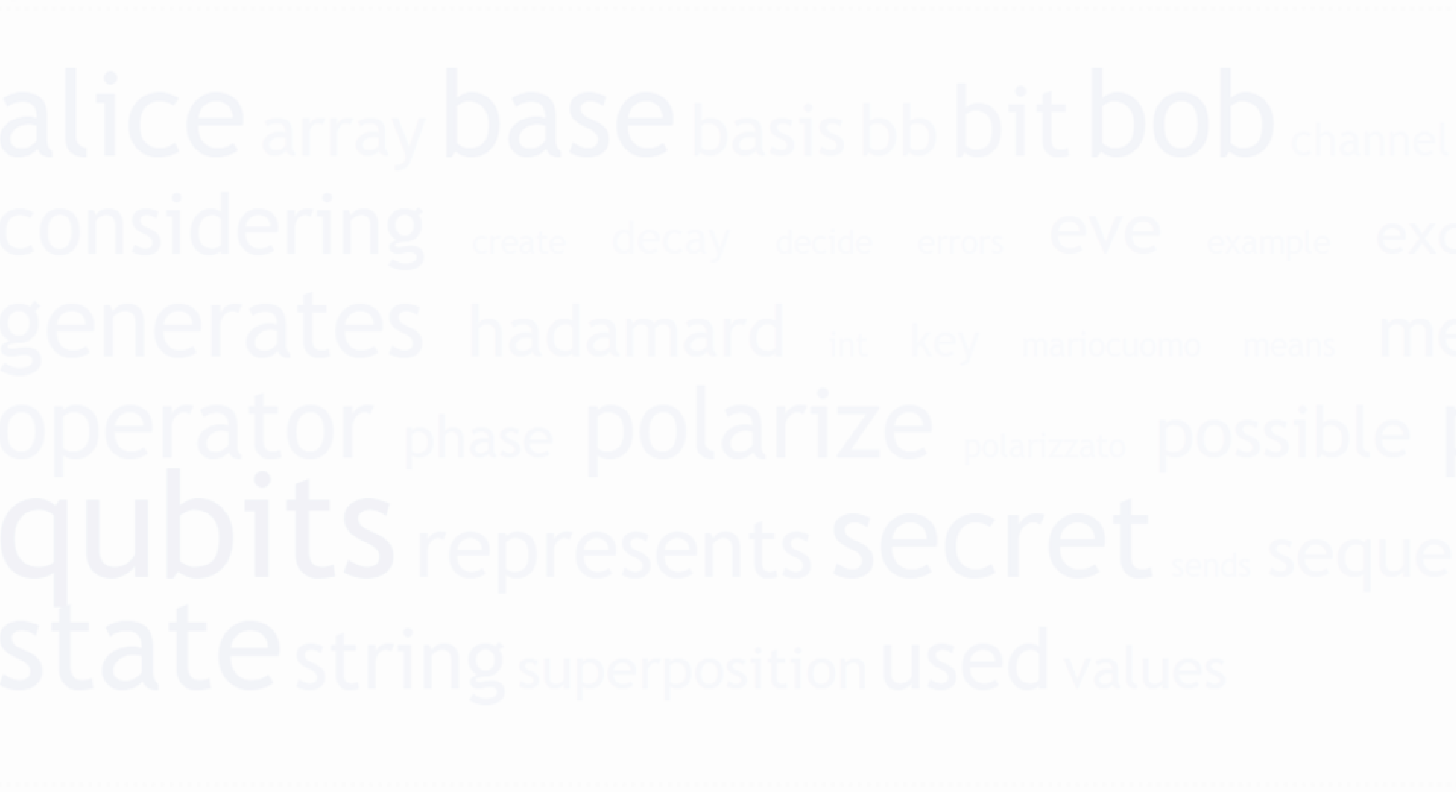
BB84 PROTOCOL – Q# Holiday Calendar 2022

MICROSOFT

MARIO CUOMO
CLOUD SOLUTION ARCHITECT



Summary

ABSTRACT

If on one hand Quantum Computing creates a strong challenge to the Church-Turing thesis, showing how it is possible to use the new computing model to efficiently solve problems with complexity np, on the other hand it offers just as many certainties.
One example is $\mathrm{BB84}$ protocol – a quantum protocol for distributing shared secrets in a channel that may be insecure.

