



Quantum Monkeys

*Jorge Giménez, Miguel Hernández, Andreu Moreno y Rubén Piles*

# Protocolo BB84

Modelo de *testeo* de sistemas QKD

QisFall  
kitFest

# Protocolo BB84

## 01 Introducción

- ¿Por qué el problema es relevante?
- ¿Qué interrogantes plantea?

## 02 Casos de estudio

- Caso ideal
- Caso realista: fuente de ruido
- Soluciones

## 03 Ideas a futuro

- ¿Hasta dónde llega el planteamiento?
- ¿Cómo mejorarlo?

# ¿Inconvenientes?

- Coste computacional (seguridad aumenta con el número de qbits)
- Falta de estandarización
- Vulnerabilidades (MiTM, Side-Channel Attacks, Supply Chain Attacks...)
- Coste económico elevado (vs. alternativas clásicas)

# ¿Inconvenientes?

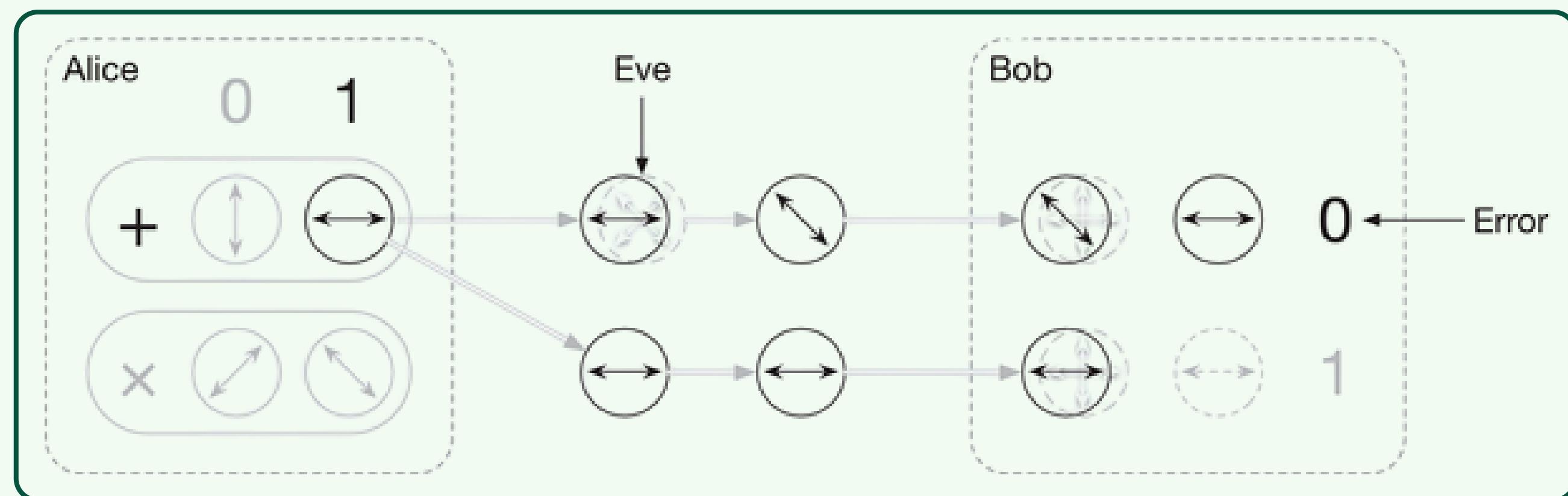
- Coste computacional (seguridad aumenta con el número de qbits)
- Falta de estandarización
- Vulnerabilidades (MiTM, Side-Channel Attacks, Supply Chain Attacks...)
- Coste económico elevado (vs. alternativas clásicas)

