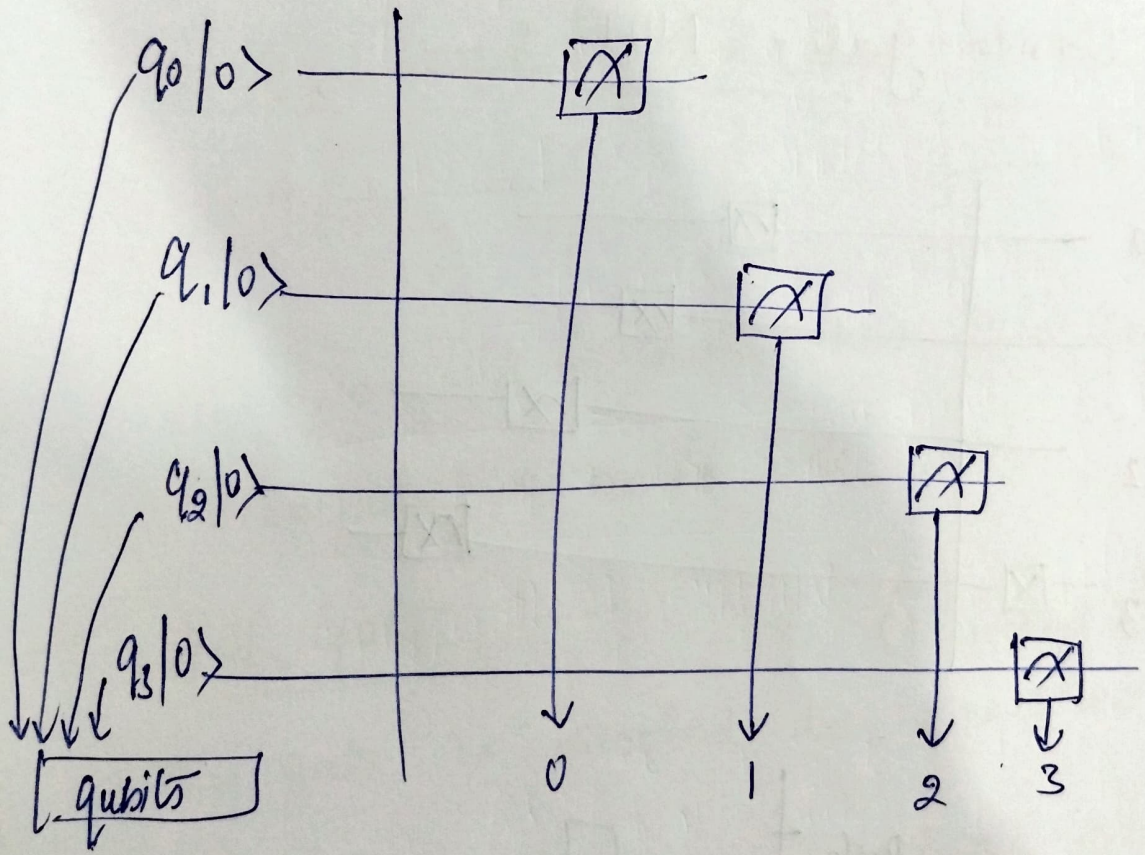
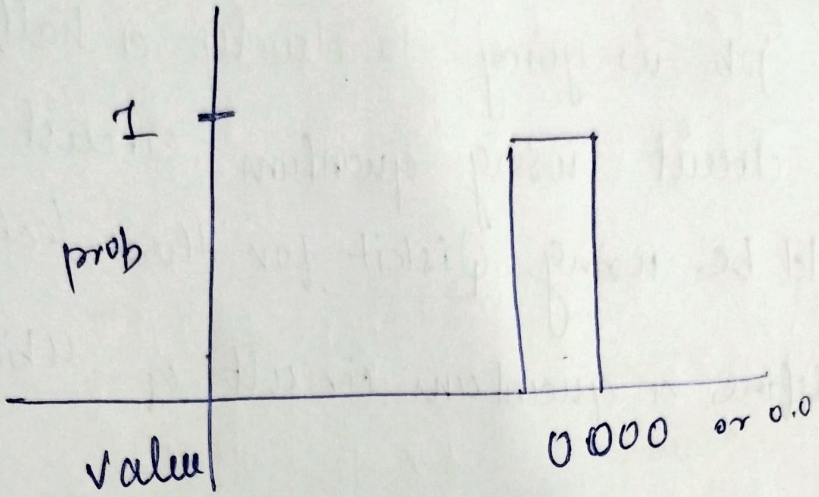
Quantum Computing :—

Our first job is going to develop a half adder circuit using quantum circuit. We would be using Qiskit for these lectures. Let's define a quantum circuit of 4 bits

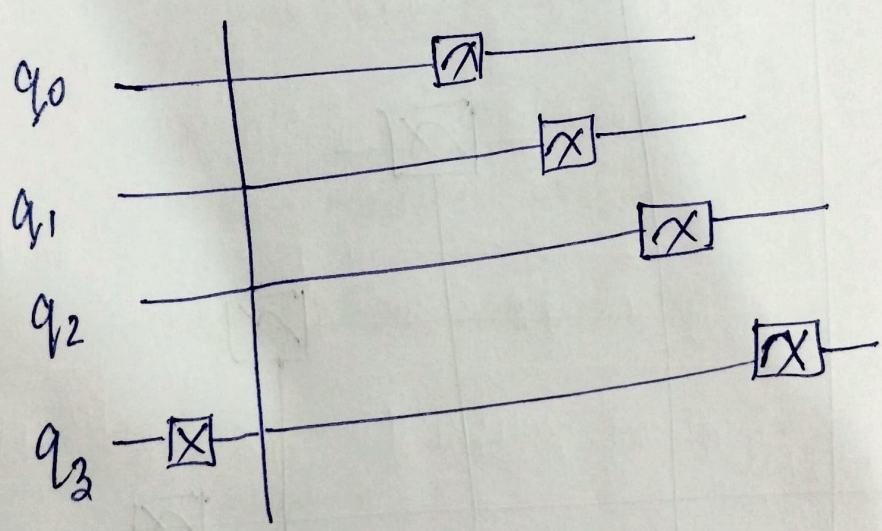


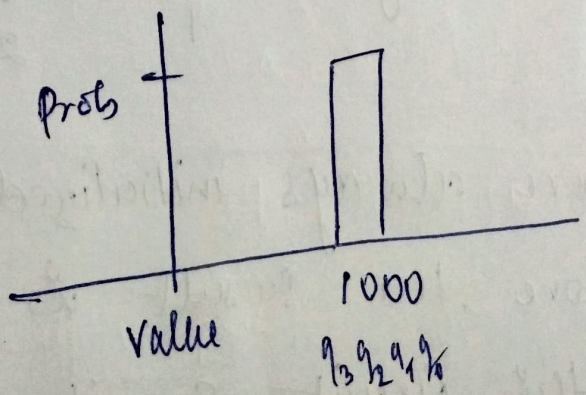
* Qubits are always initialized to zero. Therefore, the result is always zero for this circuit. So, if we draw

draw histogram of this circuit.



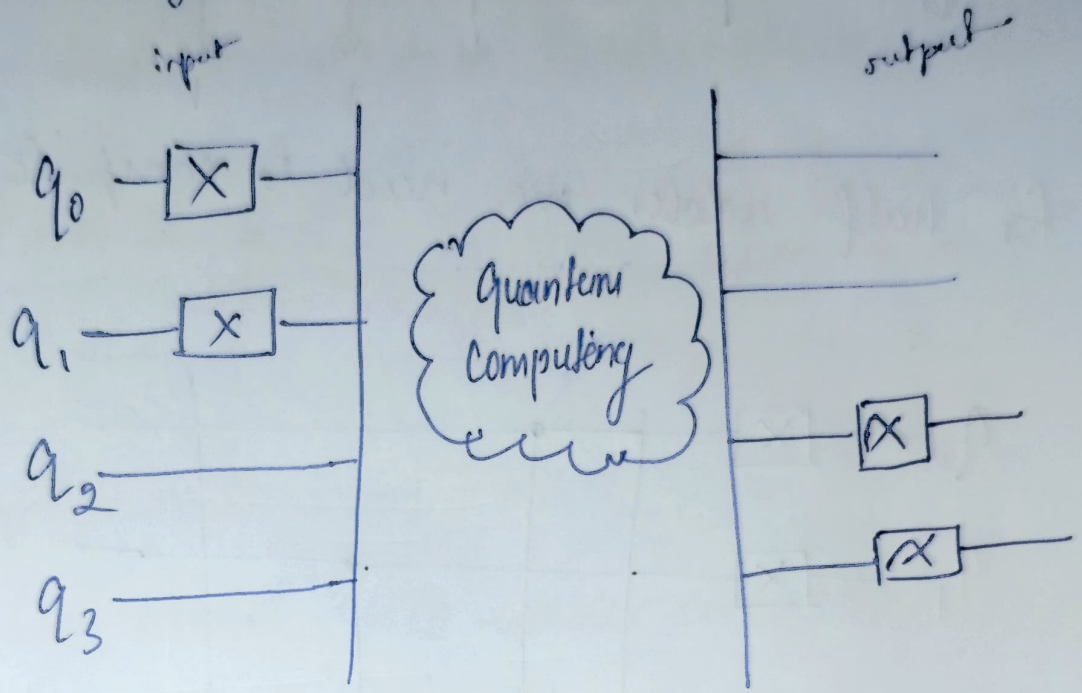
the basic gate - NOT :—





So,

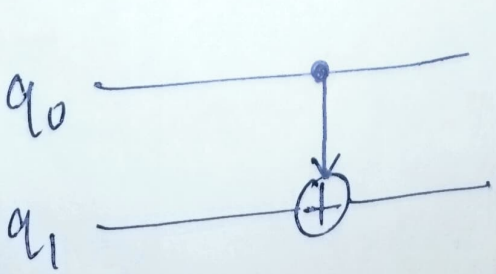
the half-adder circuit we can draw something like following.



we now need to redefine how this quantum computation would look.

XOR Gate :— (Controlled X gate)

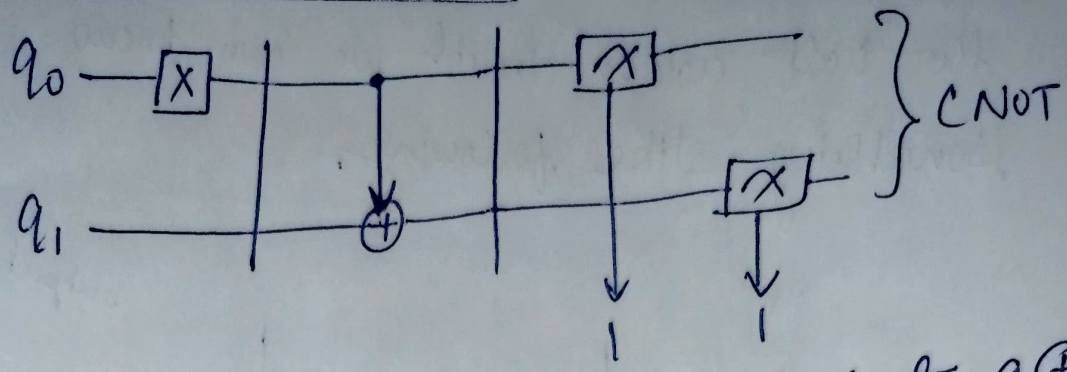
This gate is denoted as CNOT or CX



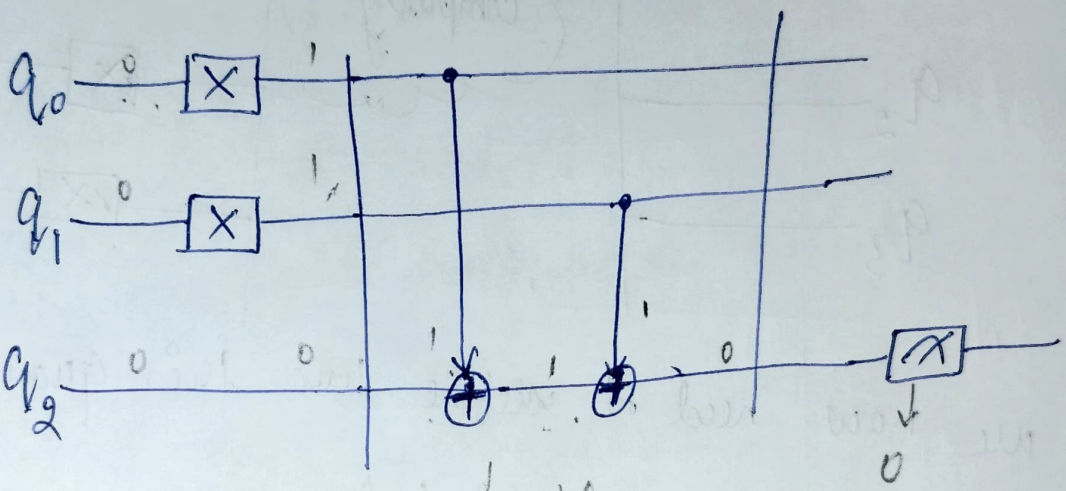
$q_0 = 1, \; q_1 = \bar{q_1}$

$q_0 = 0, \; q_1 = q_1$

if $q_0 = 0$ & $q_1 = 0$



for half adder we need to compute $q_1 \oplus q_0$



if $q_0 = q_1 = q_2 = 0$

then, result = 0

$\Rightarrow q_0 \oplus q_1 = 0$

if $q_0 = 0$, $q_1 = 1$ & $q_2 = 0$

result = 1

$\Rightarrow q_0 \oplus q_1 = 1$

and if $q_0 = 1$, $q_1 = 0$, $q_2 = 0$

result = 1

$\Rightarrow q_0 \oplus q_1 = 1$.

if $q_0 = 1$, $q_1 = 1$ & $q_2 = 0$

result = 0

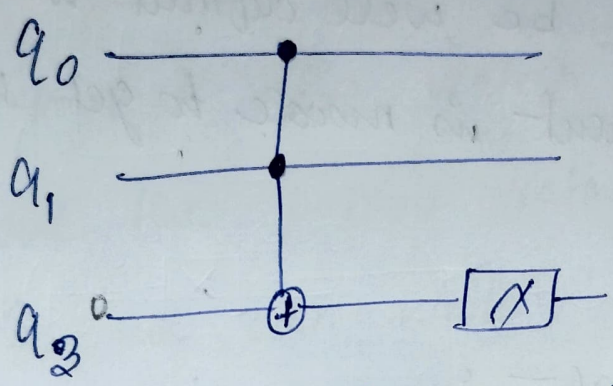
$\Rightarrow q_0 \oplus q_1 = 0$

Now, we need to define AND operation.

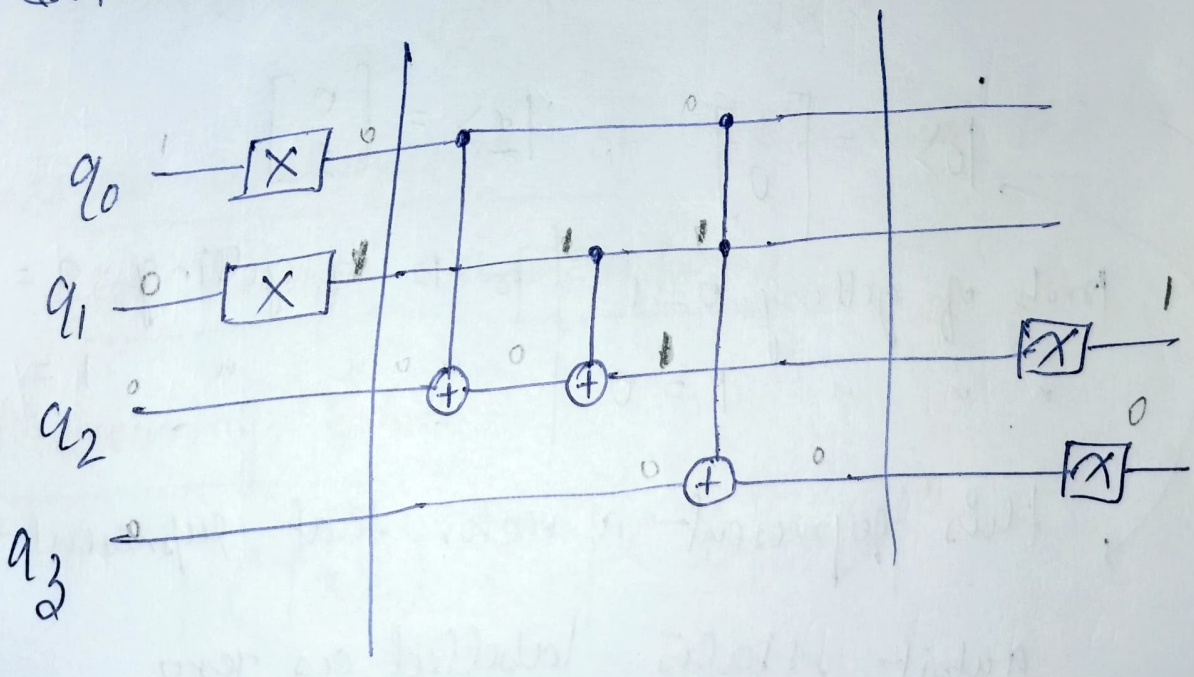
AND Gate :—

* q_0 & q_1 → output is 1, if both q_0 & q_1 are 1.

q_0	q_1	q_3	o/p
0	0	0	0
0	1	0	0
1	0	0	0
1	1	0	1



So, the complete half adder circuit



Difference b/w Qubits and bits :—

* Classical bit is deterministic, it can be either zero or 1.

for qubits, whether we will get 0 or 1, that needs to be well defined when the measurement is made to get the output.

Simple qubit state :—

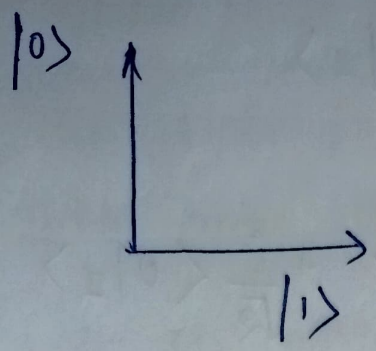
$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad , \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

prob of getting 0 = 1 | prob of getting 0 = 0
" " " 1 = 0 | " " " 1 = 1

This represent a vector that represent qubit state labelled as zero.

