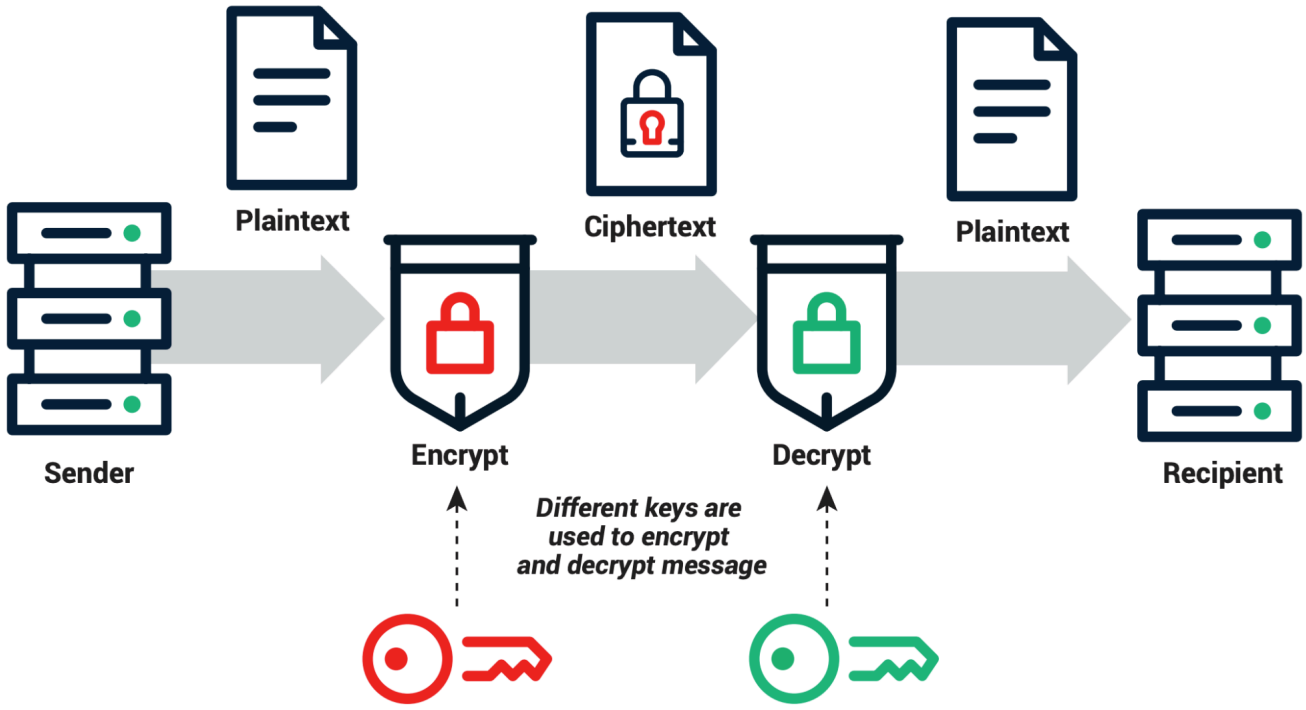
# QKD BB84 PROTOCOLS

## Private Key Method:

- The private key in this algorithm is the integer used to shift the characters during encryption and decryption.
- The key is kept secret between the communicating parties and is essential for decrypting the message.
- To decrypt the message successfully, the recipient needs to know the exact key used by the sender during encryption.

This code implements a basic Caesar cipher encryption and decryption algorithm. Here's how it works:



# Encryption (encrypt function)

- Take the input text and a key (an integer representing the number of positions to shift each character).
- Initialize an empty string for the encrypted text.
- For each character in the input text:
  - Shift the character by the key value.
  - Ensure that the shifted character remains within the range of alphabets (both lowercase and uppercase).
  - Append the shifted character to the encrypted text.
- Return the encrypted text.