A shared secret is a method used in cryptography to make communication secure. This term is often part of symmetric cryptography: the situation in which the two communicating parties use the same key to encrypt and decrypt the messages they exchange. The shared secret can be the encryption key or a seed secret from which to generate the actual key.

This short article describes in a simple – and in some cases simplistic – way the quantum computing model providing the basic tools to fully understand how the protocol works.

INTRODUCTION TO THE QUANTUM COMPUTATIONAL MODEL

This short post does not aim to explain how it is possible to manipulate quantum particles but rather explains how it is possible to use their physical characteristics creating a new computing model.

The elementary unit of calculation of this model is the qubit – an element whose state is between the two base states $|0\rangle$ or $|1\rangle$ and it is not possible to know with certainty its value until a measurement is made on it. This is called the decay of the qubit into a base state.
In general the state of a qubit is indicated as follows

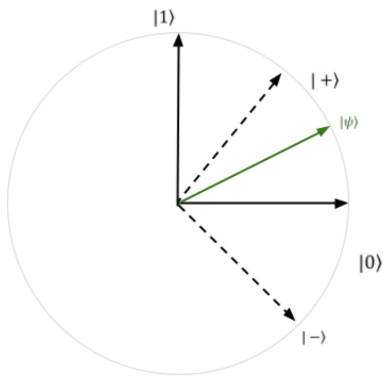
$$|\psi\rangle = a_0|0\rangle + a_1|1\rangle$$

The amplitude values a_0 and a_1 are complex numbers and represent the probability that measured the qubit decays respectively in the base state $|0\rangle$ or $|1\rangle$. The probability that by measuring $|\psi\rangle$ it decays into the state $|0\rangle$ is $|\alpha_0|^2$.

The one presented takes the name of ket-notation introduced by Paul Dirac.
In reality we can also represent qubits in matrix form with the following notation.

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle = \alpha_0\begin{pmatrix}1\\0\end{pmatrix} + \alpha_1\begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}\alpha_0\\\alpha_1\end{pmatrix}$$

For simplicity, the introduction of qubits took place with respect to the computational basis – that is, the one composed of versors $|0\rangle$ and$|1\rangle$. It is important to note that a qubit exists regardless of the base against which it is representing. In fact, it is possible to represent qubits in any other base, that is, with respect to any pair of versors orthogonal to each other.
Another basis is the Hadamard base composed of versors $|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ and $|-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$.



It is possible to manipulate qubits through different operators. The effect of an operator is none other than the rotation of the qubit in space – which is called Hilbert space.
The only constraint that we have is that the operators must be unitary transformation, that is, the application in succession of the operator U and $U^\dagger$ to the qubit $|\psi\rangle$ they must return $|\psi\rangle$ (or, in other words, the product $UU^\dagger$ must be the identity I).
An operator U is represented by a matrix and is $U^\dagger$ the transpose of the complex conjugate (Hermitian transpose).

The best known operators are Pauli operators – bit flip, phase flip, operator Y and identity I.
Considering a qubit$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ we have that

	U	U$\vert\psi\rangle$
bit flip (X)	$\begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}$	$\alpha_1\vert 0\rangle + \alpha_0\vert 1\rangle$
phase flip (Z)	$\begin{pmatrix}1 & 0\\0 & -1\end{pmatrix}$	$\alpha_0\vert 0\rangle - \alpha_1\vert 1\rangle$
operator Y	$\begin{pmatrix}0 & 1\\-1 & 0\end{pmatrix}$	$\alpha_1\vert 0\rangle - \alpha_0\vert 1\rangle$
identity operator (I)	$\begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}$	$\alpha_0\vert 0\rangle + \alpha_1\vert 1\rangle$

It is also possible to consider multiple qubits together.
When two or more qubits come into contact with each other, they must no longer be considered as individuals but as a single entity influencing each other. It is the principle of entanglement.
In general, the state of two qubits is described in the following formalism

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

An entangled state used is the Bell state and $|\Phi^+\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$ shows this principle with simplicity: the overall state of the system is balanced between the base states $|00\rangle$ and $|11\rangle$. If we measure the first qubit and – for example – we get $|1\rangle$ then inevitably the second qubit will also be in the state $|1\rangle$. It is as if the second qubit has moved from an equilibrium state to a base state as a result of the measurement and decay of the first.