But we can also have something like

$$|q_0\rangle \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{i}{\sqrt{2}} \end{bmatrix}$$



→ these are orthonormal bases → ~~orig~~ orthogonal and normalized.

These two vectors are linearly independent. Using both the vectors, we can represent all the possible vectors in 2D space

So, we can write

$$|q_0\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{i}{\sqrt{2}}|1\rangle$$

linear combination of $|0\rangle$ and $|1\rangle$ → also known as superposition.

Measuring Qubits

$$p(|x\rangle) = |\langle x|\psi\rangle|^2$$

$\langle x|$ ↓ row vector

we want to find the probability of measuring a state $|\psi\rangle$ in the state $|x\rangle$.

$$|q_0\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{i}{\sqrt{2}}|1\rangle$$

$$\langle 0|q_0\rangle = \frac{1}{\sqrt{2}}\langle 0|0\rangle + \frac{i}{\sqrt{2}}\langle 0|1\rangle$$

$$= \frac{1}{\sqrt{2}} + 0$$

$$|\langle 0|q_0\rangle|^2 = \frac{1}{2}.$$

$$p(|0\rangle) = |\langle 0|q_0\rangle|^2 = \frac{1}{2}.$$

So, the amplitude is related to the probabilities.

So, for normalization

if $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$

then, $\boxed{\alpha^2 + \beta^2 = 1}$

Global phase :-

The state $|1\rangle$ will give 1 as output with probability 1.

let us consider a state

$$|q\rangle = \begin{bmatrix} 0 \\ i \end{bmatrix} \Rightarrow = i|1\rangle$$

$$|\langle x|(i|1\rangle)|^2 = |i\langle x|1\rangle|^2 = |\langle x|1\rangle|^2$$

Thus the prob. of measuring x in the $i|1\rangle$ is identical to $|1\rangle$. Thus we can conclude that these two states are equivalent.

Observer effect :-

$$|q\rangle = \alpha|0\rangle + \beta|1\rangle \xrightarrow{\text{measure}} 0$$

$$|q\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \xrightarrow{\text{measure } |0\rangle} |q\rangle = |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Collapsing.

* If we constantly measure the state of the qubits to keep track of their value at each point in a computation, it will be a well defined state of either $|0\rangle$ or $|1\rangle$. So, no diff. b/w classical & quantum computing.

The Block sphere :-

$$|q\rangle = \alpha|0\rangle + \beta|1\rangle, \quad \alpha, \beta \in \mathbb{C}$$

(complex no.)

We can also represent $|q\rangle$ with the difference in phase b/w states $|0\rangle$ and $|1\rangle$.

(real no.)

$$|q\rangle = \alpha|0\rangle + e^{i\phi}\beta|1\rangle, \quad \alpha, \beta, \phi \in \mathbb{R}$$

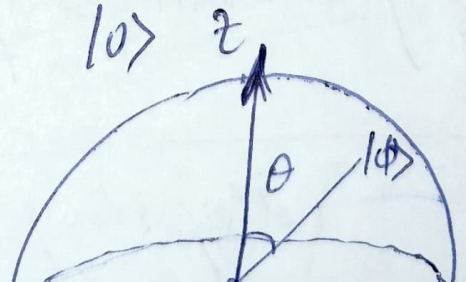
we need to ensure that

$$\sqrt{\alpha^2+\beta^2} = 1$$

if we take $\alpha = \cos\frac{\theta}{2}$ and $\sin\frac{\theta}{2}$.

we can write

$$|q\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle$$



Gates :- NOT Gate

X-Gate → Pauli - x - matrix :-

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = |0\rangle\langle 1| + |1\rangle\langle 0|$$

multiply qubit state with the matrix X.

$$X|0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle$$

$$X|1\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle$$

Thus the x-gate act as an inverter for qubit $|0\rangle$ and $|1\rangle$.

What will be the bloch sphere representation?

Y & Z gate :-

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$Y|0\rangle = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ i \end{bmatrix} \stackrel{?}{=} |1\rangle$$

$$Y|1\rangle = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -i \\ 0 \end{bmatrix} \stackrel{?}{=} |0\rangle$$

$$Z|0\rangle = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle$$

$$Z|1\rangle = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle.$$

Why Z-gate has no effect on state $|0\rangle$ and $|1\rangle$. because value of $|0\rangle$ & $|1\rangle$ are the eigen vector of (Z).

We know the concept of eigen vectors.

$S\Omega|v\rangle = \lambda|v\rangle \rightarrow v$ is the eigen vector of $S\Omega$. The Z-gate has no effect on state $|0\rangle$ & $|1\rangle$ as they are the eigen vector of Z matrix.

The computational basis formed by $|0\rangle$ & $|1\rangle$ are called Z bases.

We can have infinite no. of possible bases.

$$|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

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

Consider X gate

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = -1 \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}$$

Thus, $|+\rangle$ & $|-\rangle$ construct the X-basis.

Hadamard Gate

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$H|0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = |+\rangle$$

$$H|1\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = |-\rangle$$

So, it can convert z-base into X-base.

$$H|+\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$= \frac{1}{2} \begin{bmatrix} 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle$$

$$H|-\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{bmatrix}$$

$$= \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$= \frac{1}{2}\begin{bmatrix} 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle .$$

Now,

We can also use Hadamard gate to measure in the X-basis (from Z-basis).

$$P(|+\rangle) = |\langle +|q\rangle|^2$$

$$P(|-\rangle) = |\langle -|q\rangle|^2$$

Construction of X-gate for H-gate :-

$$HZH = \overbrace{\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}}^{H}\overbrace{\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}}^{Z}\overbrace{\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}}^{H}$$

$$= \frac{1}{2}\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$= \frac{1}{2}\begin{bmatrix} 0 & 2 \\ 2 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = X$$

Thus a deterministic outcome in the z-basis may have a random result in the x-basis & vice-versa.

P-Gate :- — universally rotation gate

$$P(\phi) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix}$$, z gate is a special gate with $\phi = \pi$.

I-Gate : Identity gate

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad I = XX$$

if we apply two inversion gates together then we get same output as input.

S-Gate :- ($\sqrt{z}$ gate)

$$S = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/2} \end{bmatrix}, \quad S^* = \begin{bmatrix} 1 & 0 \\ 0 & e^{-i\pi/2} \end{bmatrix}$$

→ P gate with $\phi = \pi/2$, (this gate is not its own inverse).

$$\boxed{SS|q\rangle = Z|q\rangle}$$

T-Gate :-

⇒ P gate with $\phi = \frac{\pi}{4}$, known as $\sqrt[4]{Z}$ gate

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}, \quad T^\dagger = \begin{bmatrix} 1 & 0 \\ 0 & e^{-i\pi/4} \end{bmatrix}.$$

U-Gate :- universal gate → we can construct all the gates.

~~⇒ P gate with ϕ~~

$$U(\theta, \phi, \lambda) = \begin{bmatrix} \cos\frac{\theta}{2} & -e^{i\lambda}\sin\frac{\theta}{2} \\ e^{i\phi}\sin\frac{\theta}{2} & e^{i(\phi+\lambda)}\cos\frac{\theta}{2} \end{bmatrix}$$

$$U\left(\frac{\pi}{2}, 0, \pi\right) = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = H$$

$$U(0, 0, \lambda) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{bmatrix} = P$$

Multi Qubit States :—

$$|a\rangle = a_{00}|00\rangle + a_{01}|01\rangle + a_{10}|10\rangle + a_{11}|11\rangle \quad \text{—— (1)}$$

A single qubit has two possible states and a qubit state has two complex magnitudes.

Thus, for two bit eqn (1) holds.

$$|a\rangle = \begin{bmatrix} a_{00} \\ a_{01} \\ a_{10} \\ a_{11} \end{bmatrix}, \quad P(|00\rangle) = |a_{00}|^2$$

Normalization :—

$$|a_{00}|^2 + |a_{01}|^2 + |a_{10}|^2 + |a_{11}|^2 = 1$$

If we have separated qubits, we can describe their collective state using Knonecker product.

$$|a\rangle = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix}, \quad |b\rangle = \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}.$$

Multi bit quantum state :— Week 3
01/11/2024

$$q_0 = \alpha|0\rangle + \beta|1\rangle$$

two bit quantum state

$$|a\rangle = a_{00}|00\rangle + a_{01}|01\rangle + a_{10}|10\rangle + a_{11}|11\rangle$$

$$|a\rangle = \begin{bmatrix} a_{00} \\ a_{01} \\ a_{10} \\ a_{11} \end{bmatrix} \Rightarrow |a_{00}|^2 + |a_{01}|^2 + |a_{10}|^2 + |a_{11}|^2 = 1$$

& let two single qubit systems. → two qubit system

$$|a\rangle = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix}, \quad |b\rangle = \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}$$

$$|b_a\rangle = |b\rangle \otimes |a\rangle$$

↓

Kronecker's product

$$|ba\rangle = \begin{bmatrix} b_0 \times \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} \\ b_1 \times \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} b_0 a_0 \\ b_0 a_1 \\ b_1 a_0 \\ b_1 a_1 \end{bmatrix}$$

(eg) In exam ;

$$a_0 = \begin{bmatrix} \frac{1}{\sqrt{3}} \\ \frac{\sqrt{2}}{\sqrt{3}} \end{bmatrix} \qquad b_0 = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$$

$$|b_0 a_0\rangle = ?$$

$a_{00} = \frac{1}{\sqrt{3}}$, $a_{01} = \frac{\sqrt{2}}{3}$

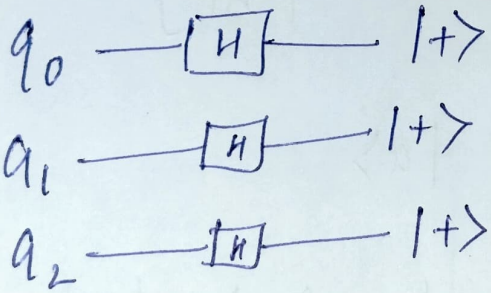
$b_{00} = \frac{1}{\sqrt{2}}$, $b_{01} = \frac{1}{\sqrt{2}}$

$$|b_0 a_0\rangle = \begin{bmatrix} b_{00}a_{00} \\ b_{00}a_{01} \\ b_{01}a_{00} \\ b_{01}a_{01} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{\sqrt{2}} \times \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{2}} \times \frac{\sqrt{2}}{3} \\ \frac{1}{\sqrt{2}} \times \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{2}} \times \frac{\sqrt{2}}{3} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}\sqrt{3}} \\ \frac{1}{3} \\ \frac{1}{\sqrt{2}\sqrt{3}} \\ \frac{1}{3} \end{bmatrix}$$

Suppose we have 3-bit qubit system

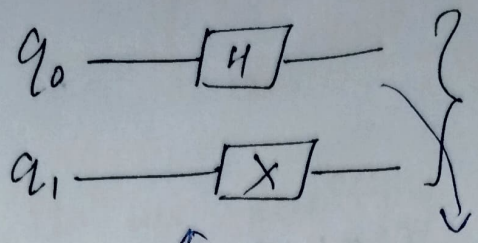
$$|+\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

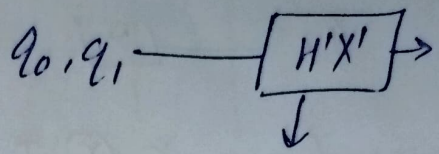


$$|+++\rangle = \frac{1}{\sqrt{8}} \begin{bmatrix} \\ \\ \\ \\ \\ \\ \\ \end{bmatrix}$$

Gates in multi bit qubit system :-

(1)





$X|q_1\rangle \otimes H|q_0\rangle$

↑ Kronecker product

$= (X \otimes H)\,|\,q_1 q_0\rangle$

↓ tensor product

$\Rightarrow$ tensor product

$X \otimes H$

$$= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \otimes \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$= \frac{1}{\sqrt{2}} \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \\ 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & H \\ H & 0 \end{bmatrix}$$

(2) q_0 —— [I] ——

q_1 —— [X] ——

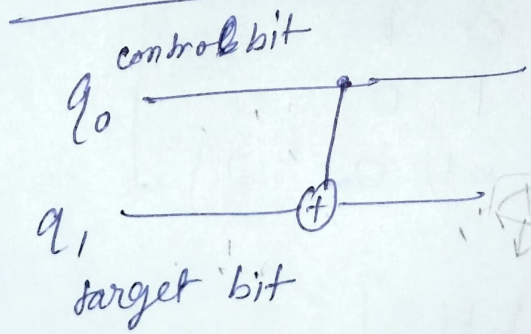
Combined structure: $X \otimes I$

$$= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & X \\ X & 0 \end{bmatrix}$$

(3) CNOT gate as CX gate :—



I/P (t, c)

O/P (t, c)

I/P (t, c)

0 0

0 1

1 0

1 1

O/P (t, c) → control bit remains same.

0 0

1 1

1 0

0 1

these will change same as XOR.

matrix representation → (4×4) (CNOT gate) → 2 bit qubit system

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \text{ or } \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Now, if we multiply CNOT gate with our qubit system, suppose,

$$|a\rangle = \begin{bmatrix} a_{00} \\ a_{01} \\ a_{10} \\ a_{11} \end{bmatrix}$$

$$\xrightarrow[\times CNOT]{} |a\rangle = \begin{bmatrix} a_{00} \\ a_{11} \\ a_{10} \\ a_{01} \end{bmatrix}$$

swapped values

example

find $CNOT|a\rangle$

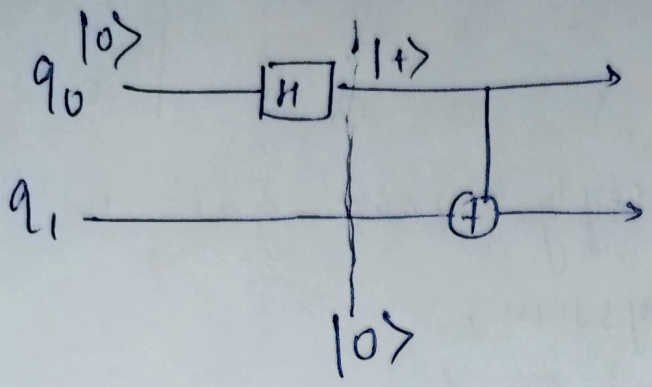
$a = |q_1 q_0\rangle$ & $|q_1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ &

$|q_0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

$$a = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\therefore CNOT|a\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

(4) $|+\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$



$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \\ 0 \end{bmatrix}$$

swap

after applying CNOT gate

$$|0+\rangle = \frac{1}{\sqrt{2}} \left[|00\rangle + |01\rangle \right]$$

$$= \frac{1}{\sqrt{2}} \left[\frac{1}{\sqrt{2}} \quad \frac{0}{\sqrt{2}} \quad 0 \quad \frac{0}{\sqrt{2}} \right]$$

$$\text{CNOT} |0+\rangle = \frac{1}{\sqrt{2}} \left(|00\rangle + |11\rangle \right)$$

$$= \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ 1/\sqrt{2} \end{bmatrix} \rightarrow \boxed{\text{entanglement or Bell state}}$$

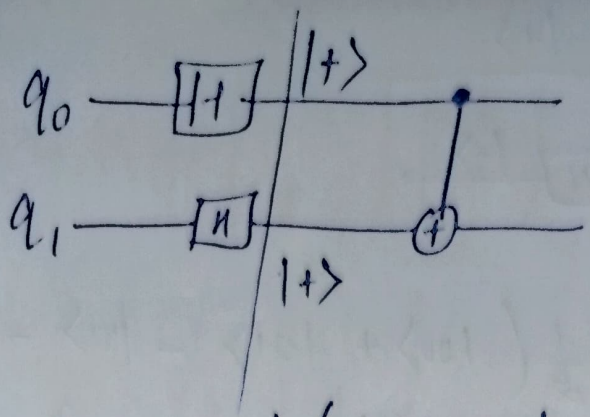
Hence, prob. of measuring state 00 = 50%.

" " " " 11 = 50%.

" " " " 01 = 0%.

" " " " 10 = 0%.

(5)



$$|++\rangle = \frac{1}{2}\left(|00\rangle + |01\rangle + |10\rangle + |11\rangle\right)$$

$$|++\rangle = |+\rangle \otimes |+\rangle$$

$$= \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix} \otimes \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}$$

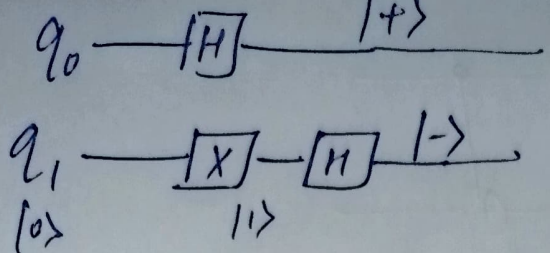
$$= \begin{pmatrix}1/\sqrt{2}\\1/\sqrt{2}\end{pmatrix} \otimes \begin{pmatrix}1/\sqrt{2}\\1/\sqrt{2}\end{pmatrix}$$

$$= \begin{bmatrix}\frac{1}{2}\\ \frac{1}{2}\\ \frac{1}{2}\\ \frac{1}{2}\end{bmatrix} \begin{matrix}\longrightarrow |00\rangle\\ \longrightarrow |01\rangle\\ \longrightarrow |10\rangle\\ \longrightarrow |11\rangle\end{matrix}$$

$$\text{CNOT}|++\rangle = \begin{bmatrix}\frac{1}{2}\\ \frac{1}{2}\\ \frac{1}{2}\\ \frac{1}{2}\end{bmatrix} = \frac{1}{2}\left(|00\rangle + |11\rangle + |10\rangle + |01\rangle\right)$$

swapped

(5)



$$|-+\rangle = \frac{1}{2}\left(|00\rangle + |01\rangle - |10\rangle - |11\rangle\right)$$

$$= \left[\frac{1}{2}, \frac{1}{2}, -\frac{1}{2}, \frac{-1}{2}\right].$$

$$\text{CNOT}|-+\rangle = \left[\frac{1}{2}, \frac{-1}{2}, \frac{-1}{2}, \frac{1}{2}\right].$$

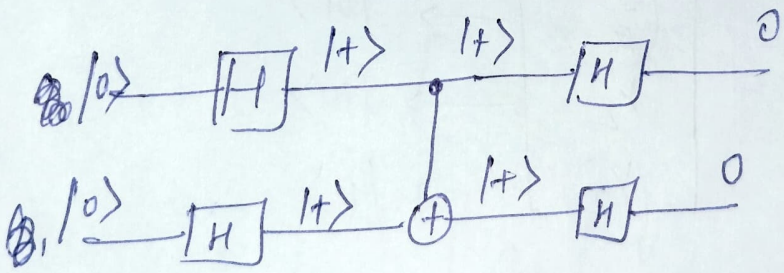
target bit, control bit

$$= |--\rangle$$

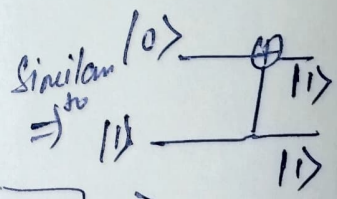
Note :- target bit is not changing here but control bit changes

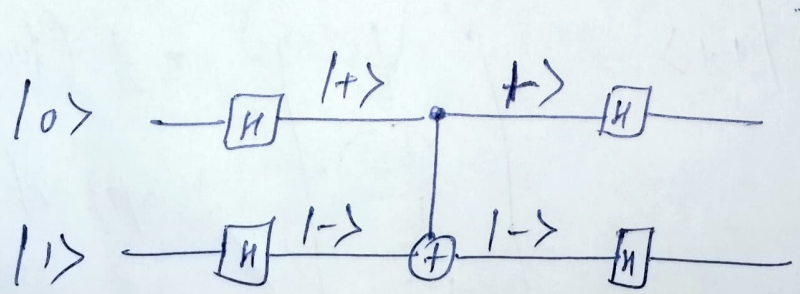
This is known as **phase kickback**.