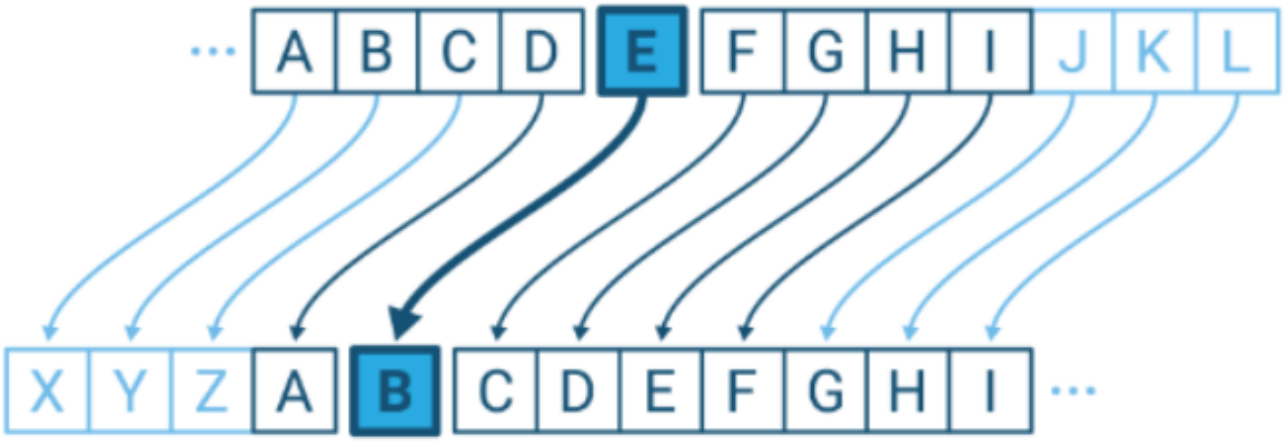



Fig: Visual Representation of how cipher encryption works

# Decryption (decrypt function)

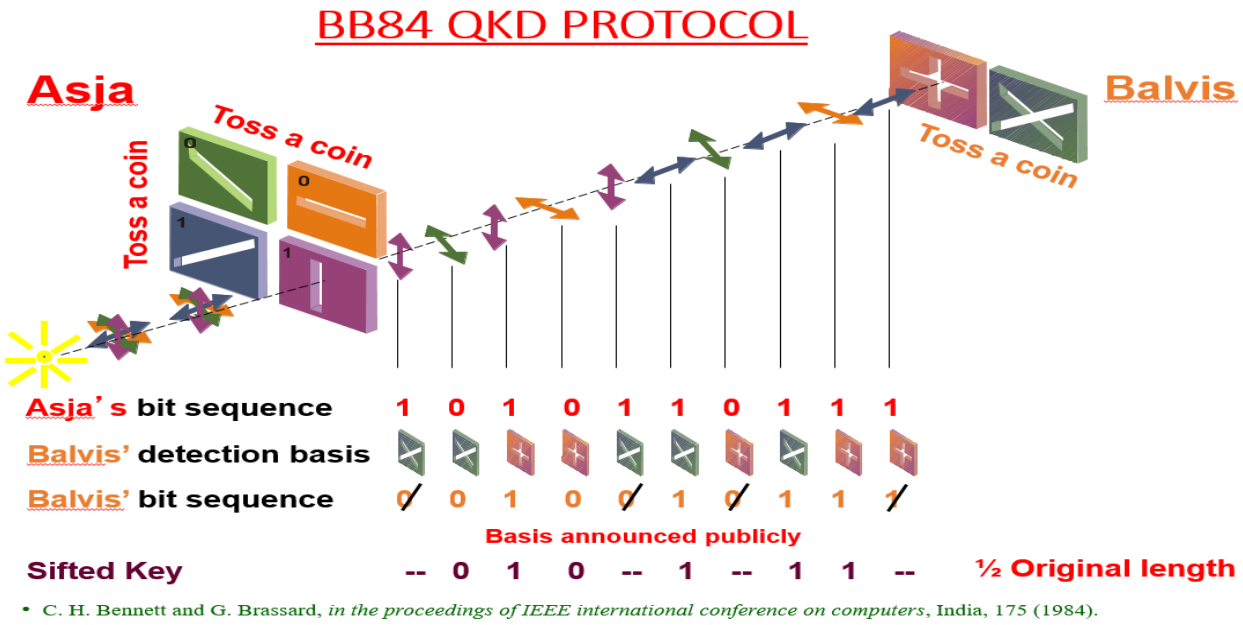
- The decryption function is essentially the same as the encryption function. Ensuring that the characters are shifted in the opposite direction during decryption.

**Types of QKD Protocols [1]**

1. **BB84 and variants (Based on Heisenberg's Uncertainty Principle)** A single-photon pulse is passed through a polarizer. Asja can use a particular polarizer to polarize a single-photon pulse and encode binary value bits to the outcome of a particular type (vertical, horizontal, circular, etc) of a polarizer. On receiving the photon beam, Balvis would guess the polarizer, and he can thus match the cases with Asja and know the correctness of his guesses. If Espian would have been trying to decode, then due to polarization by Espian's polarizer, there would be discrepancies in match cases of Asja & Balvis and thus they would know about eavesdropping. Examples are BB84, B92, SARG04.

2. **E91 and variants (Based on Quantum Entanglement)** There is a single source that emits a pair of entangled photons with Asja & Balvis receiving each particle. Similar to the BB84 scheme, Asja & Balvis would exchange encoded bits and match cases for each photon transferred. But in this scenario, the outcome of the results of the match cases of Asja & Balvis will be the opposite as a consequence of the Entanglement principle. Either of them will have complement bits in bit strings interpreted. One of them can then invert bits to agree upon a key. Since Bell's Inequality should not hold for entangled particles, thus this test can confirm the absence of eavesdroppers. Since practically it is not possible to have a third photon in entanglement with energy levels sufficient for nondetect ability, thus this system is fully secure.

## BB84 Protocols




## Introduction to BB84 Protocol

BB84, is the first in the history of QKD protocols, published in 1984 by Charles H. Bennett and Gilles Brassard. The idea introduced by BB84 has been inherited by many other QKD protocols.