# QKD básico: BB84 sin ruido

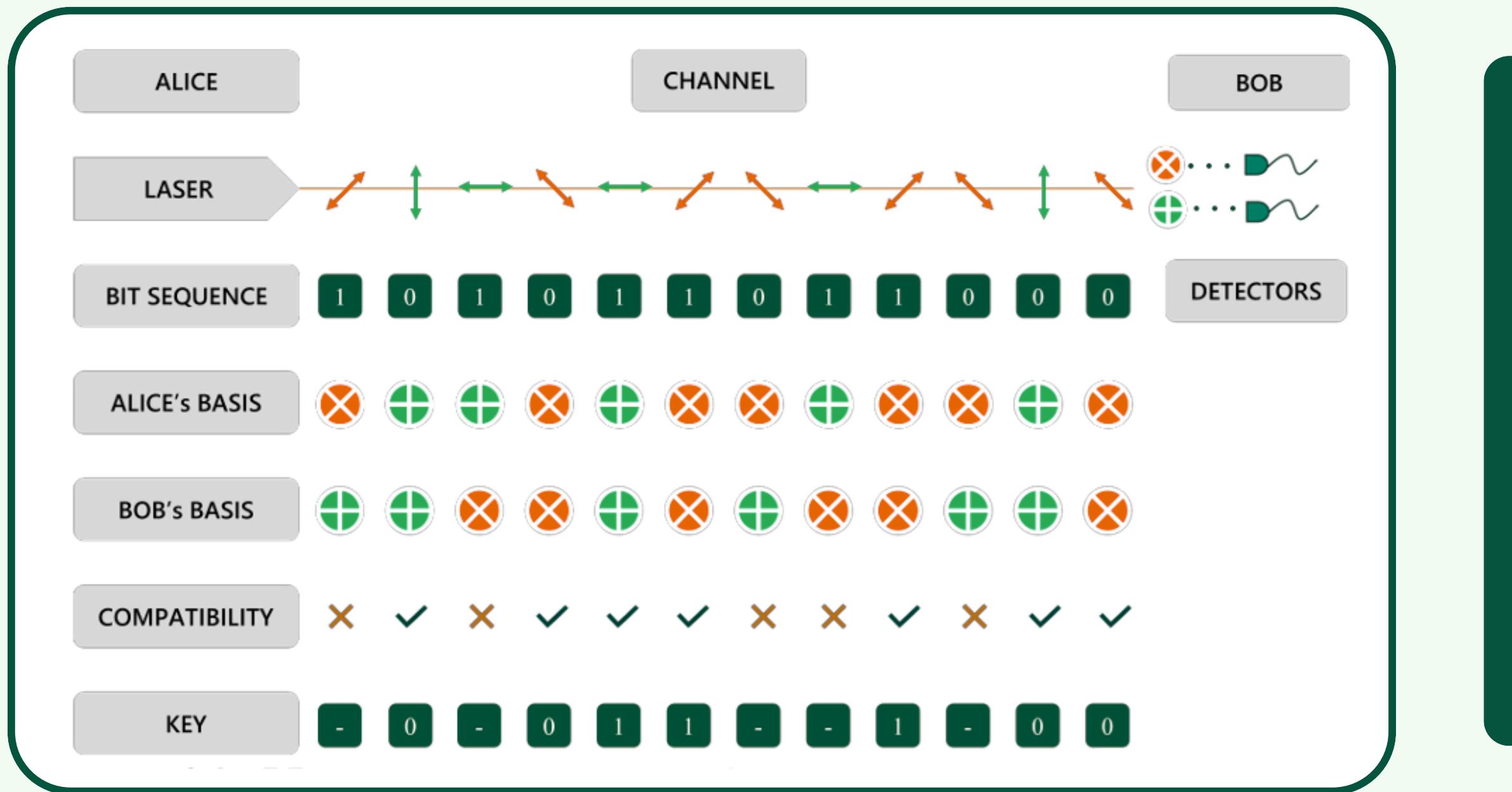
Protocolo de QKD  
basado en el colapso  
de la función de onda

# QKD: eavesdropping “impossible”

- Tercer observador (Eva) → **altera las probabilidades**
- A y B tendrán una discordancia → se podría detectar a Eva
  - Permite conocer amenazas para la privacidad



# QKD básico: BB84 sin ruido



1. Alice encoding
2. Canal clásico
3. Bob decoding
4. Repetir N veces
5. Sifting
6. Reconciliación

SCAN ME



# BB84 básico

- Código Qiskit
  - BB84 en práctica

```
def encode_bit(bit):
    qc = QuantumCircuit(1, 1)
    if bit == 1:
        qc.x(0)
    basis = random.choice(["Z", "X"])
    if basis == "X":
        qc.h(0)
    return qc, basis

def measure_bit(state, basis):
    if basis == "X":
        qc = QuantumCircuit(1)
        qc.h(0)

        state = Statevector(state).evolve(qc)

    statevector = Statevector(state)
    probabilities = statevector.probabilities_dict()
    prob_zero = probabilities.get('0', 0)
    prob_one = probabilities.get('1', 0)
    return 0 if random.random() < prob_zero else 1
```