(6)



Similar to





using two H-gates.

Here control bit changes, control becomes target & target becomes control bit of

$$X|-\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}\begin{bmatrix}1\\ -1\end{bmatrix}$$

$$= \frac{1}{\sqrt{2}}\begin{bmatrix}-1\\ 1\end{bmatrix} = -\frac{1}{\sqrt{2}}\begin{bmatrix}1\\ -1\end{bmatrix}$$

$$= -|-\rangle$$

$$\boxed{\begin{aligned} CNOT|-0\rangle &= |-0\rangle \\ CNOT|-1\rangle &= -|-1\rangle \end{aligned}}$$

$$|-0\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\ -1\end{bmatrix} \otimes \begin{bmatrix}1\\ 0\end{bmatrix}$$

$$= \begin{bmatrix} 1/\sqrt{2} & 0 & -1/\sqrt{2} & 0 \end{bmatrix}$$

$$CNOT|-0\rangle = \begin{bmatrix} 1/\sqrt{2} & 0 & -1/\sqrt{2} & 0 \end{bmatrix}$$

$$|-1\rangle = \begin{bmatrix}1/\sqrt{2}\\ -1/\sqrt{2}\end{bmatrix} \otimes \begin{bmatrix}0\\ 1\end{bmatrix}$$

$$|-1\rangle = \begin{pmatrix} 0 & 1/\sqrt{2} & 0 & -1/\sqrt{2} \end{pmatrix}$$

$$CNOT|-1\rangle = \begin{bmatrix} 0 & -\frac{1}{\sqrt{2}} & 0 & 1/\sqrt{2} \end{bmatrix}$$

$$= -|-1\rangle$$

$$\text{CNOT}\,|-+\rangle = \frac{1}{\sqrt{2}}(\text{CNOT}|-0\rangle) + \frac{1}{\sqrt{2}}(\text{CNOT}|-1\rangle)$$

$$= \frac{1}{\sqrt{2}}\begin{bmatrix}\frac{1}{\sqrt{2}} & 0 & \frac{-1}{\sqrt{2}} & 0\end{bmatrix} + \frac{1}{\sqrt{2}}\begin{bmatrix}0 & \frac{-1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}}\end{bmatrix}$$

$$= \begin{bmatrix}\frac{1}{2} & 0 & \frac{-1}{2} & 0\end{bmatrix} + \begin{bmatrix}0 & \frac{-1}{2} & 0 & \frac{1}{2}\end{bmatrix}$$

$$= \begin{bmatrix}\frac{1}{2} & -\frac{1}{2} & \frac{-1}{2} & \frac{1}{2}\end{bmatrix}$$

$$= |--\rangle$$

Controlled T-gate

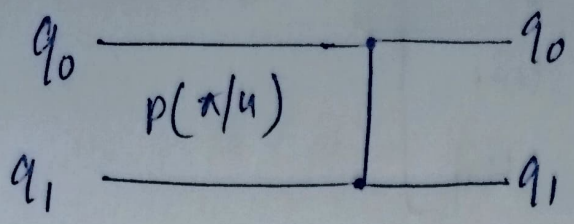
$$T = \begin{bmatrix}1 & 0\\ 0 & e^{i\pi/4}\end{bmatrix}$$

$$|1\rangle = \begin{bmatrix}0\\ 1\end{bmatrix}$$

$$T|1\rangle = e^{i\pi/4}|1\rangle$$

↑ global phase

Consider shift of $\pi/4$



$|1+\rangle$ (target, control bit) $= |1\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$

$= \frac{1}{\sqrt{2}}(|10\rangle + |11\rangle)$

$= \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$

$= \begin{bmatrix} 0 & 0 & 1/\sqrt{2} & 1/\sqrt{2} \\ \downarrow & \downarrow & \downarrow & \downarrow \\ |00\rangle & |01\rangle & |10\rangle & |11\rangle \end{bmatrix}$

$= \frac{1}{\sqrt{2}}(|10\rangle + |11\rangle)$

$\text{Cont.}T|1+\rangle = \frac{1}{\sqrt{2}}(|10\rangle + e^{i\pi/4}|11\rangle)$

$= \underbrace{|1\rangle}_{\text{target bit}} \otimes \underbrace{\frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle)}_{\text{Control bit}}$

→ This is also a phase kickback.

U gate → controlled U gate :—

$$U \text{ gate} = \begin{bmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{bmatrix}$$

$$\text{controlled U-gate} = \begin{bmatrix} I & 0 \\ 0 & U \end{bmatrix}$$

which means if all the control is zero then my o/p is same as the i/p.

And if the control is 1 then o/p is equal to the U-gate applied on i/p.

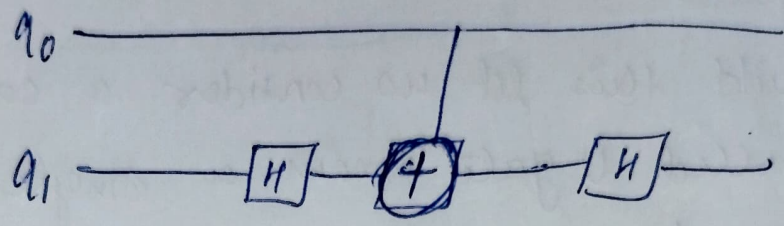
$$\text{Controlled U gate} = \begin{bmatrix} I & 0 \\ 0 & U \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & u_{00} & u_{01} \\ 0 & 0 & u_{10} & u_{11} \end{bmatrix}$$

rearranging

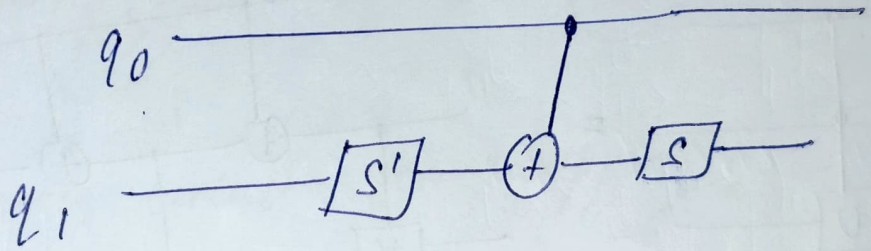
$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & u_{00} & 0 & u_{01} \\ 0 & 0 & 1 & 0 \\ 0 & u_{10} & 0 & u_{11} \end{bmatrix}$$

Controlled Z-gate :-

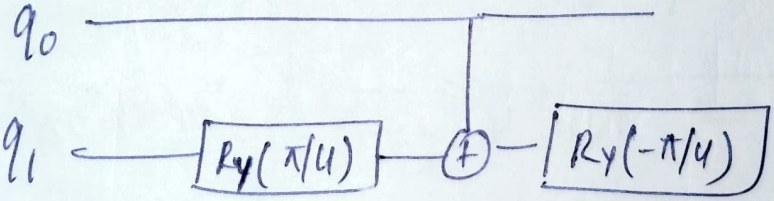
$$HXH = Z$$



Controlled Y gate :-

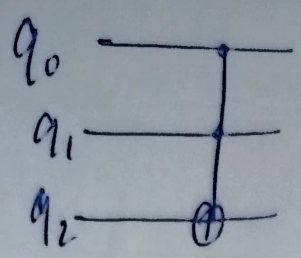


Controlled H gate :-



$$R_y(\theta) = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}.$$

Toffoli Gate (after swapping)

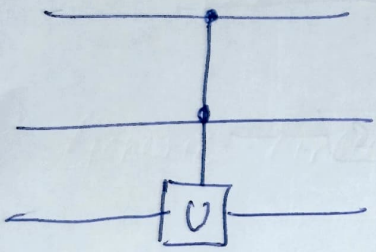


It performs an X on the target only if both controls are set to $|1\rangle$.

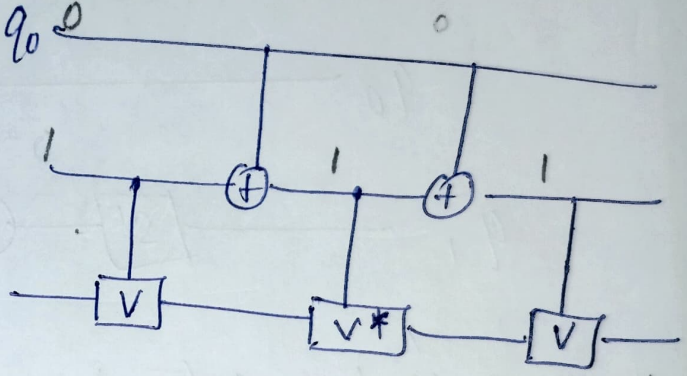
Controlled-controlled U gate

To build this let us consider a controlled controlled U gate from a single qubit rotation U.

We define two controlled version $V = \sqrt{U}$ & $V^* =$ (inverse of V).



=

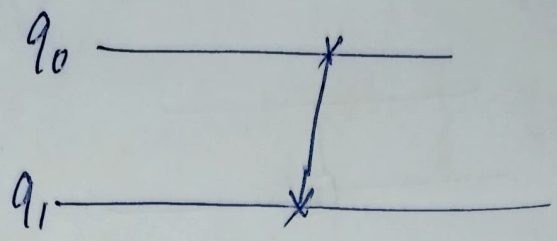


I/P 0, 1 $V \cdot V^* = I$ → this takes away the effect of gate, only one V gate will applied.

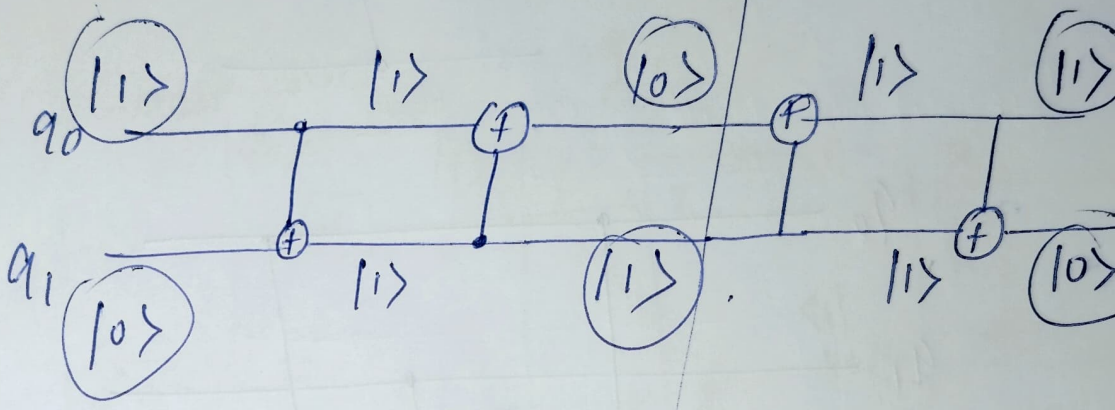
I/P 1, 1 $V^2 = U$

I/P 0, 0 → no V gate operation is applied.

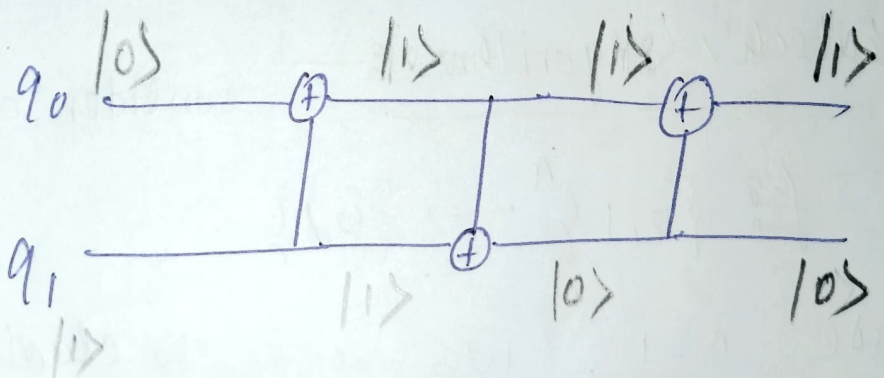
Swapping



$q_0 \rightarrow |1\rangle$, $q_1 \rightarrow |0\rangle$



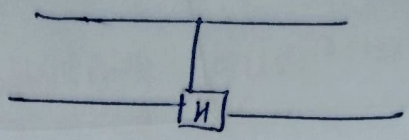
Generic circuit for swapping :—



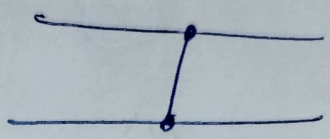
this will work for all the values.

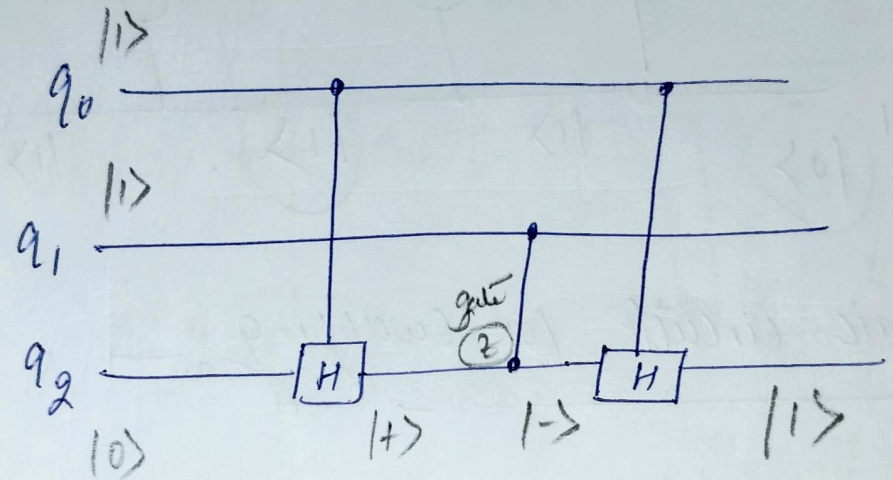
Controlled controlled AND gate :- CCX or Toffoli Gate

Controlled H →



Controlled Z →





Deutsch's Algorithm :- consider a funct^n

$$f: \{0,1\}^n \longrightarrow \{0,1\}$$

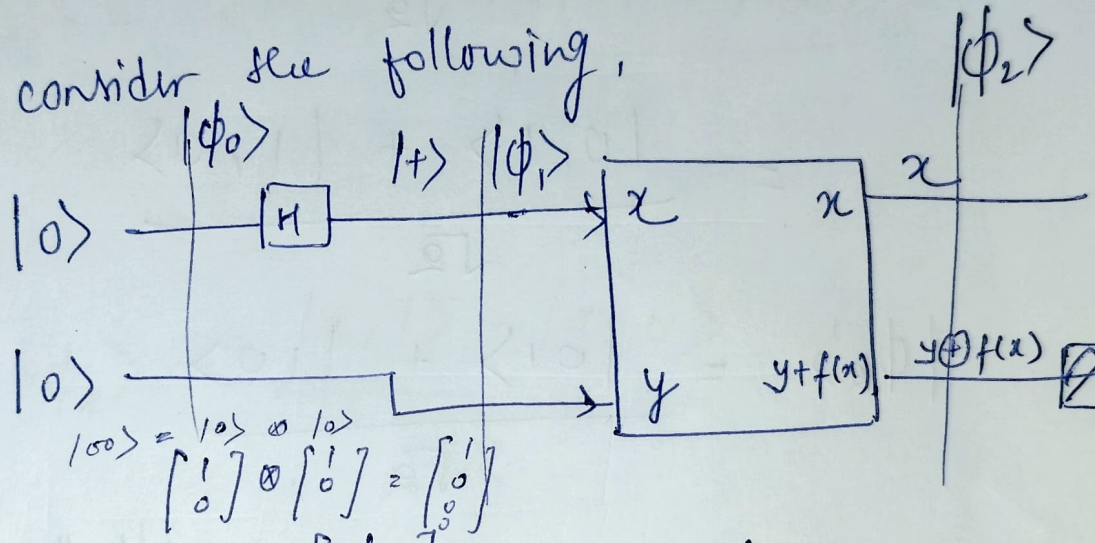
where $n=1$. We need to check if $f(0) = f(1)$. It is equivalent of checking $f(0) \oplus f(1) = 0$ or 1.

