COMP3601: Design Project A

Project Description

21T3

Approximate Computing for Quantum Robust Cryptography

Learning With Errors (LWE) [1], a cryptographic method, was proposed in 2005 and is a hard problem for quantum computers to break. Since it's a hard problem for quantum computers to crack makes it one of known quantum robust cryptographic methods. You can refer to [1] for an in-depth understanding of LWE. In this design project, we will be focusing on implementing LWE and evaluating its performance when approximate error is introduced.

Public-key Cryptography

Public-key cryptographic systems are deployed for scenarios where an entity wants to receive secret messages from multiple other entities with assurance that the message is only readable by the sender and the intended receiver.

Let's assume Bob wants to receive private messages that should not be disclosed to anybody else except Bob and the sender. If Bob chooses to use a public key cryptographic system, he will create a pair of keys: a private key and a public key. Then, Bob publicises his public key while keeping the private key undisclosed. So, Bob's public key is a readily available piece of information and is not a secret. If Alice wants to send a private message to Bob, she would lookup Bob's public key, encrypt her message with Bob's public key and send the message to Bob. Once, the message is encrypted using Bob's public key it can only be decrypted using Bob's private key. Bob's public key is only useful for encrypting messages for Bob. So, Eve, who might gain access to encrypted messages destined to Bob will not be able to infiltrate any information despite knowing Bob's public key.

Learning With Errors (LWE)

LWE can be used to implement quantum robust public-key cryptography. If Bob chooses to use LWE, first, he needs to create a secret key, s and a list of error values, e. Then, Bob needs to populate a matrix A with some random numbers in the range $(0, q)$ and calculate $B = (A \times s + e)\ mod\ q$. Here, q is a prime number that ideally should be as large as possible. But for demonstration purposes, you can take a relatively smaller prime number as q. Further, for this project, your s should be a vector. Having calculated the above values, Bob can use s as his private key and publicise A, B and q as his public key. An example pair of private-public keys and an error vector are given below.

Private key: $s = \begin{bmatrix} 27 \\ 58 \\ 8 \\ 2 \end{bmatrix}$

Public key: $\left(A = \begin{bmatrix} 30 & 45 & 75 & 43 \\ 62 & 73 & 43 & 24 \\ 64 & 25 & 30 & 11 \\ 0 & 27 & 74 & 78 \\ 48 & 78 & 41 & 25 \\ 29 & 34 & 13 & 38 \\ 19 & 60 & 17 & 28 \\ 52 & 74 & 12 & 24 \end{bmatrix}, B = \begin{bmatrix} 77 \\ 58 \\ 41 \\ 24 \\ 37 \\ 14 \\ 76 \\ 72 \end{bmatrix}, q = 79 \right)$

$$e = \begin{bmatrix} 0 \\ -1 \\ -2 \\ 1 \\ 1 \\ 2 \\ -1 \\ -1 \end{bmatrix}$$

Here,

$$B = (A \times s + e) \, mod \, q = \left(\begin{bmatrix} 30 & 45 & 75 & 43 \\ 62 & 73 & 43 & 24 \\ 64 & 25 & 30 & 11 \\ 0 & 27 & 74 & 78 \\ 48 & 78 & 41 & 25 \\ 29 & 34 & 13 & 38 \\ 19 & 60 & 17 & 28 \\ 52 & 74 & 12 & 24 \end{bmatrix} \times \begin{bmatrix} 27 \\ 58 \\ 8 \\ 2 \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \\ -2 \\ 1 \\ 1 \\ 2 \\ -1 \\ -1 \end{bmatrix} \right) mod \, 79 = \begin{bmatrix} 77 \\ 58 \\ 41 \\ 24 \\ 37 \\ 14 \\ 76 \\ 72 \end{bmatrix}$$

Let's assume Alice wants to send a secret message to Bob and the intended message is just the character "A". Alice first needs to convert the message to a bit string. So, in this case, A is 65 in ASCII which translates to 0100 0001 in binary. So, the $Message$ is $[0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1]$. Now for each bit in the $Message$, Alice must calculate a pair of values (u, v) as follows:

For each bit M in $Message$:

$$u = \left(\sum A_{sample} \right) mod \, q$$

$$v = \left(\sum B_{sample} - \frac{q}{2} M \right) mod \, q$$

Here A_{sample} is a random row chosen from A. Each B_{sample} is the element that has the same index as the chosen A_{sample}. To calculate (u, v), Alice must take the summation of an arbitrary number of A_{sample}s and B_{sample}s. For example, let's assume that Alice chose to have a sample size of 2 and for the first bit where $M = 0$, Alice randomly chose the 8th and the 5th rows of A.

