected, this pointer traverses through row of each column*

for(int matrix\_pointer = 0; matrix\_pointer < 16; matrix\_pointer++)*//iterates over the matrices(0 to 15)*

{

       lookup\_table[matrix\_pointer].resize(65536); *//This implies each row has 65536 columns.*

for(int offset\_pointer = 0; offset\_pointer < 65536; offset\_pointer++)*//goes over the columns of lookup table*

{

           packed\_z3\_counter = 0; *//reset the counter that goes bit by bit ovre packedmodz3*

temp\_result\_vec.reset(); *//reset the temporary 81 bit vector which will store the result*

while(packed\_z3\_counter < 81)

           {

               int sum = 0; *//initialize*

matrix\_internal\_pointer = 15; *//start with 15th row of a matrix*

while(matrix\_internal\_pointer >= 0)

               {

                   sum += ((Rmat16[matrix\_pointer][matrix\_internal\_pointer].at(packed\_z3\_counter))

                           \* ((offset\_pointer>>matrix\_internal\_pointer) & 1)); *//taking each bit of the 16 bit value(0-65535)*

matrix\_internal\_pointer--; *//go upward toward the first row*

}

               temp\_result\_vec.set(packed\_z3\_counter,(sum%3)); *//set the value computed after the multiplication*

packed\_z3\_counter++;

           }

           lookup\_table[matrix\_pointer][offset\_pointer] = temp\_result\_vec; *//set the packedmod3 value as an entry in lookup table*

}

   }

}

Use Lookup table function:

void uselookup(PackedZ3<81>& result\_sum, std::vector<uint64\_t>& outKX\_input, std::vector<std::vector<PackedZ3<81> > >& lookup\_table)

{

   std::vector<PackedZ3<81> > result\_table(16); *//stores the result of multiplication using lookup table*

for(int word\_count = 0; word\_count < 16; word\_count++)

   {

       uint64\_t value = outKX\_input[word\_count];

       result\_table[word\_count] = lookup\_table[word\_count][value];

   }

*//add up the 16 count of packedmod3 values*

result\_sum = result\_table[0];

   for(int count = 1; count < 16; count++)

   {

       result\_sum += result\_table[count];

   }

}