0 ↑ equal — 1 ↑ not equal.

for simplicity lets assume that $f(0)=1$ and $f(1)=0$. Then we can write corresponding quantum circuit U_f as

$$U_f = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

lets consider the following,



$|00\rangle = |0\rangle \otimes |0\rangle$

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$|\phi_0\rangle = |00\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad |\phi_1\rangle = \left| \frac{|0\rangle + |1\rangle}{\sqrt{2}}, 0 \right\rangle$$

$$= \frac{|00\rangle + |10\rangle}{\sqrt{2}}$$

$$= |+0\rangle$$

$$|+0\rangle = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1/\sqrt{2} & 0 & 1/\sqrt{2} & 0 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ |00\rangle & |01\rangle & |10\rangle & |11\rangle \end{bmatrix}$$

$$|\phi_2\rangle = \frac{|0, 0+f(0)\rangle + |1, 0 \oplus f(1)\rangle}{\sqrt{2}}$$

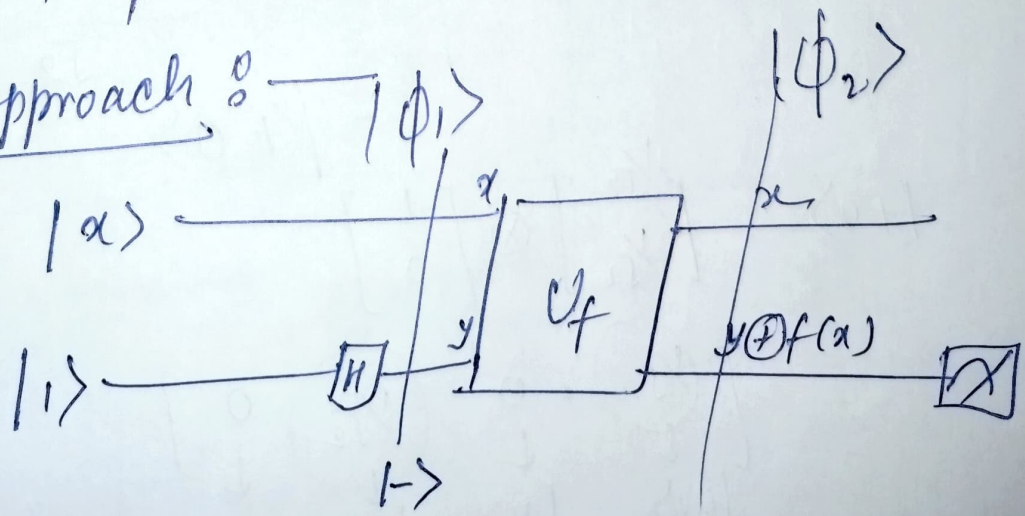
$$= \frac{|0, 0 \oplus 1\rangle + |1, 0 \oplus 0\rangle}{\sqrt{2}}$$

$$= \frac{|0,1\rangle + |1,0\rangle}{\sqrt{2}}$$

$$|\phi_2\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}}.$$

So, if we measure the top/bottom output we can obtain with both the states with 50% prob.

Next Approach :



$$|\phi_1\rangle = \frac{|x,0\rangle - |x,1\rangle}{\sqrt{2}}$$

$$|\phi_2\rangle = \frac{|x, 0 \oplus f(x)\rangle - |x, 1 \oplus f(x)\rangle}{\sqrt{2}}$$

(if)

$f(x) = 0$

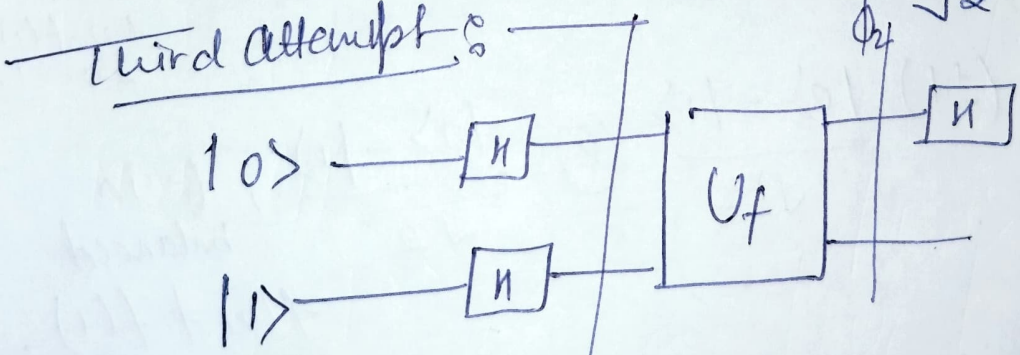
$$|\phi_2\rangle = \frac{|x,0\rangle - |x,1\rangle}{\sqrt{2}}$$

$f(x) = 1$

$$|\phi_2\rangle = \frac{|x,1\rangle - |x,0\rangle}{\sqrt{2}}$$

$$|\phi_2\rangle = (-1)^{f(x)} \cdot \frac{|x,0\rangle - |x,1\rangle}{\sqrt{2}} = (-1)^{f(x)}|x\rangle \otimes$$

Third attempt:



$\phi_0 = |01\rangle$

$\phi_1 = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$ ⊗ $\frac{|0\rangle - |1\rangle}{\sqrt{2}}$

ϕ_2 $\sqrt{2}$

$$|\phi_2\rangle = \frac{(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

in case of

$f(0) = 1$ & $f(1) = 0$

$$|\phi_2\rangle = (-1)\frac{|0\rangle - |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

$f(0) = 0$, $f(1) = 0$

$f(0) = 1$, $f(1) = 1$

$f(0) = 0$ $f(1) = 1$

$$|\phi_2\rangle = \begin{cases} (\pm 1)\frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}, & f \text{ is const } f(0)=f(1)=0 \text{ or } 1. \\ (\pm 1)\frac{|0\rangle - |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}, & f \text{ is balanced } f(0) \neq f(1). \end{cases}$$

$$H|\phi_2\rangle = \begin{cases} (\pm 1)\cdot|0\rangle \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}, & f \text{ is const} \\ (\pm 1)\cdot|1\rangle \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}, & f \text{ is balanced.} \end{cases}$$

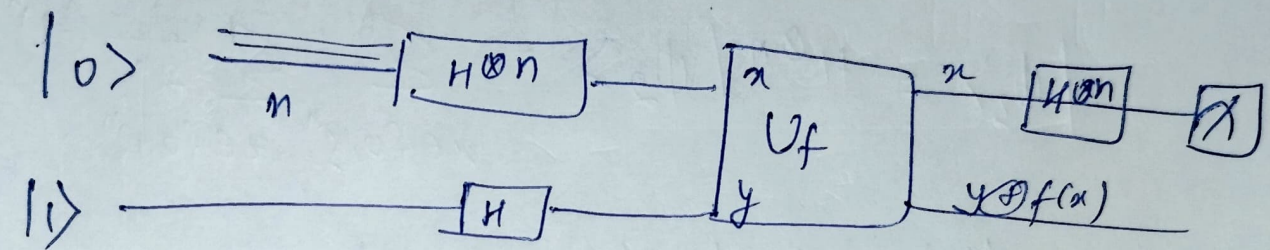
Generalized form :- Deutsch-Jozsa Algo :—

for $f: \{0,1\}^n \rightarrow \{0,1\} \rightarrow n \neq 1$

⇒ f is constant $f(x_i) = f(x_j) \cdots = f(x_n) = 0$ or 1

⇒ f is balanced $n/2 \rightarrow 0$

$n/2 \rightarrow 1$.



$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \quad H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

$$\boxed{H|a\rangle = \frac{1}{\sqrt{2}} \sum_{x \in \{0,1\}} (-1)^{ax} |x\rangle}$$

$$H^{\otimes 2}|00\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$$

$$H^{\otimes 2}|a\rangle = \frac{1}{2}\left(\sum_{x \in \{0,1\}^2} (-1)^{ax} \cdot |x\rangle\right)$$

$$H^{\otimes n}|a\rangle = \frac{1}{(\sqrt{2})^n}\left(\sum_{x \in \{0,1\}^n} (-1)^{ax} \cdot |x\rangle\right)$$

$$|\phi_2\rangle = \sum_{x \in \{0,1\}^n} \frac{1}{\sqrt{2^{n+1}}} (-1)^{f(x)} \cdot |x\rangle (|0\rangle - |1\rangle)$$

$$H^{\otimes n}|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{z \in \{0,1\}^n} (-1)^{xz} \cdot |z\rangle$$

$$|\phi_3\rangle = H^{\otimes n}|\phi_2\rangle = \sum_{x \in \{0,1\}^n} \sum_{z \in \{0,1\}^n} \frac{1}{2^n} (-1)^{xz+f(x)} \cdot |z\rangle \cdot \frac{(|0\rangle - |1\rangle)}{\sqrt{2}}$$

When $f(x)$ is constant then

$$\sum_{x \in \{0,1\}^n} \frac{(-1)^{f(x)}}{2^n} = 1$$

Coeff. of $|0\rangle^{\otimes n}$ will be ↑

so, we will measure $|0\rangle$

$f(x) \Rightarrow$ is balanced

$$\sum_{x \in \{0,1\}^n} \frac{(-1)^{f(x)}}{2^n} = 0$$

coeff of $|0\rangle^{\otimes n}$ is

we will never measure $|0\rangle^{\otimes n}$.

Week 5 — 08/Nov/2024

Bernstein Vazirani Algorithm :—

$f(\{x_0, x_1, \ldots x_n\}) \rightarrow 0$ or 1

given an input — (x) (vector of length 'n') $f(x)$ produces $s \cdot x \bmod 2$.

We need to find (s) → 'n' bit secret scalar

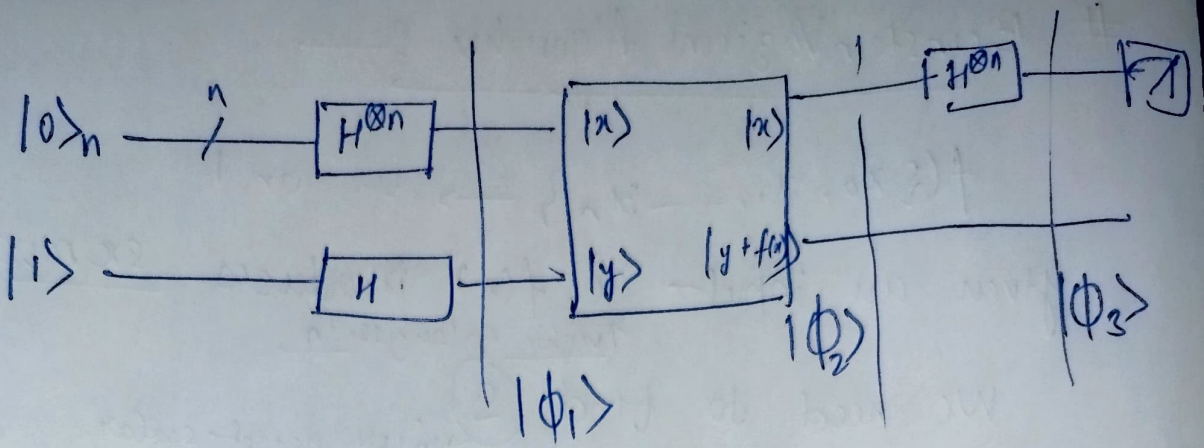
In this classical scenario, given an input — x, the hidden bit string s can be revealed by quering the $f(x)$ by following inputs.

100…0, 010…0, 001…0, ……
(x)

here, we are setting only one of the bit as in this (x) vector and we are checking whether the output of this vector multiplied with 's' is equal to zero or not.

Each query reveals one bit of s, thus we need to query the function 'n' times.

The Quantum Circuit :—



$$|\phi_3\rangle = \frac{1}{2^n} \sum_{(x,y)\in\{0,1\}^n} (-1)^{f(x)+x\cdot y} \cdot |y\rangle$$

→ from previous lecture.

The last state can be simplified to

$$|\phi_3\rangle = \frac{1}{2^n} \sum_{x\in\{0,1\}^n} (-1)^{s\cdot x + x\cdot y}$$

$$|\phi_3\rangle = \frac{1}{2^n} \sum_{x\in\{0,1\}^n} (-1)^{x(s\oplus y)}$$

$$|\phi_3\rangle = \frac{1}{2^n} \prod_{j=0}^{n-1} \left[\sum_{x_j=0}^{1} (-1)^{(s_j \oplus y_j) x_j} \right]$$

So if $s=y$ then this value is (1).

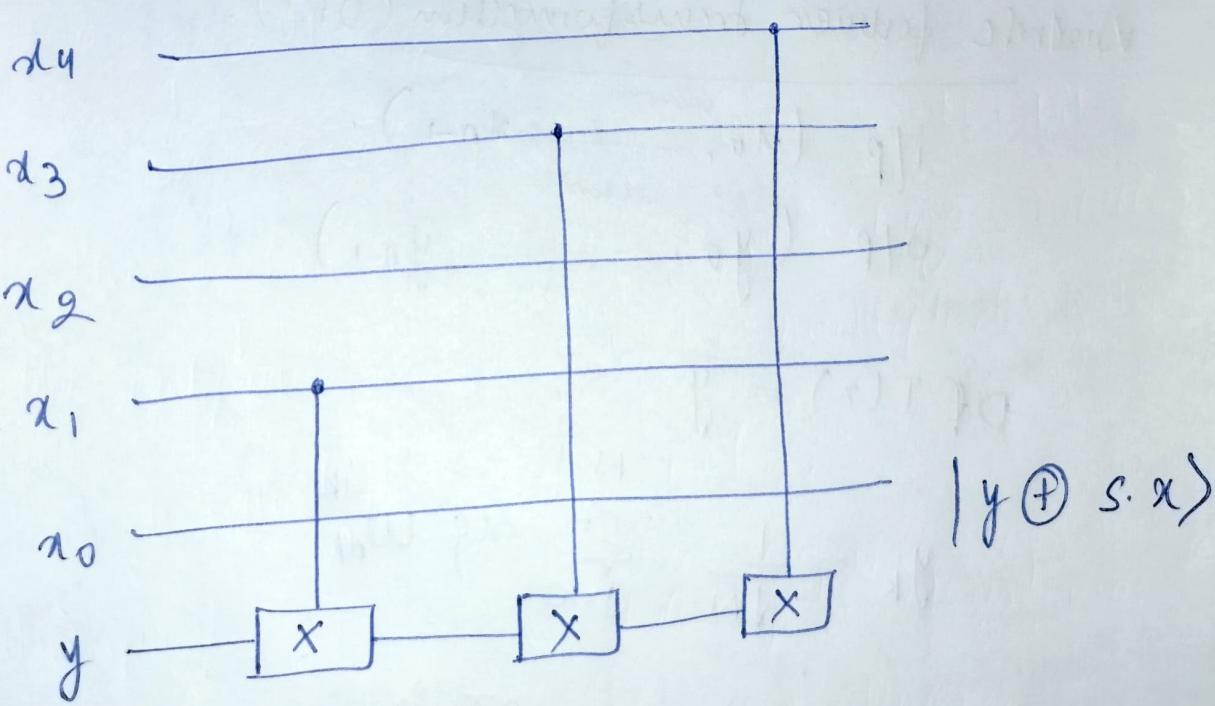
if $s \neq y$ then there exist i such that $s_i \oplus y_i = 1$, so the output in this case will be $|0\rangle$.

So, the final output will be

$$|s\rangle|1\rangle$$

example Consider $n=5$, $s = \underset{x_4}{1}\underset{x_3}{1}\underset{x_2}{0}\underset{x_1}{1}\underset{x_0}{0}$,

from here implement U_f



verify BV algorithm for this construction,

Since the final o/p of BV algo is

$$|s\rangle \otimes |1\rangle$$

$$|\phi_3\rangle = \frac{1}{2^{3/2}} \prod_{j=0}^{3-1} \left| \sum_{x_j=0}^{1} (-1)^{(s_j \oplus y_j) x_j} \right|$$

Quantum fourier Transformation :-

We have about,

Discrete fourier transformation (DFT):

i/p $(x_0, \text{——} x_{n-1})$ vector

o/p $(y_0, \text{——} y_{n-1})$ vector

$$DFT(\bar{x}) = \bar{y}$$

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \omega_N^{jk}$$

(omega) $\omega_N^{jk} = e^{\frac{2\pi i jk}{N}}$

$$\omega_N^N = e^{2\pi i} = 1$$

QFT :- for multi qubit state.

$$|x\rangle = \sum_{j=0}^{N-1} x_j |j\rangle$$

if $j=1$ or single bit quantum state

$N = 2$,

$$|x\rangle = x_0|0\rangle + x_1|1\rangle$$

$$QFT(|x\rangle) = |y\rangle$$

$$|y\rangle = \sum_{k=0}^{N-1} y_k |k\rangle$$

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \omega_N^{jk}$$

$$\omega_N^{jk} = e^{\frac{2\pi i \cdot jk}{N}}$$ ∵ [ω (omega) is n^{th} root of unity]

$$\omega_N^N = 1$$

Concept

QFT

I/P → z basis ($|0\rangle, |1\rangle$) means

↓ transform

O/P → fourier basis

State in computational basis —QFT→ fourier basis

DFT

signal time domain

↓ transform

freq. domain

Now : let us consider a circuit for a single qubit state, and apply QFT on that,

Single qubit system,

QFT is —

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

$QFT(|\psi\rangle)$

$N = 2 \Rightarrow x_0 = \alpha, \; x_1 = \beta$

$$y_0 = \frac{1}{\sqrt{2}}\left(\alpha \cdot e^{\frac{2\pi i \times 0 \times 0}{2}} + \beta \cdot e^{\frac{2\pi i \cdot \times 1 \cdot \times 0}{2}} \right)$$

$[j=0, k=0]$ $\quad$ $[j=1, k=0]$

$\because [k=0]$

$$y_0 = \frac{1}{\sqrt{2}}\left[\alpha \cdot e^0 + \beta \cdot e^0 \right]$$

$\because [e^0 = 1]$

$$\boxed{y_0 = \frac{1}{\sqrt{2}}(\alpha + \beta)}$$

Now,

$$y_1 = \frac{1}{\sqrt{2}}\left(\alpha \cdot e^{\frac{2\pi i \times 0 \wedge 1}{2}} + \beta \cdot e^{\frac{2\pi i \cdot 1 \times 1}{2}} \right)$$

$[j=0, k=1]$ $\quad$ $[j=1, k=1]$

$$y_1 = \frac{1}{\sqrt{2}}\left[\alpha + \beta \cdot e^{2\pi i} \right] = \frac{1}{\sqrt{2}}(\alpha - \beta)$$

$\because e^{i\pi} = -1$

$e^{2\pi i} = 1$

$$QFT|\psi\rangle = \frac{1}{\sqrt{2}}(\alpha+\beta)|0\rangle + \frac{1}{\sqrt{2}}(\alpha-\beta)|1\rangle$$

→ H gate.