In summary, two qubits can be individually each in a state $\alpha_0|0\rangle + \alpha_1|1\rangle$ and $\beta_0|0\rangle + \beta_1|1\rangle$ when they are together their state is described by $\alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$.
The operator used is the tensor product ⊗.

$$\begin{pmatrix}\alpha_0\\\alpha_1\end{pmatrix} \otimes \begin{pmatrix}\beta_0\\\beta_1\end{pmatrix} = \begin{pmatrix}\alpha_0\begin{pmatrix}\beta_0\\\beta_1\end{pmatrix}\\ \alpha_1\begin{pmatrix}\beta_0\\\beta_1\end{pmatrix}\end{pmatrix} = \begin{pmatrix}\alpha_0\beta_0\\\alpha_0\beta_1\\\alpha_1\beta_0\\\alpha_1\beta_1\end{pmatrix} = \alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + \alpha_1\beta_0|10\rangle + \alpha_1\beta_1|11\rangle$$

It is possible to apply different operators to two qubits – always with the constraint of being unitary transformations.
The most commonly used operators are the ControlledNOT and Hadamard. The first is the equivalent of the CNOT on classical bits, and the second is useful for placing a decayed qubit on a base state in balanced superposition.
Considering $|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$

	U
CNOT	$\begin{pmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 0 & 1\\0 & 0 & 1 & 0\end{pmatrix}$
Hadamard(H_2)	$\frac{1}{\sqrt{2}}\begin{pmatrix}H_1 & H_1\\H_1 & -H_1\end{pmatrix}$

with

$$H_1 = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\1 & -1\end{pmatrix}$$

BB84 PROTOCOL

BB84 protocol is used for generating a cryptographic secret shared between two parties – which we call Alice and Bob for simplicity. It is important to specify that the shared secret is generated randomly and not decided a priori by one of the two parties: this is an additional degree of protection as it is not necessary to use a specific algorithm to generate it.
The protocol is proposed by Charles Bennet and Gilles Brassard in 1984.

The protocol consists mainly of 3 phases:

1. Alice generates a superposition of qubits from two randomly chosen sequences and sends it to Bob
2. Bob generates a random sequence and uses it to polarize the superposition of qubits.
3. Alice and Bob confront each other to get the shared secret

To these two phases are added two others that ensure robustness against the noise of the channel and a hypothetical malicious part – called Eve.

Let's start analyzing the phase 1.
Alice has n qubits in the base states and randomly generates two sequences of bits α and β – both of length n.
The sequence α is a simple sequence of bits that represents a superposition of qubits in base states: $\alpha_i = 0$ means to have a qubit in the base state $|0\rangle$, $\alpha_i = 1$ means to have a qubit in the base state $|1\rangle$.
The sequence β indicates how to polarize the qubits: $\beta_i = 0$ means that the qubit i is polarized with the computational base, $\beta_i = 1$ means that the qubit i is polarized with the Hadamard base.

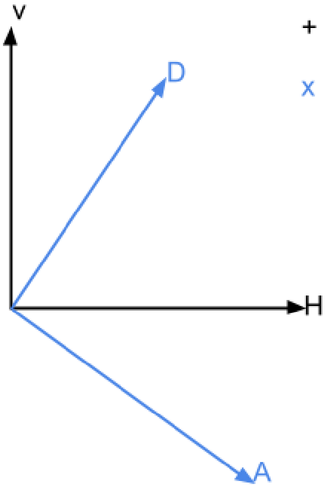
The overall state of n qubits is the tensor product of qubits $|\psi_{\alpha_i\beta_i}\rangle$.
Alice transmits the following superposition

$$|\psi\rangle =$$

To have a better readability of the superposition we assign names to the various forms that qubits can take.