Therefore,

$$u = \left(\sum A_{sample} \right) mod \, q = ([52 \quad 74 \quad 12 \quad 24] + [48 \quad 78 \quad 41 \quad 25]) \, mod \, 79$$

$$= [21 \quad 73 \quad 53 \quad 49]$$

And

$$v = \left(\sum B_{sample} - \frac{q}{2} M \right) mod \, q = \left(72 + 37 - \frac{79}{2} \times 0 \right) mod \, 79 = 30$$

After calculating (u, v) pairs for all the bits in the $Message$, Alice sends the list of (u, v) to Bob. For the above message Alice would be sending something as follows:

$<([21 \quad 73 \quad 53 \quad 49], 30), ([59 \quad 0 \quad 9 \quad 2], 51), ([19 \quad 8 \quad 12 \quad 27], 21),$
$([78 \quad 44 \quad 37 \quad 68], 35), ([30 \quad 72 \quad 70 \quad 42], 22), ([67 \quad 59 \quad 58 \quad 53], 34),$
$([52 \quad 22 \quad 7 \quad 23], 17), \ ([12 \quad 28 \quad 56 \quad 62], 32)>$

Upon receiving the encrypted $Message$, Bob can decrypt the $Message$ as follows:

For each bit (u, v)in the encrypted $Message$:

$$dec = (v - u \cdot s) \ mod \ q$$

If $dec < \frac{q}{2}$

$$M = 0$$

Else

$$M = 1$$

So, for $u = [21 \quad 73 \quad 53 \quad 49]$ and $v = 30$,

$$dec = (v - u \cdot s) \ mod \ q = \left(30 - [21 \quad 73 \quad 53 \quad 49] \cdot \begin{bmatrix} 27 \\ 58 \\ 8 \\ 2 \end{bmatrix} \right) mod \ 79 = (30 - 5323) \ mod \ 79$$

$$= 0$$

Since $0 < \frac{79}{2}$, the decrypted bit value $M = 0$.

By repeating the process for all pairs of (u, v), Bob can bit by bit decrypt M to fully recover the $Message$.

Pair	dec	M
$([21 \quad 73 \quad 53 \quad 49], 30)$	0	0
$([59 \quad 0 \quad 9 \quad 2], 51)$	41	1
$([19 \quad 8 \quad 12 \quad 27], 21)$	0	0
$([78 \quad 44 \quad 37 \quad 68], 35)$	1	0
$([30 \quad 72 \quad 70 \quad 42], 22)$	1	0
$([67 \quad 59 \quad 58 \quad 53], 34)$	0	0
$([52 \quad 22 \quad 7 \quad 23], 17)$	0	0
$([12 \quad 28 \quad 56 \quad 62], 32)$	40	1

Task 1

As the first task of this project, you are expected to demonstrate the functionality of LWE using a MATLAB model followed by a synthesisable HDL model. You should use Xilinx Vivado toolchain to obtain a synthesisable design for Xilinx Kintex-7 XXXXXXXXX. You should take into account the area, throughput and the flexibility (ability to vary parameters such as size of A, s, etc.) of your design. When presenting results (area and throughput), you are expected to base on the following configurations:

	Size of A	Range of q	Range of e
1	256×4	1 to 128	-1 to 1
2	8192×8	2048 to 8192	-4 to 4
3	32768×16	16384 to 65535	-16 to 16

You may choose other interesting configurations in addition to the three configurations above.

Further, you can use the model that you developed to solve the challenge that you will receive later in the course. The challenge will be in the form of a text file that you can feed to your model via the top-level testbench of the HDL model.

Task 2

As the second task of the project, you will be replacing the multiplier used in $B = (A \times s + e) \, mod \, q$ with an approximate multiplier. Approximate multipliers are a class of multipliers that can deliver higher performance and better energy efficiency at the expense of accuracy. Approximate multipliers produce approximately correct results causing the result to inherently include an error. Therefore, with approximate multipliers, you would be able to perform $B = (A \times s) \, mod \, q$ as opposed to the $B = (A \times s + e) \, mod \, q$ operation performed on a precise multiplier.

Similar to Task 1, you will be demonstrating the function of LWE using a MATLAB model and afterward with a synthesisable HDL model. You should use the same configurations detailed in Task 1 for comparison purposes.

References

[1] http://www.cims.nyu.edu/~regev/papers/lwesurvey.pdf