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

$$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

} H gate is sufficient to do single bit QFT.

for 2-qubit system, QFT.

$$|\psi\rangle = \alpha_{00}|0\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$$

$$y_0 = \frac{1}{2}(\alpha_{00} + \alpha_{01} + \alpha_{10} + \alpha_{11})$$

$$y_1 = \frac{1}{2}(\alpha_{00} + e^{i\pi/2}\alpha_{01} + e^{i\pi}\alpha_{10} + e^{3\pi/2}\alpha_{11})$$

$$y_2 = \frac{1}{2}(\alpha_{00} + \alpha_{01}e^{i\pi} + \alpha_{10}\cdot e^{2\pi i^0} + \alpha_{11}\cdot e^{3i\pi})$$

$$y_3 = \frac{1}{2}(\alpha_{00} + \alpha_{01}e^{\frac{3\pi i}{2}} + \alpha_{10}\cdot e^{3\pi i} + \alpha_{11}e^{\frac{9\pi i}{2}})$$

→ This is more complicated than single qubit Quantum fourier transformation.

Generic Quantum Circuit for N-qubit system :-

$$QFT_N |x\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega_N^{xy} |y\rangle \quad \text{---- (1)}$$

$$QFT \rightarrow |x\rangle \quad (\overleftarrow{x}).$$

$$N = 2^n.$$

$$y = \{y_1, y_2 \text{ ---- } y_n\} \quad \because \; n = \log N, \; N = 2^n.$$

(omega) $\omega_N^{xy} = e^{\frac{2\pi i xy}{N}}$.

replace the values of y & ω in eq^n (1).

$$QFT_N |x\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{\frac{2\pi i xy}{N}} |y_1 y_2 \text{ ---- } y_n\rangle$$

$\because N = 2^n$.

$$\frac{y}{N} = \frac{y}{2^n} = \sum_{k=1}^{n} \frac{y_k}{2^k}.$$

$$QFT_N |x\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi i x \cdot \sum_{k=1}^{n} \frac{y_k}{2^k}} |y_1, y_2 \text{ ---- } y_n\rangle$$

$$= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \prod_{k=1}^{n} e^{\frac{2\pi i x y_k}{2^k}} |y_1, y_2 \text{ ---- } y_n\rangle.$$

we can take,

$$\sum_{y=0}^{N-1} = \sum_{y_1=0}^{1} \cdot \sum_{y_2=0}^{1} \cdots \sum_{y_n=0}^{1}$$

after rearranging we can get,

$$QFT_N|x\rangle = \frac{1}{\sqrt{N}}\left[\left(|0\rangle + e^{\frac{2\pi i x}{2}}|1\rangle\right)\otimes\left(|0\rangle + e^{\frac{2\pi i x}{2^2}}|1\rangle\right)\otimes \cdots \otimes\left(|0\rangle + e^{\frac{2\pi i x}{2^n}}|1\rangle\right)\right].$$

Now,

Summarizing entire discussion,

if $|x\rangle = |x_1, x_2, \cdots x_n\rangle = |x_1\rangle \otimes |x_2\rangle \otimes \cdots |x_n\rangle$

$$|\tilde{x}\rangle = \frac{1}{\sqrt{N}}\left[\left(|0\rangle + e^{\frac{2\pi i x}{2}}|1\rangle\right)\otimes\left(|0\rangle + e^{\frac{2\pi i x}{2^2}}|1\rangle\right)\otimes \cdots \otimes\left(|0\rangle + e^{\frac{2\pi i x}{2^n}}|1\rangle\right)\right].$$

example

Consider $n=3 \Rightarrow N = 2^3 = 8$

$$|x\rangle = |5\rangle = |101\rangle$$

$$QFT|x\rangle = \frac{1}{\sqrt{8}}\left(|0\rangle + e^{\frac{2\pi i \cdot 5}{2}}|1\rangle\right)\otimes \left(|0\rangle + e^{\frac{2\pi i \cdot 5}{2^2}}|1\rangle\right)\otimes \left(|0\rangle + e^{\frac{2\pi i \cdot 5}{2^3}}|1\rangle\right).$$

$$QFT|x\rangle = \frac{1}{\sqrt{8}}\Big[\left(|0\rangle + e^{5\cdot\pi i}|1\rangle\right)\otimes \left(|0\rangle + e^{\frac{5\pi i}{2}}|1\rangle\right)\otimes \left(|0\rangle + e^{\frac{5\cdot\pi i}{4}}|1\rangle\right)\Big]$$

Circuit :—

1 Qubit QFT → H gate

$UROT_k$ (Unitary Rotation) :— (New gate required).

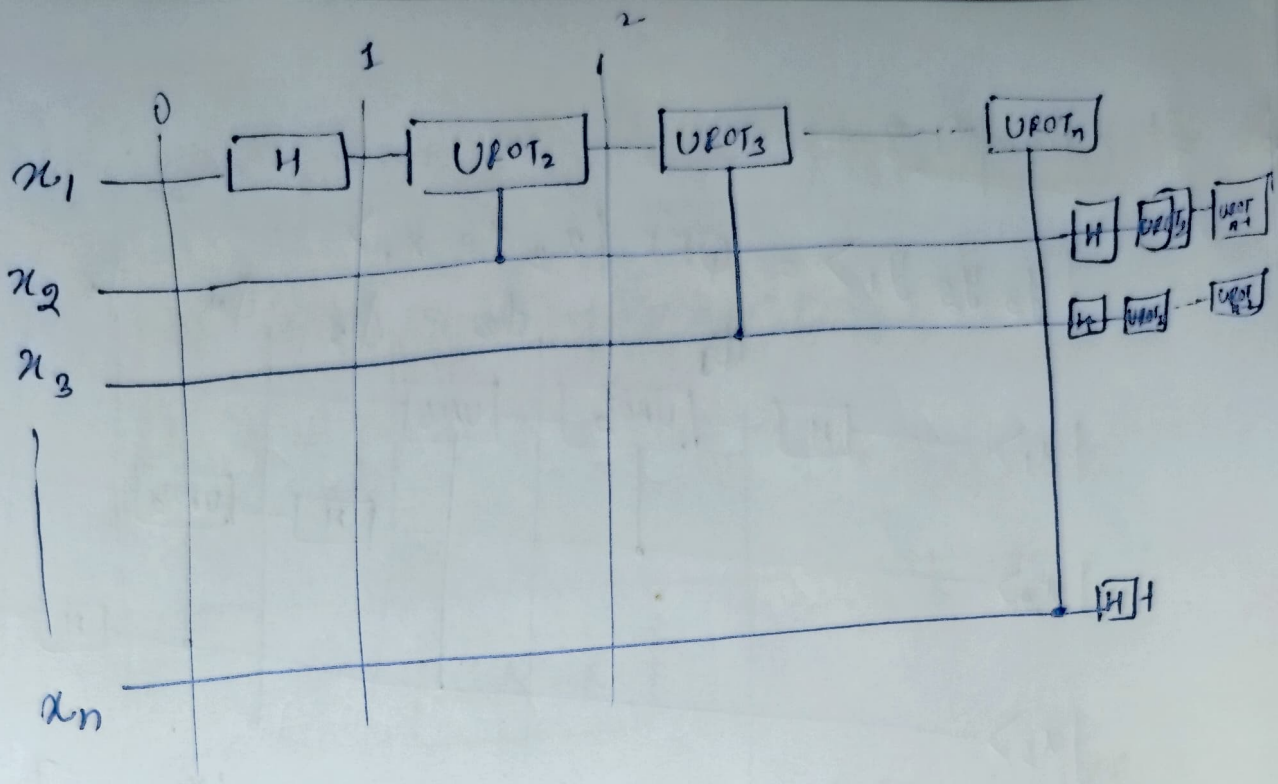
$$UROT_k|x_j\rangle = e^{\frac{2\pi i x_j}{2^k}}|x_j\rangle$$

$$UROT_k = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{2\pi i}{2^k}} \end{bmatrix}$$

Control ROT = CROT

$$CROT_k = \begin{bmatrix} I & 0 \\ 0 & UROT_k \end{bmatrix}.$$

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$



state 0: $|x_1 x_2 \cdots x_n\rangle$

state 1: $\frac{1}{\sqrt{2}}\left(|0\rangle + e^{\frac{2\pi i x_1}{2}}|1\rangle\right) \otimes |x_2 x_3 \cdots x_n\rangle$

state 2: $\frac{1}{\sqrt{2}}\left(|0\rangle + e^{\frac{2\pi i x_2}{2^2}} \cdot e^{\frac{2\pi i \cdot x_1}{2}}|1\rangle\right) \otimes |x_2 x_3 \cdots x_n\rangle$

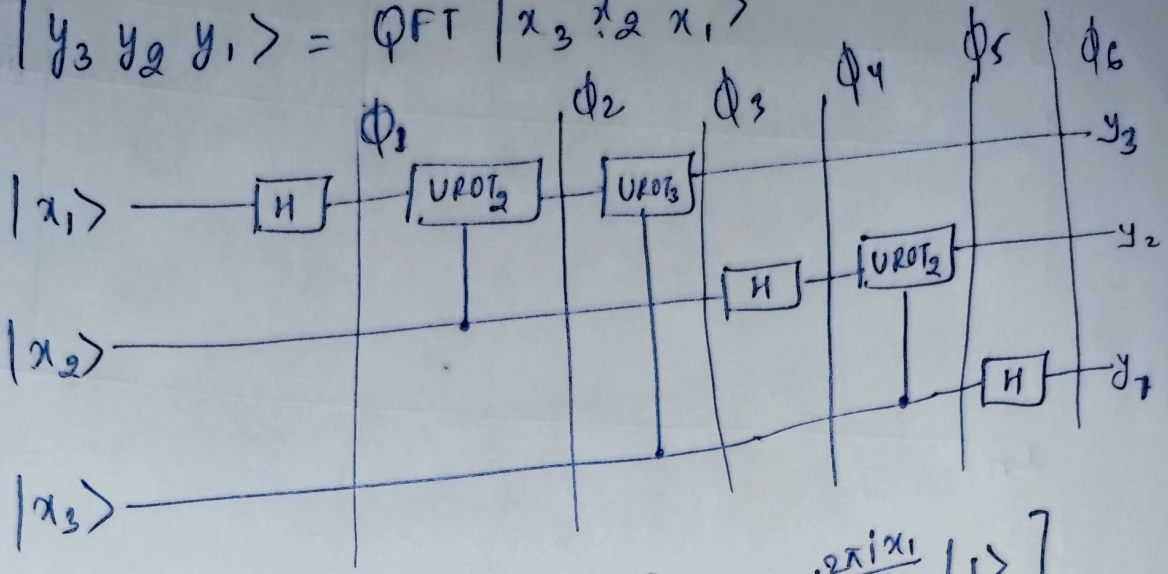
at step n :-

$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{\frac{2\pi i x_n}{2^n}} \cdots e^{\frac{2\pi i x_1}{2}}|1\rangle\right) \otimes |x_2 x_3 \cdots x_n\rangle$

this way we can find terms of quantum fourier transformation.

example :-

$|y_3\ y_2\ y_1\rangle = QFT\ |x_3\ x_2\ x_1\rangle$



$$|\phi_1\rangle = |x_3\rangle \otimes |x_2\rangle \otimes \frac{1}{\sqrt{2}}\left[|0\rangle + e^{\frac{2\pi i x_1}{2}}|1\rangle\right]$$

$$|\phi_2\rangle = \left(|0\rangle + \left(e^{\frac{2\pi i x_2}{2^2}} \cdot e^{\frac{2\pi i x_1}{2}}\right)|1\rangle\right)\frac{1}{\sqrt{2}} \otimes |x_2\rangle \otimes |x_3\rangle$$

$$|\phi_3\rangle = \left(|0\rangle + \left(e^{\frac{2\pi i x_3}{2^3}} \cdot e^{\frac{2\pi i x_2}{2^2}} \cdot e^{\frac{2\pi i x_1}{2}}\right)|1\rangle\right)\frac{1}{\sqrt{2}} \otimes |x_2\rangle \otimes |x_3\rangle$$

$$= |t_1\rangle$$

(1-gate)

$$|\phi_4\rangle = |x_3\rangle \otimes \frac{1}{\sqrt{2}}\left[|0\rangle + e^{\frac{2\pi i x_2}{2}}|1\rangle\right] \otimes |t_1\rangle$$

$$|\phi_5\rangle = |x_3\rangle \otimes \underbrace{\frac{1}{\sqrt{2}}\left[|0\rangle + \left(e^{\frac{2\pi i}{2^2} x_3} \cdot e^{\frac{2\pi i}{2} x_2}\right)|1\rangle\right]}_{|t_2\rangle} \otimes |t_1\rangle$$

$$|\phi_6\rangle = \underbrace{\frac{1}{\sqrt{2}}\left[|0\rangle + e^{\frac{2\pi i \cdot x_3}{2}}|1\rangle\right] \otimes |t_2\rangle \otimes |t_1\rangle}_{|t_3\rangle}$$

$|t_1\rangle = |y_3\rangle$ ~~$|t_2\rangle \neq |y_2\rangle$~~

$|t_3\rangle = |y_1\rangle$

Week-6

Shor's Algorithm :: —— to factorize $N = p \times q$

(1) pick a, $\gcd(a, N) = 1$, a, N are coprime

(2) find out r, such that $a^r \pmod N = 1$.

$r \rightarrow$ order

(3) if r is even

$$x \equiv a^{r/2} \pmod N$$

(4) if $(x+1) \not\equiv 0 \pmod N$

$\gcd[(x+1), N]$ or $\gcd[(x-1), N]$ } contain either p or q.

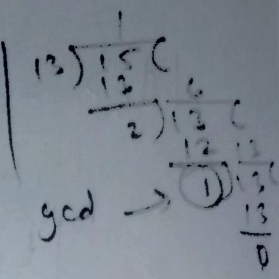
else

find another a, repeat the same process again.

Example

$N = 15 = 3 \times 5$ (q) (p)

(1) pick $a = 13$, $\gcd(13, 15) = 1$ (coprime)



(2)

$a^r \pmod N = 1$

$13^r \pmod{15} = 1$, $r = 4$ → important point to obtain value of (r) .

$13^4 \pmod{15} = 1$

$28561 \bmod 15 = 1$

↓

we use quantum circuit for large values.

(3) if $r = 4$ is even ✓

$x \equiv a^{r/2} \bmod N$

$x \equiv 13^2 \bmod 15 = 4$

Now, check

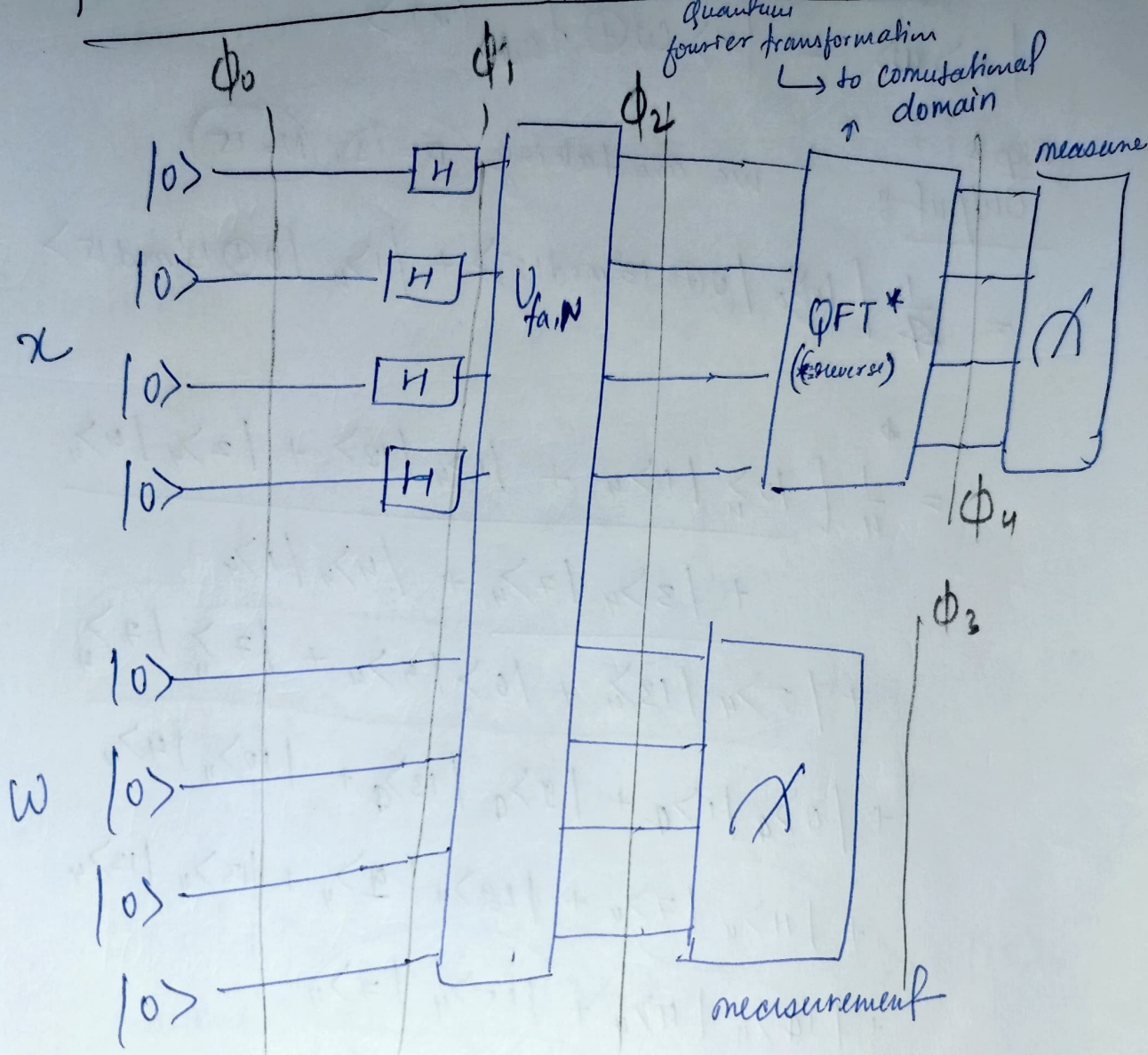
$x + 1 = 5 \bmod 15 \neq 0 \bmod 15$

$\gcd(5, 15) \rightarrow (5) = p$

or $\gcd(3, 15) \rightarrow (3) = q$

} factors

Quantum circuit to implement Shor's algo :-



$$f_{a,N} = a^x \pmod N$$

step 0: ⊗ of all the bits

step 1: $\left[H^{\otimes 4} |0\rangle \right] |0\rangle^{\otimes 4}$

$$= \frac{1}{4} \left[|0\rangle_4 + |1\rangle_4 + |2\rangle_4 + \cdots |15\rangle_4 \right] |0\rangle_4$$

Step 2: →

Now → function in middle

two input

$|x\rangle|w\rangle \rightarrow |x\rangle|w \oplus f_{a,N}(x)\rangle$

Step 2

Output: we are taking $a = 13$, $N = 15$.

$$= \frac{1}{q}\left[|0\rangle_4 |0 \oplus 13^0 \bmod 15\rangle + |1\rangle_4 |0 \oplus 13^1 \bmod 15\rangle \ldots\right]$$

$$= \frac{1}{4}\Big[|0\rangle_4|1\rangle_4 + |1\rangle_4|13\rangle + |2\rangle_4|4\rangle_4 + |3\rangle_4|7\rangle_4 + |4\rangle_4|1\rangle_4 + |5\rangle_4|13\rangle_4 + |6\rangle_4|4\rangle_4 + |7\rangle_4|7\rangle_4 + |8\rangle_4|1\rangle_4 + |9\rangle_4|13\rangle_4 + |10\rangle_4|4\rangle_4 + |11\rangle_4|7\rangle_4 + |12\rangle_4|1\rangle_4 + |13\rangle_4|13\rangle_4 + |14\rangle_4|4\rangle_4 + |15\rangle_4|7\rangle_4\Big]$$

So, we conclude that the value of

$|x\rangle$ can be 1 to 15

& value of $w \oplus f_{a,N}(x)$ → can be 1, 13, 4, 7

So, if we measure the lower four bits

we can get either: 1, 13, 4 or 7.