$\lvert\psi_{\alpha_i\beta_i}\rangle$	Symbolic name
$\lvert\psi_{00}\rangle = \lvert 0\rangle$	Vertical (V)
$\lvert\psi_{10}\rangle = \lvert 1\rangle$	Horizontal (H)
$\lvert\psi_{01}\rangle = \lvert +\rangle = \frac{1}{\sqrt{2}}\lvert 0\rangle + \frac{1}{\sqrt{2}}\lvert 1\rangle$	Diagonal (D)
$\lvert\psi_{11}\rangle = \lvert -\rangle = \frac{1}{\sqrt{2}}\lvert 0\rangle - \frac{1}{\sqrt{2}}\lvert 1\rangle$	Anti – Diagonal (A)

What explain the table is very simple.
When we polarize state $|0\rangle$ considering computational basis we get state V.
When we polarize state $|1\rangle$ considering computational basis we get state H.
When we polarize state $|0\rangle$ considering Hadamard base we get state D.
When we polarize state $|1\rangle$ considering Hadamard base we get state A.
In the literature it is not difficult to find the following nomenclature H(→), V(↑), D(↗), A(↘), operational basis (+) and Hadamard basis(×).



Polarizing a qubit considering another base is a very simple operation.
Consider the qubit $|1\rangle$ in the computational basis. To express it in the Hadamard basis, simply apply the Hadamard operator to it.

$$H_1|1\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}0\\ 1\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\ -1\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\ 0\end{pmatrix} - \frac{1}{\sqrt{2}}\begin{pmatrix}0\\ 1\end{pmatrix} = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle = |-\rangle$$

Let's assume Alice generated the following strings.

$$\alpha = 0111010010100101$$
$$\beta = 0110010100010111$$

Alice polarizes her qubits as follows

α	0	1	1	1	0	1	0	0	1	0	1	0	0	1	0	1
β	+	×	×	+	+	×	+	×	+	+	+	×	+	×	×	×
$\psi_{\alpha\beta}$	↑	↘	↘	→	↑	↘	↑	↗	→	↑	→	↗	↑	↘	↗	↘

Alice then sends the qubits $|\psi\rangle$ to Bob. She can do this using a quantum channel or using the teleportation protocol.

Phase 2 begins.
Bob generates a random sequence $\beta'$of n bits. He tries to guess what are the encoding bases used by Alice and in this way he can retrieve the message α.
Statistically, Bob guesses 50% of the time.
Let's assume Bob generated the string

$$\beta' = 0101010010100100$$

Bob polarizes $\psi_{\alpha\beta}$ using β'

α	0	1	1	1	0	1	0	0	1	0	1	0	0	1	0	1
β	+	×	×	+	+	×	+	×	+	+	+	×	+	×	×	×
$\psi_{\alpha\beta}$	↑	↘	↘	→	↑	↘	↑	↗	→	↑	→	↗	↑	↘	↗	↘
β'	+	×	+	×	+	×	+	+	×	+	×	+	+	×	+	+
α' polarizzato	↑	↘	→	↗	↑	↘	↑	→	↗	↑	↗	→	↑	↘	→	→
α'se misurato	0	1	1	0	0	1	0	1	0	0	0	1	0	1	1	1

Inevitably there are errors in Bob's measurements due to not knowing the string β.

In the phase 3 Bob and Alice exchange on a classical channel the encoding bases used and discard the qubits for which they do not match. The values α_i and α'_i where β_i and β'_i matching are the shared secret. What remains is a secret shared between the two!

β	+	×	×	+	+	×	+	×	+	+	+	×	+	×	×	×
β'	+	×	+	×	+	×	+	+	×	+	×	+	+	×	+	+
shared secret	0	1			0	1	0			0			0	1		

If the length of the shared secret is less than $\mathrm{n}/2$ it is recommended to start again with the execution of the protocol.

What happens if there is a malicious user on the broadcast channel? Thanks to No cloning theorem, Eve cannot have a copy of the qubits sent by Alice to Bob.
The only thing she can do is an active man in the middle at the step2: Eve acquires qubits from Alice, polarizes them and sends them to Bob.
Eve is in the same situation as Bob because she has to guess the choices and Alice. It does so with a statistical success of 50% . When Eve makes measurements of qubits, she perturbs their state and indirectly leaving traces of her presence. Let's see why.