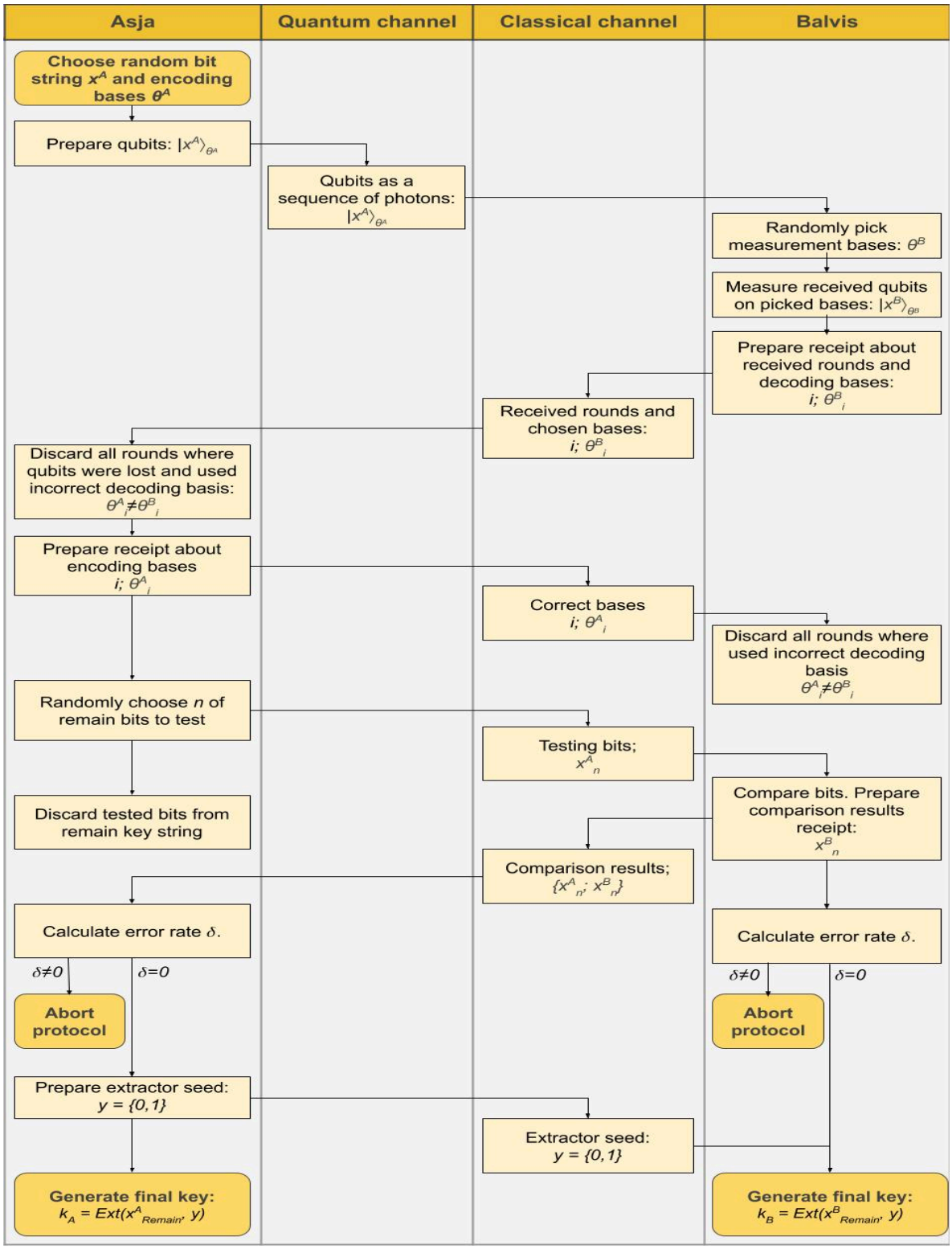
The protocol is secure, relying on two conditions:

1. Information gain is only possible at the expense of disturbing the states (if the two states are not orthogonal)
2. An authenticated public classical channel should exist.

BB84 is a method of securely communicating a private key from one party to another for use in one-time pad encryption.

Thus, we will review BB84 in detail, create its program step-by-step and verify its security.

# The BB84 Protocol Goes as:




Source: QKD Notebook QWorld

# EveEyes Attack

## 1. `interceptAndSend(circuit)`:

- **What happens**: This function simulates an eavesdropper intercepting qubits being transmitted in a quantum communication channel.
- **Details**:
  - It measures the qubits in the circuit and stores the measurement results.
  - These measurement results represent the intercepted information.
  - After obtaining the measurement results, the eavesdropper encodes the qubits based on their measurement.

## 2. `entangleQubit(circuit)`:

- **What happens**: This function entangles additional qubits controlled by the original qubits in the quantum circuit.
- **Details**:
  - It creates a set of qubits for the eavesdropper.
  - For each qubit in the original set, it entangles it with a corresponding qubit from the eavesdropper's set.

## Summary:

- `interceptAndSend(circuit)` allows an eavesdropper to intercept qubits by measuring them and then encoding them based on the measurement results.
- `entangleQubit(circuit)` lets an eavesdropper entangle their own qubits with the transmitted qubits.

Detail implementation at :- Code