Step 3 :-

measure $(\omega \oplus f_{a,N}(x)) \rightarrow 7$ (let's assume)

$$x \rightarrow \frac{1}{4}\left[|3\rangle_4 + |7\rangle_4 + |11\rangle_4 + |15\rangle_4 \right]$$

Step 4 :-

$QFT^* \rightarrow QFT^\dagger$ [QFT inverse].

$$QFT|x\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{\frac{2\pi i}{N} xy} |y\rangle$$

$$QFT^\dagger |\tilde{x}\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{-\frac{2\pi i}{N} \tilde{x} y} |y\rangle$$

$\rightarrow$ $QFT^* |x\rangle$

where $|x\rangle = \frac{1}{4}\left[|3\rangle_4 + |7\rangle_4 + |11\rangle_4 + |15\rangle_4 \right]$

$$\rightarrow QFT^* |3\rangle = \frac{1}{4} \sum_{y=0}^{15} e^{-\frac{2\pi i 3y}{16}} |y\rangle$$

$$\rightarrow QFT^* |7\rangle = \frac{1}{4} \sum_{y=0}^{15} e^{-\frac{2\pi i 7y}{16}} |y\rangle$$

$$\rightarrow QFT^* |11\rangle = \frac{1}{4} \sum_{y=0}^{15} e^{-\frac{2\pi i (11) y}{16}} |y\rangle$$

$$\rightarrow QFT^* |15\rangle = \frac{1}{4} \sum_{y=0}^{15} e^{-\frac{2\pi i (15) y}{16}} |y\rangle$$

Combining all of them together,

$$QFT^{*}|x\rangle = \frac{1}{8}\sum_{y=0}^{15}\left[e^{-\frac{3\pi i}{8}y} + e^{-\frac{7\pi i}{8}y} + e^{-\frac{11\pi i}{8}y} + e^{-\frac{15\pi i}{8}y}\right]|y\rangle$$

Should be → 8 → because of probability

if we compute value of this expression over every possible value of 'y' then we will get,

$$QFT^{*}|x\rangle = \frac{1}{4}\left[4|0\rangle_4 + 4i|4\rangle_4 - 4|8\rangle_4 - 4i|12\rangle_4\right]$$

Here, only for $y = 0, 4, 8, 12$ we get some terms but for all other values (like $y=1$) they cancel out each other.

0, 4, 8, 12 → with equal probability.

we will measure only one of these.

$j \cdot \frac{N}{r}$, $j \in \mathbb{Z}$; N → no. of points on QFT, i.e. = 16.

If we measure 4

$\frac{j \cdot 16}{r}$, we can get $r = 4$ for $j = 1$.

if we measure 8,

$$j \cdot \frac{16}{r} = 8$$

$j = 1$, $r = 2$ → check if this works or not.

$j = 2$, $r = 4$

if we measure 12

$$j \cdot \frac{16}{r} = 12$$

$j = 1$, $r = \frac{4}{3}$ | $j = 2$, $r = \frac{16 \times 2}{12} = \frac{8}{3}$ | $j = 3$, $r = 4$,

if we measure 0

$$\frac{j \cdot 16}{r} = 0$$

$j = 0$, $r = 0$

No of qubit required in 2n bit

16 qubit → required for 8 qubit

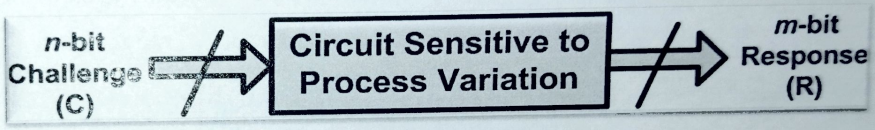
for large bits we will require very large no. of qubits to implement this.

Week 7

Physically Unclonable Function based Authentication Protocols

Urbi Chatterjee
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur.

Physically Unclonable Functions (PUFs)

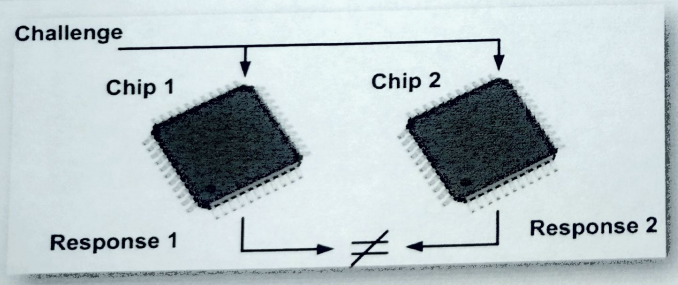


Unclonable, unpredictable instance specific system behavior

Easy to design and fabricate, but infeasible to replicate

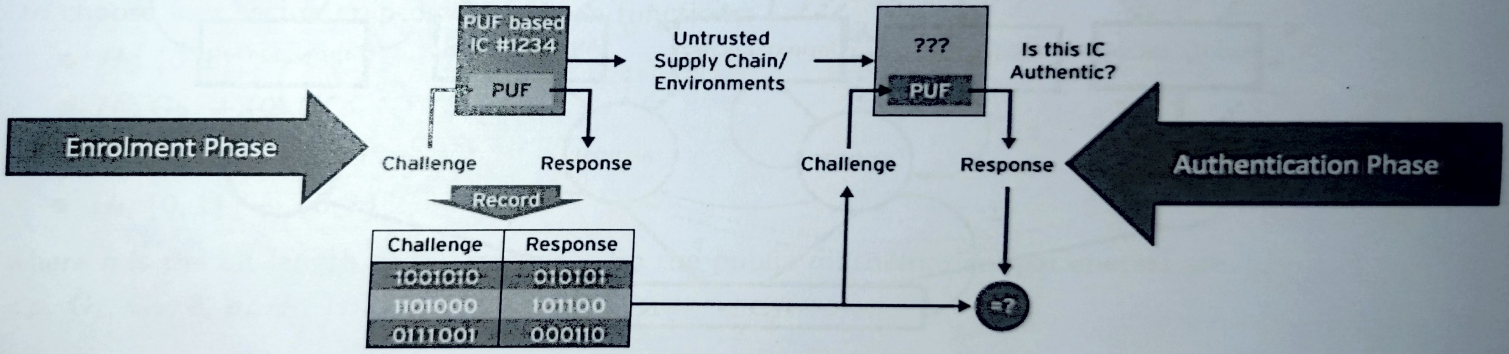
Offloads computational expense

Relatively low hardware overhead

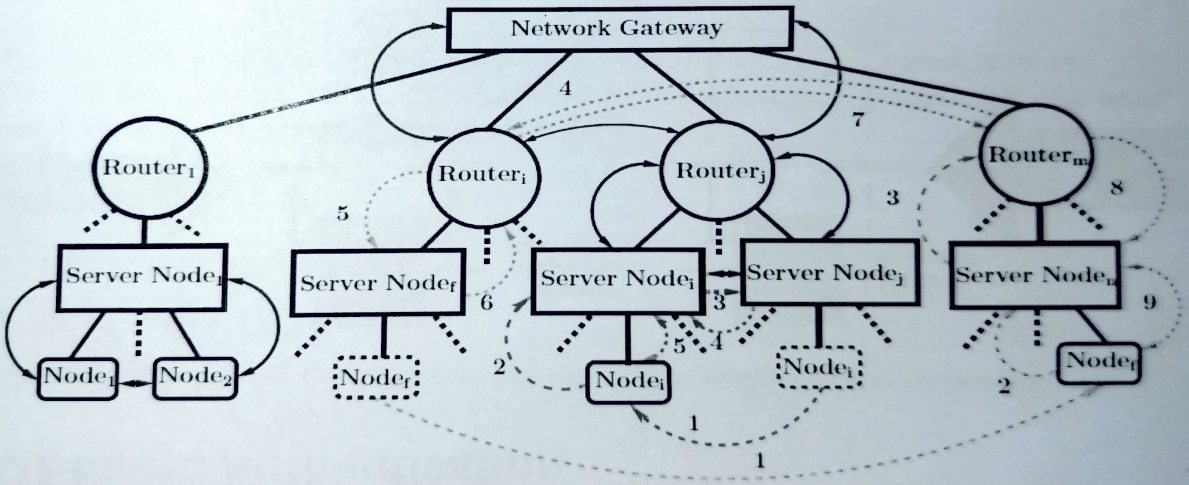


Good candidate as unconventional cryptographic primitive for resource constrained devices!

Naïve PUF based Authentication



Secure Communication Mechanism in Different Levels of IoT Architecture



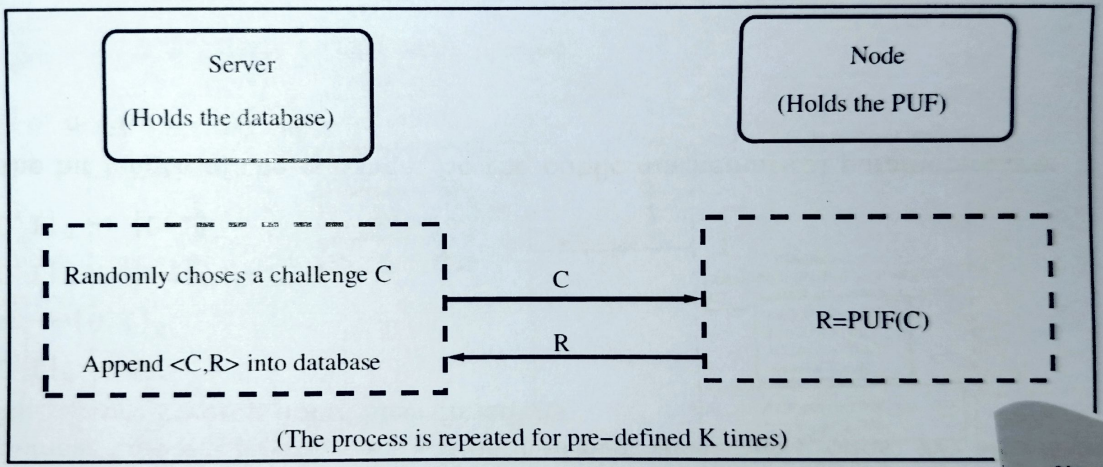
Public Mathematical Parameters

For some large prime value p, two groups $\mathbb{G}_1$, $\mathbb{G}_2$ of order p are generated, and an admissible bilinear map $\hat{e}$: $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is defined over these two groups. We also need to choose four secure cryptographic hash functions:

- H_1: $\{0,1\}^n \rightarrow \mathbb{G}_1^*$
- H_2: $\mathbb{G}_2 \rightarrow \{0,1\}^n$
- H_3: $\{0,1\}^n \times \{0,1\}^n \rightarrow Z_p^*$
- H_4: $\{0,1\}^n \rightarrow \{0,1\}^n$

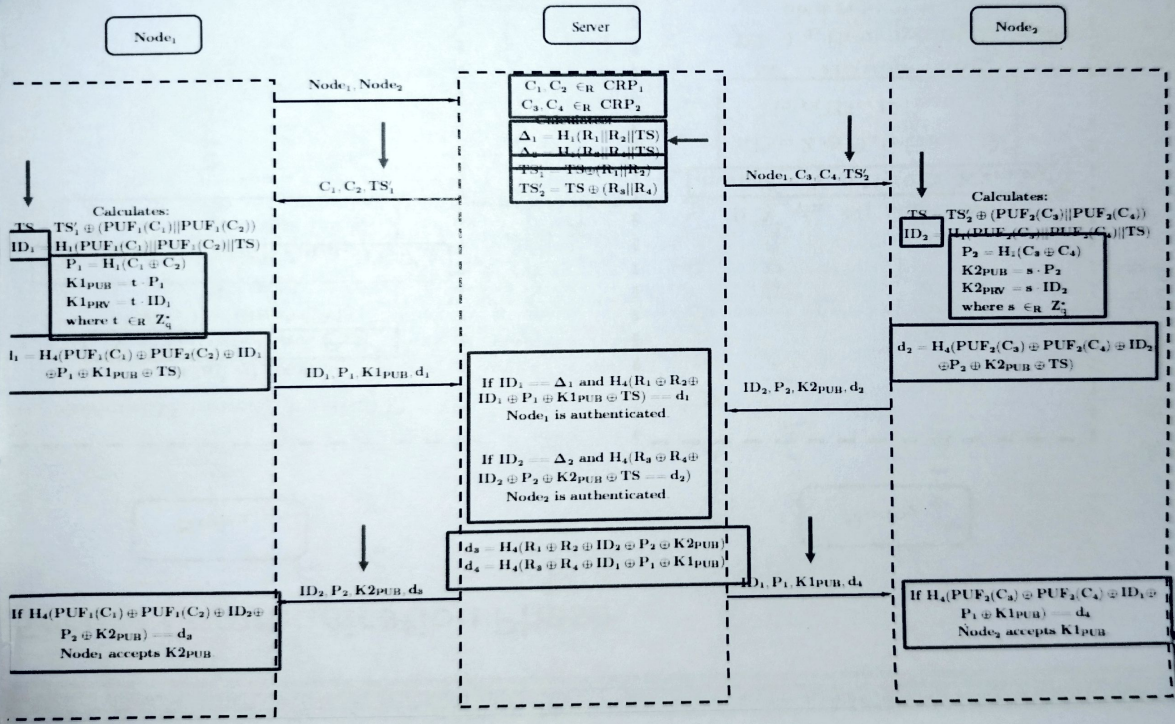
where n is the bit length of the message. So the public mathematical parameters are: $<p, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, H_1, H_2, H_3, H_4>$.

Enrolment Phase



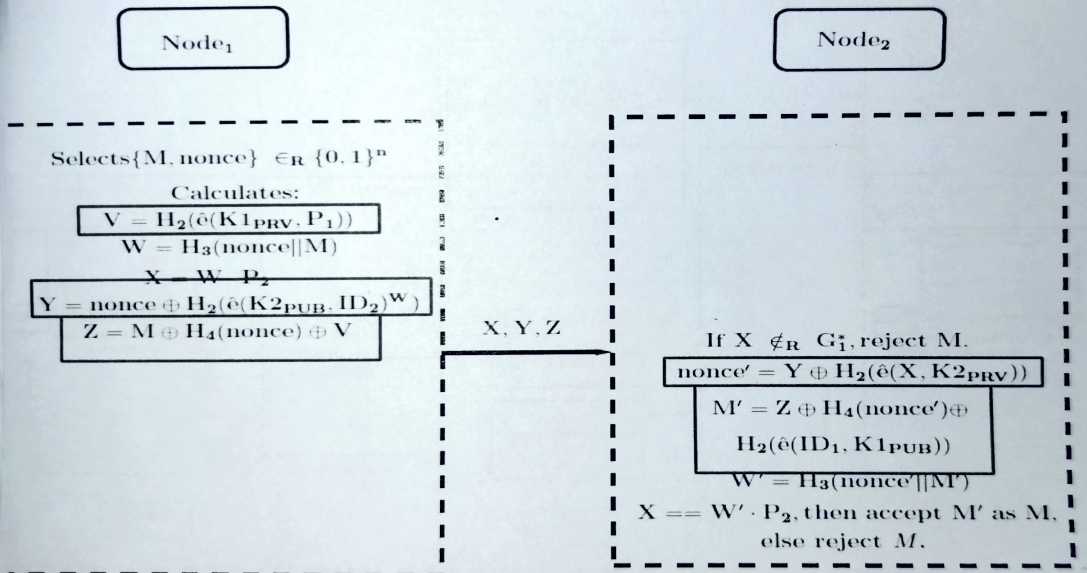
No storage in resource constraint nodes in the lowest level of the architecture

Authentication and Key Sharing Phase



- No requirement of CA
- No requirement of secure channel for transferring SKs.
- Mutual Authentication
- TS for replay attack
- New PK/SK for each protocol run.

Secure Communication Phase



$K1_{PRV} = t \cdot ID_1$, $\hat{e}(K1_{PRV}, P_1) = \hat{e}(t \cdot ID_1, P_1) = \hat{e}(ID_1, P_1)^t$

$$M' = Z \oplus H_4(nonce') \oplus H_2(\hat{e}(ID_1, K1_{PUB}$$
$$= Z \oplus H_4(nonce') \oplus H_2(\hat{e}(ID_1, t \cdot P_1))$$
$$= Z \oplus H_4(nonce') \oplus H_2(\hat{e}(ID_1, P_1)^t)$$

$$Y = nonce \oplus H_2(\hat{e}(K2_{PUB}, ID_2)^W)$$
$$= nonce \oplus H_2(\hat{e}(s \cdot P_2, ID_2)^W)$$
$$= nonce \oplus H_2(\hat{e}(P_2, ID_2)^{s \cdot W})$$

$$nonce' = Y \oplus H_2(\hat{e}(X, K2_{PRV})$$
$$= Y \oplus H_2(\hat{e}(W \cdot P_2, s \cdot ID_2)$$
$$= Y \oplus H_2(\hat{e}(P_2, ID_2)^{s \cdot W})$$

Week 8 — 15/Nov/2024

#NTRU:—

→ It is a lattice-based cryptography

→ It is a public key cryptography

→ The current version of NTRU is called NTRUEncrypt.