Let's imagine that Eve impersonated Bob and receives $|\psi\rangle$ from Alice.
Eve generates a string β'' of n bits that it uses to polarize the message.

$$\beta'' = 1010001100100010$$

α	0	1	1	1	0	1	0	0	1	0	1	0	0	1	0	1
β	+	×	×	+	+	×	+	×	+	+	+	×	+	×	×	×
$\psi_{\alpha\beta}$	↑	↘	↘	→	↑	↘	↑	↗	→	↑	→	↗	↑	↘	↗	↘
β''	×	+	×	+	+	+	×	×	+	+	×	+	+	+	×	+
α''polarizzato	↗	→	↘	→	↑	→	↗	↗	→	↑	↗	→	↑	→	↗	→

Eve sends $\alpha''$to Bob who makes his measurements with the string β'.

α	0	1	1	1	0	1	0	0	1	0	1	0	0	1	0	1
β	+	×	×	+	+	×	+	×	+	+	+	×	+	×	×	×
$\psi_{\alpha\beta}$	↑	↘	↘	→	↑	↘	↑	↗	→	↑	→	↗	↑	↘	↗	↘
β''	×	+	×	+	+	+	×	×	+	+	×	+	+	+	×	+
α''polarizzato	↗	→	↘	→	↑	→	↗	↗	→	↑	↗	→	↑	→	↗	→
β'	+	×	+	×	+	×	+	+	×	+	×	+	+	×	+	+
α'polarizzato	→	↗	→	↗	↑	↗	→	→	↗	↑	↗	→	↑	↗	→	→
α'se misurato	1	0	1	0	0	0	1	1	0	0	$\underline{0}$	1	0	0	1	1

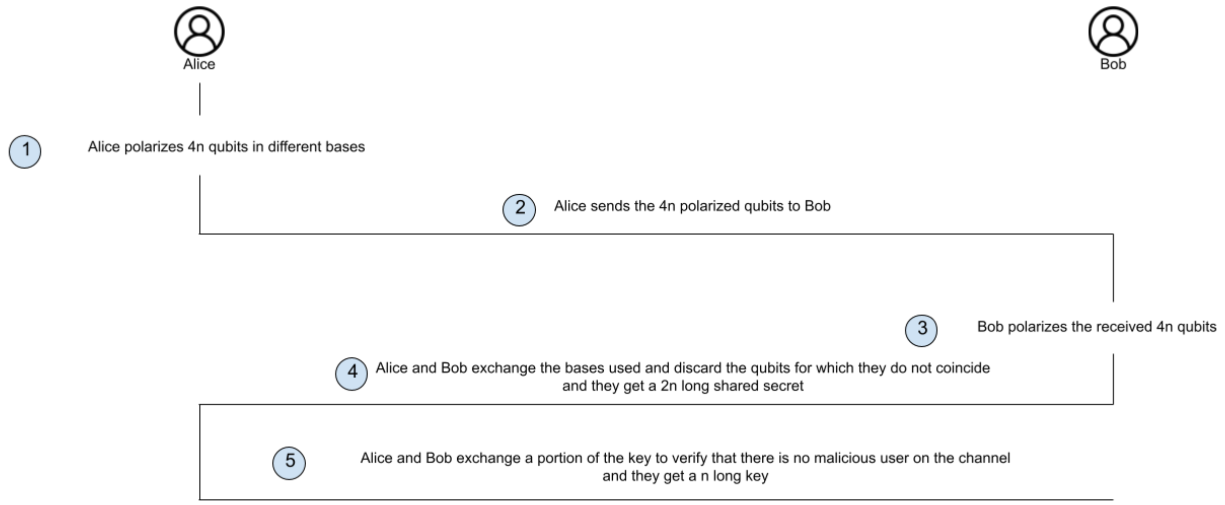
Alice and Bob correctly exchange the bases used to acquire the shared secret.

