**Toeplitz Matrix**

Toeplitz matrix is of the form.A close up of a logo

Description automatically generated

Which follows the property: **Ai, j = Ai+1, j+1**

* Not necessarily a square matrix, but we are concerned with the square matrix.

**Solving matrix vector multiplication using Toeplitz matrix.**

**Challenges:**

1. Preparing secret key matrix as a Toeplitz matrix. The elements can be stored in one of two ways; either array inside an array or a single array can be used to store them.
2. Finding a relation to quickly perform matrix vector multiplication by packing variable.
3. Unpack the value back to required format.

Solution:

By packing binary digits into decimal, I am trying to find a relation between their multiplication in binary and decimal.

Say security parameter (n) = 4. For creating a Toeplitz matrix, we need 2n-1 elements, i.e. 7 elements.

Say the element is **1011011** and input is x = **1100**.

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

Method 1: Toeplitz matrix created will be

|  |  |
| --- | --- |
| 2 | 3 |
| 1 | 1 |
| 2 | 2 |
| 3 | 1 |

Pack two bits at a time. The result will be

Similarly, the input can be written as 1100 🡺 30

The result can be found by taking the minimum among the input and secret-key

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Key | Input | Output | Binary | Addition of bits/ Number of ones |
| 2,3 | 3,0 | 2,0 | 1000 | 1 |
| 1,1 | 3,0 | 1,0 | 0100 | 1 |
| 2,2 | 3,0 | 1,0 | 0100 | 1 |
| 3,1 | 3,0 | 3,0 | 1100 | 2 |

Explanation:

* This involves dividing entire string of bits into sets of two bits. So say, 1011010011 can be written as 23103. This will be done both for input and key.
* Once both the input and key are in the same 2-bit representational format. We can take the minimum to perform a bitwise AND type operation. **But there is ONE exception.**
* Notice that we get four possible values when we convert bit string into its two-bit representation. 00 for 0, 01 for 1, 10 for 2 and 11 for 3.
* Following is the possible multiplication of all the possible values.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Operands | 2-bit representation | Output of their multiplication | Decimal value | Minimum of two operands |
| 0,0 | 00, 00 | 00 | 0 | 0 |
| 0,1 | 00, 01 | 00 | 0 | 0 |
| 0,2 | 00, 10 | 00 | 0 | 0 |
| 0,3 | 00, 11 | 00 | 0 | 0 |
| 1,1 | 01, 01 | 01 | 1 | 1 |
| 1,2 | 01, 10 | 00 | 0 | 1 |
| 1,3 | 01, 11 | 01 | 1 | 1 |
| 2,2 | 10, 10 | 10 | 2 | 2 |
| 2,3 | 10, 11 | 10 | 2 | 2 |
| 3,3 | 11, 11 | 11 | 3 | 3 |

* The highlighted row is the exception which doesn’t follow this rule. If this could be handled, the multiplication of numbers would be as simple as finding the minimum among two numbers.