→ It is very fast (algo) cryptosystem that is easy to implement

→ It is based on shortest vector & closest vector problem.

→ NTRUEncrypt is defined in terms of three parameters, N, p, and q which are fixed integers.

→ Computations are performed in the Ring,

$$R = \mathbb{Z}[x]/(x^N - 1).$$

multiplication of two polynomials is easy in R; it suffices to compute the product of two polynomia in $\mathbb{Z}[x]$ & then reduce all exponents modulo N.

for example →

Suppose $N = 3$

and we want to compute the product

$(x^2 + 3x + 1)(2x^2 + x - 4)$ in R.

We'll compute it as follows:

$$(x^2 + 3x + 1)(2x^2 + x - 4)$$

$$= 2x^4 + x^3 - 4x^2 + 6x^3 + 3x^2 - 12x$$

$$+ 2x^2 + x - 4$$

$$= (2x^4 + 7x^3 + x^2 - 11x - 4) \pmod 3$$

↑ only for powers

$$= 2x + 7 + x^2 - 11x - 4$$

$$= x^2 - 9x + 3 \rightarrow \text{can of degree } (N-1)$$

It is often convenient to represent a polynomial in R by its vector of coefficients:

$$a(x) = \sum_{i=0}^{N-1} a_i x^i \text{ corresponds to } a = (a_0, a_1, \ldots a_{N-1})$$

Ily another vector,

$$b(x) = \sum_{i=0}^{N-1} b_i x_i$$

and

$$c(x) = a(x) \cdot b(x)$$

$$= \sum_{i=0}^{N-1} c_i x^i$$

the corresponding coeff. vectors know the relation,

$$c = a \circledast b = a * b$$

where '*' is a convolution operation.

for $0 \le i \le N-1$, we have

$$c_i = \sum_{j=0}^{N-1} a_j b_{i-j}$$, where all subscripts are reduced modulo N

Main properties in NTRUEncrypt encryption & decryption

→ coefficients will be reduced modulo p or modulo q.

→ q will be quite a bit larger than p

→ q & p should be relatively prime.

→ p should be odd.

→ the values p = 3 & q = 2048 are popular choices.

→ N is usually taken to be a prime, .

N = 401 is a currently recommended value.

* Centered modular reduction :—

Definition :— for an odd integer n and integers a and b, define

$$a \bmod_s n = b \quad \text{if } a \equiv b \pmod{n} \text{ and } -\frac{n-1}{2} \leq b \leq \frac{n}{2}$$

for example,

$$a \bmod_s 5 \in \{-2, -1, 0, 1, 2\}$$

whereas $a \bmod 5 \in \{0, 1, 2, 3, 4\}$.

* Defining public & private key :—

→ At first F(x) & G(x) are secret polynomials chosen from R .

→ All coefficients of F(x) and G(x) will be in the set $\{-1, 0, 1\}$.

→ Next, define $f(x) = 1 + pF(x)$ and

$$g(x) = pG(x)$$

→ finally, compute $f^{-1}(x)$ in the ring R mod q.

→ then compute, $\boxed{h(x) = f^{-1}(x)\, g(x) \bmod q}$

→ After this is done, F and G can be discarded.

→ the public key is the coefficient vector h and the private key is the coefficient vector f.

→ the polynomial g(x) is used in the construction of the public key h(x); g(x) is not part of the public or private key, but it should be kept secret & then discarded after h(x) is formed.

→ the polynomial $f^{-1}(x)$ can be computed using extended euclidean algorithm for polynomials.

computing $f^{-1}(x)$:-

let $c(x) = \gcd(f(x), x^N - 1)$

which is computed in $\mathbb{Z}_q[x]$.

→ the extended euclidean algo computed polynomials $a(x), b(x) \in Z_q[x]$ such that

$$a(x) \cdot f(x) + b(x)(x^N - 1) = c(x)$$

then $f^{-1}(x)$ exists if and only if $c(x) = 1$

further, if $c(x) = 1$

then, $\boxed{f^{-1}(x) = a(x) \bmod q}$

Encryption :-

→ A plaintext m is an N-tuple in the set $\{-1, 0, 1\}$

→ the encryption process in NTRUEncrypt is randomized

→ first, $r \in \{-1, 0, 1\}^N$ is choosen uniformly at random from a specified subset of R.

→ the ciphertext,

$$\boxed{e = r * h + m \bmod q}$$

Decryption :-

→ to decrypt a ciphertext e, perform following :

① compute $a = f * e \bmod p$

② compute $m' = a \bmod q$.

→ if all goes well, it will be the case $m' = m$.

→ first, it is easy to verify that

$$a \equiv r * g + f * m \pmod{q}$$

as we know,

$$\begin{aligned} a &\equiv f * y \pmod{q} \\ &\equiv f * (r * h + m) \pmod{q} \\ &\equiv f * (r * f^{-1} * g + m) \pmod{q} \\ &\equiv r * g + f * m \pmod{q} \end{aligned}$$

Now, suppose that this congruence is actually an equality in R, i.e.

$$a = r * g + f * m \qquad (1)$$

this happens if & only if every coeff. of $r * g + f * m$ lies in the interval $\left[-\frac{q-1}{2}, \frac{q}{2}\right]$

which will hold high probability if the parameters of the system are chosen in a suitable way.

Now, assuming eqn ① holds & reducing modulo p, we have

$$a \equiv r*g + f*m \pmod p$$

$$\equiv \underline{r*pG} + (1+pF)*m \pmod p$$

$$\equiv (0 + m + 0) \underset{\text{mod } p}{\equiv} \underline{m \pmod p}$$

from this relation, we set that

$$m = a \bmod p$$

because all the coeff. of m are in the set $\{-1, 0, 1\}$. Therefore, the ciphertext is decrypted correctly.

Example :— definition :—

Suppose p, q and N are integers, where $q >> p$ and p, q are relatively prime, p is odd and N is prime.

→ values :-

$p = 3, q = 2040, N = 401$

Let $P = \{-1, 0, 1\}^N$ and $C = (\mathbb{Z}_q)^N$.

Choose $F, G \in \{-1, 0, 1\}^N$

let $f = 1 + pF$ and $g = pG$

& define $h = f^{-1} g \bmod q$.

→ the associated key is $k = (f, h)$
where f is private & h is public.

→ Now define,

$$e_k(m) = y = r * h + m \bmod q$$

for a randomly chosen $r \in \{-1, 0, 1\}^N$
and

$$d_k(y) = (f * y \bmod q) \bmod p.$$

Now example :→

Suppose, $N = 23$, $p = 3$, $q = 31$

let $F(x) = x^{10} - x^9 + x^8 - x^4 - x^2$, $f(x) = 1 + pF(x)$

so, $f(x) = 3x^{10} - 3x^9 + 3x^8 - 3x^4 - 3x^2 + 1$

& $G(x) = x^{17} + x^{12} + x^9 - x^3 - x$

$g(x) = pG(x)$

so, $g(x) = 3x^{17} + 3x^{12} + 3x^9 - 3x^3 - 3x$.

Next we compute,

$$h(x) = -13x^{22} - 15x^{21} + 12x^{19} - 14x^{18} + 0x^{16} - 14x^{15} - 6x^{14} + 14x^{13} - 3x^{12} + 7x^{11} - 5x^{10} - 14x^9 + 3x^8 + 10x^7 + 5x^6 - 0x^5 + 4x^2 + x + 0$$

Suppose we wish to encrypt

plaintext, $m(x) = x^{15} - x^{12} + x^7 - 1$

and we choose,

$$r(x) = x^{19} + x^{10} + x^6 - x^2.$$

$q = 31$.

then,

$$y = r * h + m \bmod q.$$

$$y = [-13x^{41} - 15x^{40} + 12x^{30} - 14x^{37} + 0x^{35} - 14x^{34} - 6x^{33} + 14x^{32} - 3x^{31} + 7x^{30} - 5x^{29} - 14x^{28} + 3x^{27} + 10x^{26} + 5x^{25} - 0x^{24} + 4x^{21} + x^{20} + 0x^{19} - 13x^{32} - 15x^{31}$$

Not sure about this — skip

$$+ 12x^{29} - 14x^{28} + 0x^{26} - 14x^{25} - 6x^{24} + 14x^{23} - 3x^{22} + 7x^{21} - 5x^{20} - 14x^{19} + 3x^{18} + 10x^{17} + 5x^{16} - 0x^{15} + 4x^{12} + x^{1} + 0x^{10} - 13x^{28} - 15x^{27} + 12x^{25} - 14x^{24} + 0x^{22} - 14x^{21} - 6x^{20} + 14x^{19} - 3x^{18} + 7x^{17} - 5x^{16} - 14x^{1} + 3x^{14} + 10x^{13} + 5x^{12} - 0x^{11} + 4x^{8} + x^{7} + 0 + 13x^{24} + 15x^{23} - 12x^{21} + 14x^{20} - 0x^{18} + 14x^{17} + 6x^{16} - 14x^{15} + 3x^{14} - 7x^{13} + 5x^{12} + 14x^{11} - 3x^{10} - 10x^{9} - 5x^{8} + 0x^{7} - 4x^{4} - x^{3} - 0x^{2} + x^{15} - x^{12} + x^{7} - 1]$$

mod q.

$$y(x) = 5x^{22} - 15x^{21} + 4x^{20} + 8x^{19} + 10x^{18} - 15x^{17}$$
$$+ 6x^{16} + 8x^{15} - 8x^{14} + 3x^{13} - 10x^{12} - 7x^{11}$$
$$- x^{10} - 9x^9 + 12x^8 - 14x^7 + 15x^6 - 10x^5$$
$$+ 15x^4 - 14x^3 - 5x^2 - 15x - 3 .$$

→ the decryption process will begin,

$$a(x) = 6x^{22} + 3x^{21} - 6x^{20} - 3x^{19} - 3x^{17} + 7x^{15}$$
$$+ 6x^{13} - x^{12} - 9x^{11} + 3x^{10} + 3x^9 - 5x^7 + 6x^4$$
$$+ 3x^3 + 6x^2 - 3x + 5$$

reducing the coeff of $a(x)$ modulo 3 yields,

$$a(x) = x^{15} - x^{12} + x^7 - 1 .$$

which is plaintext.

lattices & security of NTRU :-

→ **Lattice** :- it is very similar to a vector space.

⇒ A real vector space can be defined by starting with a **basis**, which is a set of linearly independent vectors in $\mathbb{R}^n$ for some int. n.

∘ The vector space generated by the given basis consists of all linear combinations of basis vectors

→ If there are r vectors in the basis, then we have an r-dimentional vector space.

→ suppose r basis vectors are $b^1, \ldots b^r$. the vector space generated by this basis consists of all the vectors of the form →

$$\alpha_1 b^1 + \ldots + \alpha_r b^r.$$

where $\alpha_1, \ldots \alpha_r$ are real no.

In lattice, the vectors are integers linear combinations of basis vectors

$$\alpha_1 b^1 + \ldots + \alpha_r b^r$$

where $\alpha_1, \ldots \alpha_r$ are integers.

Norm of vector v is —

$$\|v\| = \sqrt{\sum_{i=1}^{n} v_i^2}$$

Shortest Vector problem :—

Instance - A basis for a lattice $\mathcal{L}$ in $\mathbb{R}^n$

find - A vector $v \in \mathcal{L}$, $v \neq (0, \ldots, 0)$ such that $\|v\|$ is minimized. Such a vector v is called a shortest vector in $\mathcal{L}$.

Closest vector problem :—

Instance - A basis for a lattice $\mathcal{L}$ in $\mathbb{R}^n$ & a vector $w \in \mathbb{R}^n$ that is not in $\mathcal{L}$.

find - A vector $v \in \mathcal{L}$ such that $\|v - w\|$ is minimized. Such a vector v is called a closest vector to $w \in \mathcal{L}$.

Adversary :—

Can break NTRUEncrypt would be to compute the polynomials $f(x)$ & $g(x)$ that were used to construct the public key h.

Learning with errors :- (LWE problem)

Instance - A prime q, an integer n, a discrete random variable ξ with probability distribution χ defined on the set Z_q and m samples $(a^i, b^i) \in (Z_q)^{n+1}$.

The m samples are all constructed from a secret $s = (s_1, s_2, \ldots, s_n) \in (Z_q)^n$.

for $1 \le i \le m$, $a^i = (a_1^i, \ldots, a_n^i)$ is chosen uniformly at a random from $(Z_q)^n$, e^i is chosen using the probability distribution χ and

$$b^i = e^i + \sum_{j=1}^{n} a_j^i s_j \bmod q$$

find - secret $(s_1, s_2, \ldots, s_n)$

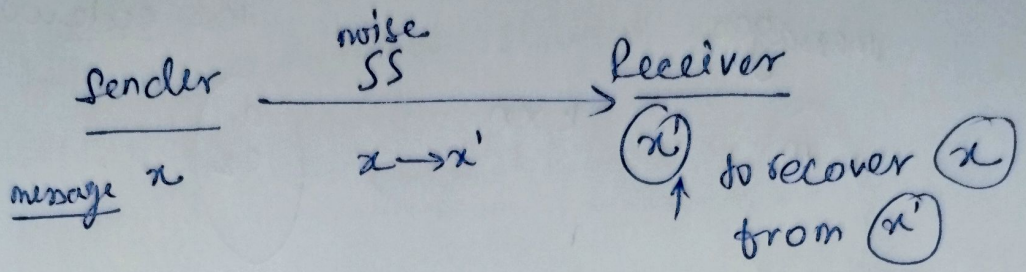
→ considered to be a difficult problem.

→ regarded as the problem of finding a solution modulo q to the approximate system of linear eqn.

→ cannot be solved with quantum computer.

Code Based Cryptography Algorithm :- 16/NOV/2024 7:50 am

Code here means → error correcting codes



we can use redundant code

for example :-

$5 \to (101)_2 \longrightarrow$ 111 000 111 → x

110 010 110 → x'

now we can recover the original message as

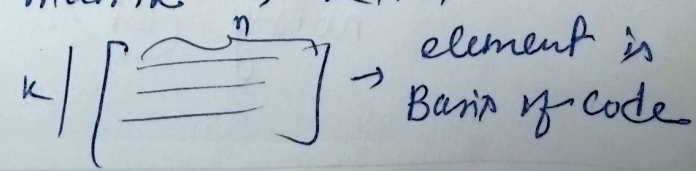
x → 111 000 111

linear Code :-

let $k, n \in Z$ (integer) , $k \leq n$

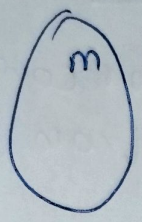
and let this linear code be a 'k' dimensional subspace of $(Z_2)^n$ → Binary string of length 'n'. where $(Z_2)^n$ is

$C \to [n, k]$, generating matrix → $k \times n$



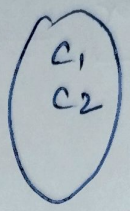
Now, suppose

space of the messages



$k \times n$

space of the codewords



The main idea here is take a message 'm' multiply it with generating matrix $k \times n$ to generate a codeword of length n

$m_i \longrightarrow c_i$

$m_j \longrightarrow c_j$

so there needs to be some distance (Hamming weight) b/w c_i & c_j as if there will be any noise in channel and

$00101 \rightarrow c_i$ becomes $00100 \rightarrow c_i'$

$00100 \rightarrow c_j$ then receiver can get confused if it is actual (c_j) or noisy (c_i').

$H.W(x)$ = no. of '1' in x.

here $H.W(c_i \oplus c_j) > 1$

but generally it should be

$$\boxed{H.W(c_i \oplus c_j) \geq d}$$

$[n, k, d]$ where d is the min. distance b/w any two valid codewords.

defining it

$(x, y) \in (Z_2)^n$

$$dist(x, y) = |\{i : 1 \leq i \leq n,\ x_i \neq y_i\}|$$

Orthogonal Codes :—

$x, y \in (Z_2)^n$

$x = \{x_1, x_2, x_3 \text{ —— } x_n\}$

$y = \{y_1, y_2, y_3 \text{ —— } y_n\}$

Consider a code space C,

$$C = \begin{Bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{Bmatrix}$$

dual code of C :-

Set of all the vectors that are orthogonal to all the vectors in C.

C → (G) is the generating matrix of C

& (H) is the generating matrix of the dual code of C

that also means,

(H) is the parity check matrix of the code C.

This basically means, rows of H matrix are linearly independant.

i.e. we can write,

Now, lets see how it works.

As there is no key involved, but still we need to recover x from x' using codeword.

Alice $\rightarrow x \rightarrow y = x \cdot G$

here, $len(x) = k$

so, $(1 \times k) \times (k \times n) \rightarrow (1 \times n)$

$len(y) = n$

suppose y becomes 'r' after noise.

$y \rightarrow (r)$ received

Case (i) r has no noise or error

i.e $(r = y)$

& $y \in C$ so, we can check & detect there is no noise.

$C = \{ c_1, c_2, c_3, \ldots$ with d between c_1 and c_2, and d between c_2 and c_3

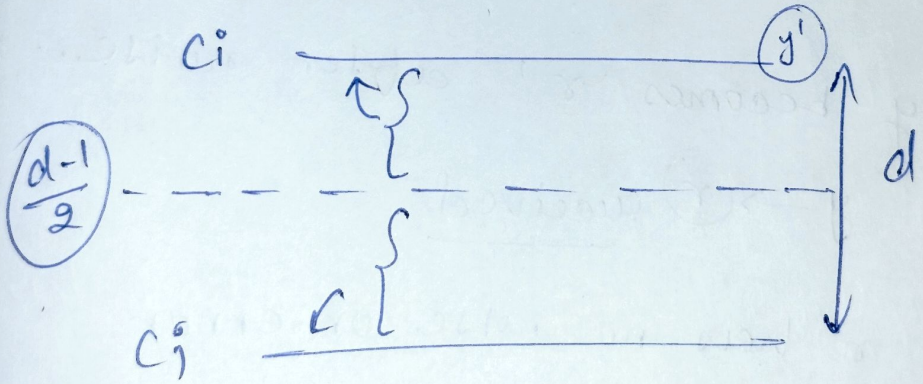
Case (ii) $r \neq y$. & still $y \in C_i, C = \begin{matrix} c_1 \\ c_2 \\ c_3 \\ ! \end{matrix}$ $\updownarrow d$ $\updownarrow d$

$$y = r + \underset{\text{Noise}}{N}(d')$$

$$d' \leq \frac{d-1}{2}$$

Here the main objective is we need to find the nearest neighbour of the received message r.

we denote it as $nn(r)$.