β	+	×	×	+	+	×	+	×	+	+	+	×	+	×	×	×
β'	+	×	+	×	+	×	+	+	×	+	×	+	+	×	+	+
shared secret per Alice	0	1			0	1	0			0			0	1		
shared secret per Bob	1	0			0	0	1			0			0	0		

As you can see the two secret do not match but Alice and Bob still do not know it.
They decide to exchange $\mathrm{n}/4$ bits – for example the first ones – of the shared secret for a check.

Statistically, $\mathrm{bit\ error\ rate}$ is the same between untransmitted and transmitted sequences. If the errors present are fewer than $\mathrm{n}/8$, Alice and Bob can decide to use the remaining $\mathrm{n}/4$ bits as a shared secret. At this point the errors on the shared secret can be mitigated using the Privacy Amplification technique – a method that allows Alice and Bob to generate a completely secret key knowing that Eve has partial information about the starting string.

To generate a shared bit secret, n you must then use $4\mathrm{n}$ qubits.
Below a summary about the protocol.


Alice
Bob
1
Alice polarizes 4n qubits in different bases
2
Alice sends the 4n polarized qubits to Bob
3
Bob polarizes the received 4n qubits
4
Alice and Bob exchange the bases used and discard the qubits for which they do not coincide and they get a 2n long shared secret
5
Alice and Bob exchange a portion of the key to verify that there is no malicious user on the channel and they get a n long key

IMPLEMENTATION IN Q

You can use a quantum simulator to test applications that use qubits locally, and Microsoft provides one in the Microsoft Quantum Developer Kit.
A quantum simulator is nothing more than a software running on a traditional computer but that allows you to simulate the behavior of qubits.

Below is a python script that interacts with a .qs.
Two functions have been implemented – BB84WithoutEve and BB84WithEve – that return the shared secret perceived by Bob. As the name suggests, only in the first case the secret coincides with one possessed by Alice.

As an example, consider the BB84WithoutEve function.

```
operation BB84WithoutEve(a : Int[], b : Int[], c : Int[], d : Int[]): String
```

a	It is a 16-bit array. It represents α Alice's string.
b	It is a 16-bit array. It represents β Alice's string.
c	It is a 16-bit array. It represents β' Bob's string.
d	It is a 16-bit array. It represents the moment when Alice and Bob exchange keys. d[i]==1 if b[i]==c[i]

First, you create an array containing 16 qubits in the base states $|0\rangle$ and $|1\rangle$ according to the values in array a.

```
use qubits = Qubit[16];
for i in 0 .. 15 {
    let bit = a[i];
    if bit==1 {
        X(qubits[i]);
    }
}
```

Initially the 16 qubits are all in the state $|0\rangle$. With the operator X you can transform them into $|1\rangle$.

Alice then polarizes the qubits according to the values in array b.

```
for i in 0 .. 15 {
   let _base = b[i];
   if _base == 1 {
       H(qubits[i]);
   }
}
```

Bob does the same according to the values in the c array.

```
for i in 0 .. 15 {
   let _base = c[i];
   if _base == 1 {
       H(qubits[i]);
   }
}
```

Alice and Bob exchange their basics and keep only the values such that d[i]==1.

```
mutable shared_secret = "";
for i in 0 .. 15 {
  let common_base = d[i];
  if common_base == 1 {
     let misure = M(qubits[i]);
     mutable x = "0";
     if misure == One {
       set x = "1";
     }
     set shared_secret = shared_secret + x;
  }
}
```

Before concluding the function, it is a good practice to clean the states of the qubits.

```
for i in 0 .. 15 {
   Reset(qubits[i]);
}
```

RESOURCES

My interest in Quantum Computing was born thanks to the lecture in Next Generation Computing Models course at Roma Tre University.
Below some my blog post

- *https://github.com/mariocuomo/Microsoft-Q-Advent-Calendar-2021*
  (No-Cloning Theorem)
- *https://medium.com/@mariocuomo/teletrasporto-non-solo-fantascienza-f4663b5a1c3a*
  (Teleportation protocol)
- *https://medium.com/@mariocuomo/superdense-coding-due-bit-al-prezzo-di-un-qubit-a341457352d5*
  (Superdense coding protocol)

The code proposed is available at *https://github.com/mariocuomo/Microsoft-Q-Holiday-Calendar-2022*

For any clarification, correction or doubt `cuomomario@hotmail.com`