$nn(r) = y'$ → nearest neighbour

$x' \rightarrow y' = x' \cdot G$

$x' = x$

$len(x) = k$,

no. of possible codeword $= 2^k$

if the k value is large, it is very difficult to find the nearest neighbour. to solve this we use syndrome decoding method.

Syndrome decoding method :—

(H) → is a generating matrix of dual codeword of C [all code words that are orthogonal to actual code words]

if $x \in (\mathbb{Z}_2)^n$ is a valid codeword

then, we can write

$$H \cdot x^T = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Now, if we receiver 'r'

$$r = x + e \quad (\text{error})$$

$$H \cdot r^T = H \cdot (x+e)^T = H \cdot x^T + H \cdot e^T$$

$$= H \cdot e^T$$

here

length $(r) = n$

but

$$e \rightarrow \left(\frac{d-1}{2}\right)$$

error cannot exceed this $\left(\frac{d-1}{2}\right)$ value.

so, we can check for all possible error in $(H \cdot e^T)$, checking that will be much smaller, compared to checking for all possible codewords for all possible messages.

Here, the idea is very simple,

Suppose, we first consider

$$H \cdot r^T = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$ that means $(e = 0)$.

but if

$$H \cdot r^T \neq \begin{pmatrix} 0 \\ 0 \\ \vdots \\ b \end{pmatrix} \rightarrow e \neq 0$$

Now, we will check,

$(e \rightarrow 1)$ → consider hamming weight of (1

$$|wt(e)| = n$$

Then how many possible errors can be there of hamming weight 1 is given as (nC_1)

Ily, with hamming weight 2, (nC_2)

we can calculate all possible errors,

$$nC_0 + nC_1 + nC_2 + \text{---} + nC_{\frac{d-1}{2}} < 2^k$$

nC_0 ↑ no error

Classic Mceliece algo → Goppa code → (m, t)

linear code → n, k, d

in Goppa code,

$n = 2^m$

$d = 2t+1$

$k = n - mt$

earlier given in one conference.

$m = 10$, $t = 50$ → [1024, 524, 101]

→ considered not secure later so, gave new one.

$m = 11$, $t = 27$ → [2048, 1751, 55]

$k \times n$ → 524×1024 → earlier

1751×2048 → now

Now, lets define it,

$G \rightarrow n, k, d$, $n = 2^m$, $d = 2t+1$, $k = n - mt$.

$S \rightarrow k \times k$ → invertible matrix.

$P \rightarrow n \times n$ → permutation matrix

$$G' = S \cdot G \cdot P$$

key space

$K = \{ G, S, P, G' \}$

G' → public key

G, S, P → private key

Encryption :- (using public key)

$x \in (Z_2)^k$ $\quad len(x) = k$

$$\boxed{y = x \cdot G' + e}$$

$e \in (Z_2)^n$

$HW(e) = t$.

where $G' = S \cdot G \cdot P$.

Decryption :-

$y_1 = y \cdot P^{-1}$ → removed effect of P

$y_1 = x_1 + e_1$ → removed effect of e

where $x_1 \in C$ → removed effect of G

decode using error correcting code

find → $x_0 \in (Z_2)^k$ such that $x_0 G = x_1$

i.e. $x_1 \cdot G^{-1} = x_0$

$x = x_0 \cdot S^{-1}$ → removed effect of S.

example :-

$$G_{4\times 7} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

linear code (7, 4, 3) (n, k, d)

4 × 7 (k × n).

$$S_{k\times k \to 4\times 4} = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

$$P_{n\times n} = P_{7\times 7} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

notice that every row/column has only one '1' value that means hamming weight of 1. ⟶ permutation matrix.

$$G'_{k\times n} = S_{k\times k} \cdot G_{k\times n} \cdot P_{n\times n} = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

$k\times n$, 4×7

$x = (1,1,0,1)$, $e = (0,0,0,0,1,0,0)$

ciphertext,

$$y = x \cdot G' + e$$

$$= (0,1,1,0,1,1,0)$$

decryption :-

$$y_1 = y \cdot P^{-1} = (1,0,0,0,1,1,1)$$

$$y_1 = x_1 + e$$

we'll get

$$x_1 = (1,0,0,0,1,1,0)$$

$$x_0 \rightarrow (1,0,0,0)$$

$$x = x_0 \cdot S^{-1} = \underline{(1,1,0,1)}$$

Lamport Signature Scheme :- 16 Nov 2024

→ It is a one time signature scheme.

→ (public key, secret key)
↓ ↓
→ verification key signing key

→ Let message $m = (m_1, m_2, \ldots m_n)$ (binary string)
length of message, $|len(m)| = n$

→ So for this signature scheme, we require a strong hash function (a strong one way function)

→ Lets define the function,

$f(x) = \alpha^x \bmod p$, α is the primitive element modulo p.

$f : \{0, \ldots, p-2\} \to \mathbb{Z}_p^*$

example :-

$f(x) = 3^x \bmod 7079$, 3 is the primitive element in $\mathbb{Z}_{7079}^*$

We know that-

$$m = \{m_1, m_2, \ldots, m_n\}$$

$$m_i = m_j \;,\; j \in \{0,1\}$$

Case 1 :- $m_i = 0 \longrightarrow y_{i,0}$

Case 2 : $m_i = 1 \longrightarrow y_{i,1}$

→ preimage of public key $z_{i,0}$, $z_{i,1}$

example :-

$$f(x) = 3^x \bmod 7879$$

$$m = \{m_1, m_2, m_3\} \Rightarrow len(m) = 3$$

Now, we will choose 6 random no,

Not published (secret key)

$y_{1,0} = 5831$ | $z_{1,0} = f(y_{1,0}) = 2009$

$y_{1,1} = 735$ | $z_{1,1} = f(y_{1,1}) = 3010$

$y_{2,0} = 803$ | $z_{2,0} = f(y_{2,0}) = 4672$

$y_{2,1} = 2467$ | $z_{2,1} = f(y_{2,1}) = 4721$

$y_{3,0} = 4205$ | $z_{3,0} = f(y_{3,0}) = 260$

$y_{3,1} = 6449$ | $z_{3,1} = f(y_{3,1}) = 5831$

public key

Now, signing a message,

$x = \{1, 1, 0\}$

$sign(x) = \{y_{1,1}, y_{2,1}, y_{3,0}\}$

$= \{735, 2467, 4205\}$

Now, we are verifying,

Compute

$3^{735} \bmod 7079 = 3810 = z_{1,1}$ (already published)

if the signature is valid

it will be equal to $z_{1,1}$ which is already published.

Ily Compute

$3^{2467} \bmod 7079 = z_{2,1} = 4721$ → match

& $3^{4205} \bmod 7079 = z_{3,0} = 2800$ → match

Signature Guarantee :-

$$\underbrace{(x_1 \dots x_k)}_{m}, \underbrace{(y_{1,x_1}, \dots y_{k,x_k})}_{\text{signature}}$$

Here attacker wants to sign a new message but do not have access to $[y_{1,0}, y_{1,1}, y_{2,0}, y_{2,1}, \dots y_{k,0}, y_{k,1}]$.

Suppose :-

it is known $1001 \xrightarrow{m} (y_{1,1}, y_{2,0}, y_{3,0}, y_{4,1})$

(not (suppose known signature))

$1000 \longrightarrow (y_{1,1}, y_{2,0}, y_{3,0}, y_{4,0})$

But don't know this.

So, hence the attacker cannot sign a new message that is why we say it is a one time signature scheme.

→ secure, if used only one time.

Signed two message with same signature :—

lets say Alice has signed two messages,

$$m \quad \text{(signature)}$$
$$(1001) \rightarrow (y_{1,1}, y_{2,0}, y_{3,0}, y_{4,1})$$
$$(1010) \rightarrow (y_{1,1}, y_{2,0}, y_{3,1}, y_{4,0})$$

attacker can construct the signature through above.

$$(1011) \rightarrow (y_{1,1}, y_{2,0}, y_{3,1}, y_{4,1})$$

that means authenticity is breached hence, it means we cannot sign two messages with ~~know~~ the same secret key.

Issue of lamport signature scheme :—

① It is a one-time signature scheme, hence we cannot sign two messages with the same secret key. As it is vulnerable to do so, anybody can forge the signature.

(2) if message length,

$len(x) = l$ then the secret key will have $(2 \cdot l)$ no. of possibilities $(y_{i,0} \& y_{i,1})$

suppose we are signing a message of

length $2^{l} \longrightarrow (4^{48})^{2l}$

then $y_{i,0} \longrightarrow 224$

$y_{i,1}$

448

total no. of bits to create signature

$= 448 \times 224$

that's why this method is not useful in terms of creating a efficient signature with manageable no. of bits in secret key (size).

(3) So, Winternitz signature scheme is a more efficient version.

Winternitz Signature scheme :—

ω bits , f , (w = 3) → fixing for simplicity.

sign together at a time.

f is a one way function.

Lets start with one random value y_0

then make a hash chain,

$$y_0 \xrightarrow{f} y^1 \xrightarrow{f} y^2 \xrightarrow{f} y^3 \xrightarrow{f} y^4 \xrightarrow{f} y^5 \xrightarrow{f}$$

$$y^6 \xrightarrow{f} y^7 \xrightarrow{f} z$$

as $w = 3 \longrightarrow 2^3 \rightarrow$ 8 possible messages

0 0 0
0 0 1
0 1 0
⋮ ⋮ ⋮
1 1 1

} 8 messages

hence, we can write,

$$y^j = f(y^{j-1}) \quad 1 \le j \le 7$$

$$z = f(y^7)$$

we can also write, $y^j = f^j(y_0)$

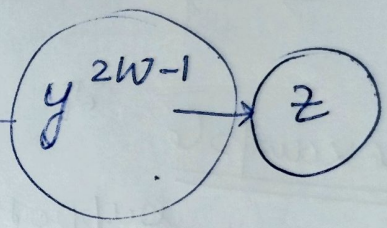
Writing again for clarity,

$$y^j = f(y^{j-1}) \quad \text{where } 1 \le j \le 7, \; z = f(y^7)$$

$$y^j = f^j(y_0) \quad \text{where } 1 \le j \le 7, \; z = f^0(y_0)$$

Publish z → public key

So, in general the hash chain would consist of $2^w + 1$, $y_0^0 \rightarrow y^1 \rightarrow \cdots \rightarrow y^{2^w-1} \rightarrow z$

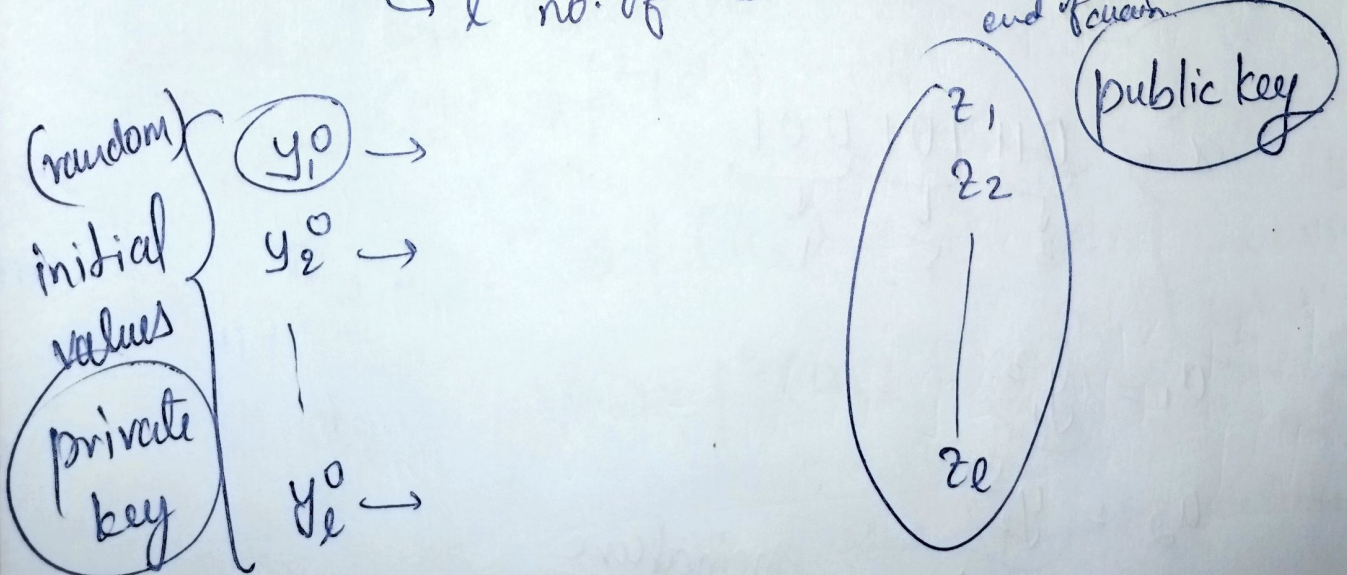


Suppose

k bit message

$\ell = k/w$

↳ ℓ no. of hash chain.



Signing the message :—

$x \rightarrow (x_1, x_2, \ldots x_\ell)$

$x_i \rightarrow w$ bit block

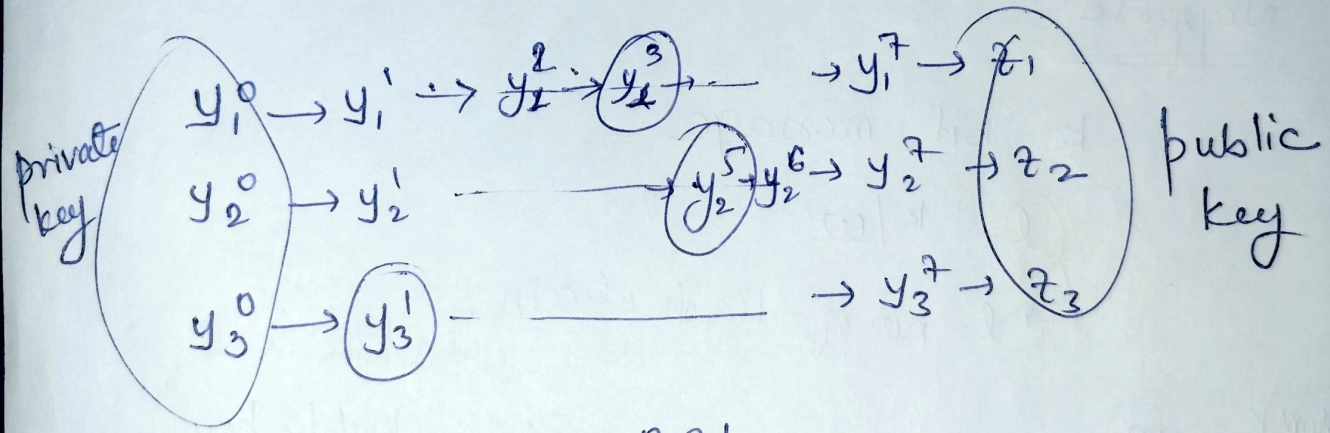
$$a_i = y_i^x = f^{x_i}(y_i)$$

Verification

↳ $f^{2^w - x_i}(a_i) = z_i$ or not

example

suppose $k = 9$, $w = 3$, $\ell = 3 = \frac{k}{w}$



$x = \underbrace{011}_{3}\ \underbrace{101}_{5}\ \underbrace{001}_{1}$

$a_1 = y_1^3$

$a_2 = y_2^5$

$a_3 = y_3^1$

signature

for verification we need to check

$$f^5(a_1) = z_1$$

$$f^3(a_2) = z_2$$

$$f^1(a_3) = z_3$$

if they are equal, signature is valid.

→ trying to copy the signature or retrieve it to sign a new message

for example :-

$$x = \underset{3}{011}\,\underset{5}{101}\,\underset{1}{001}$$

→ $a_1 = y_1^3$, $a_2 = y_2^5$, $a_3 = y_3^1$

attacker can calculate

$a_1 = y_1^3 \Rightarrow f^2(a_1) = y_1^5$

$a_2 = y_2^5 \Rightarrow f(a_2) = y_1^6$

$a_3 = y_3^1 \Rightarrow f^3(a_3) = y_1^4$

signature of message 101 110 100 x'.

without knowing the secret values, we can forge the signature to create valid sign.

So, we need to add some additional security in this, so we'll add checksum.

Checksum :—

$$c = \sum_{i=1}^{l} (2^w - 1 - x_i)$$

$w = 3$ $\quad l = \frac{9}{3} = 3$

$2^w = 8$.

$x = \underline{011}\ \underline{101}\ 001$

$$c = (7-3) + (7-5) + (7-1)$$

$$= 4 + 2 + 6 = (12).$$

$12 \rightarrow \underline{001}\ \underline{100}$ [added '0' to get it in group of 3 bits].

$c = \underbrace{\underline{001}}_{x_4}\ \underbrace{\underline{100}}_{x_5}$ $\quad x_3 \quad x_2 \quad x_1$

$a_4 = y_4' = f(y_4)$

$a_5 = y_5^4 = f^4(y_5)$

} adding two additional values & got their sign as well.

$[a_4, a_5, a_3, a_2, a_1]$ → signature

New chain

$y_4^0 \rightarrow (y_4^1)$ — — — —

$y_5^0 \rightarrow y_5^1 \rightarrow$ — — (y_5^4) — —

Verification →

(1) verify a_1, a_2, a_3

(2) compute checksum

(3) verify a_4, a_5.

check if it secure or sign can be forged ?

example →

$(x_1, x_2, x_3) \rightarrow (a_1, a_2, a_3, a_4, a_5)$

$(x_1', x_2', x_3') \rightarrow$

→ moving backward is difficult such means that attacker is trying find reverse of hash funct^n.

→ Attack can only happen if :—

(i) $x_i' \geq x_i$ → if true

but $C' < C$ → this means attacker cannot compute (c')

now check

↑ then this should be also true.

As we know,

$$C = \sum_{i=1}^{l} (2^w - 1 - x_i)$$

then,

$$C' = \sum_{i=1}^{l} (2^w - 1 - x_i')$$

then putting $C > C'$

we get as we know $x_i \leq x_i'$

then

$$\sum_{i=1}^{l} (2^w - 1 - x_i) > \sum_{i=1}^{l} (2^w - 1 - x_i')$$

Now we can claim $C > C'$.

it is secure scheme even when it is a one-time signature algo.