# BB84 básico

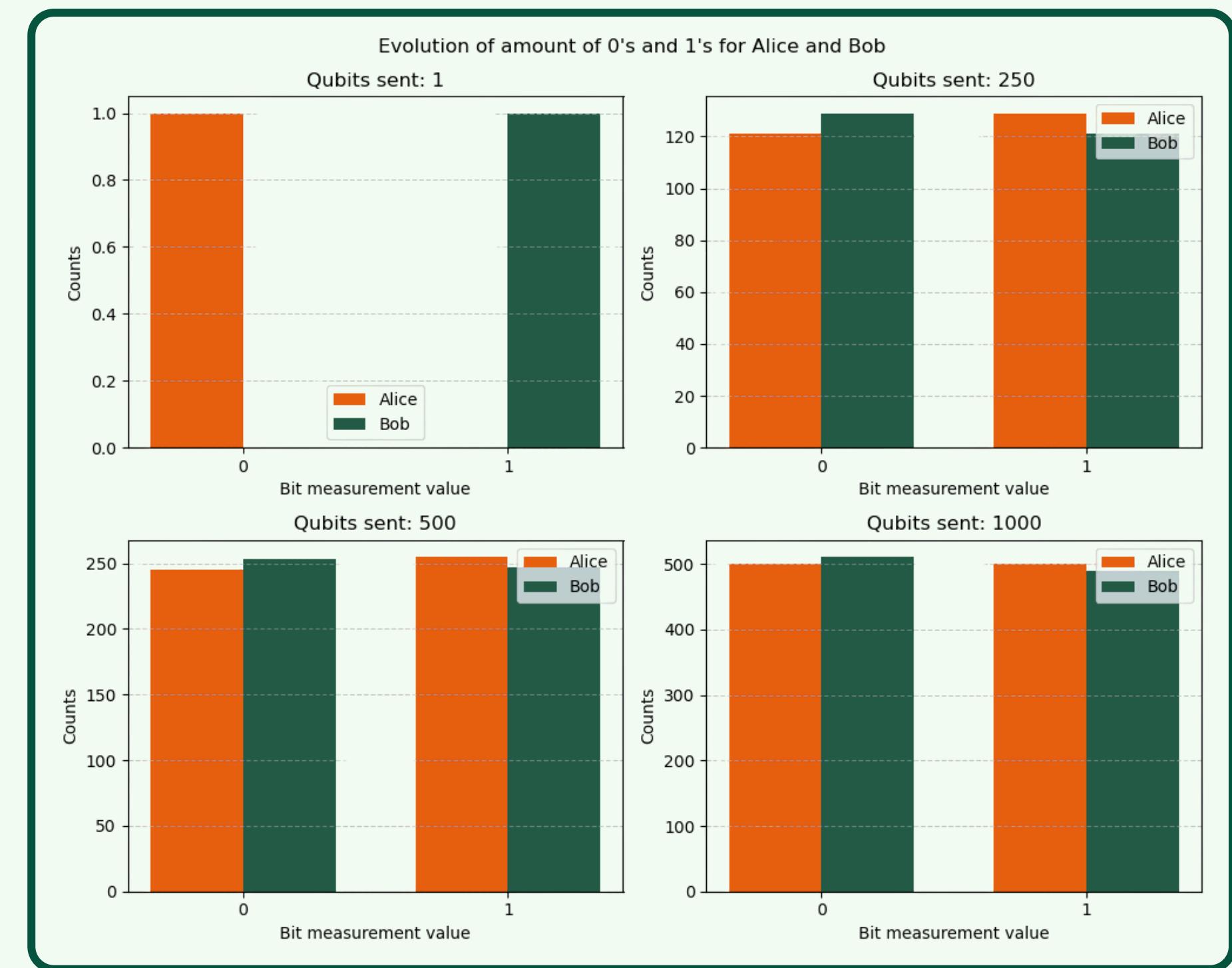
- Código Qiskit
  - **Sifting**: comprobación de la seguridad del canal cuántico
  - **Reconciliación**: selección de los bits con bases idénticas

```
key_indices = [i for i in range(n_bits) if alice_bases[i] == bob_bases[i]]  
alice_key = [alice_bitstring[i] for i in key_indices]  
bob_key = [bob_results[i] for i in key_indices]  
  
print("Alice's final key:", alice_key)  
print("Bob's final key:", bob_key)
```



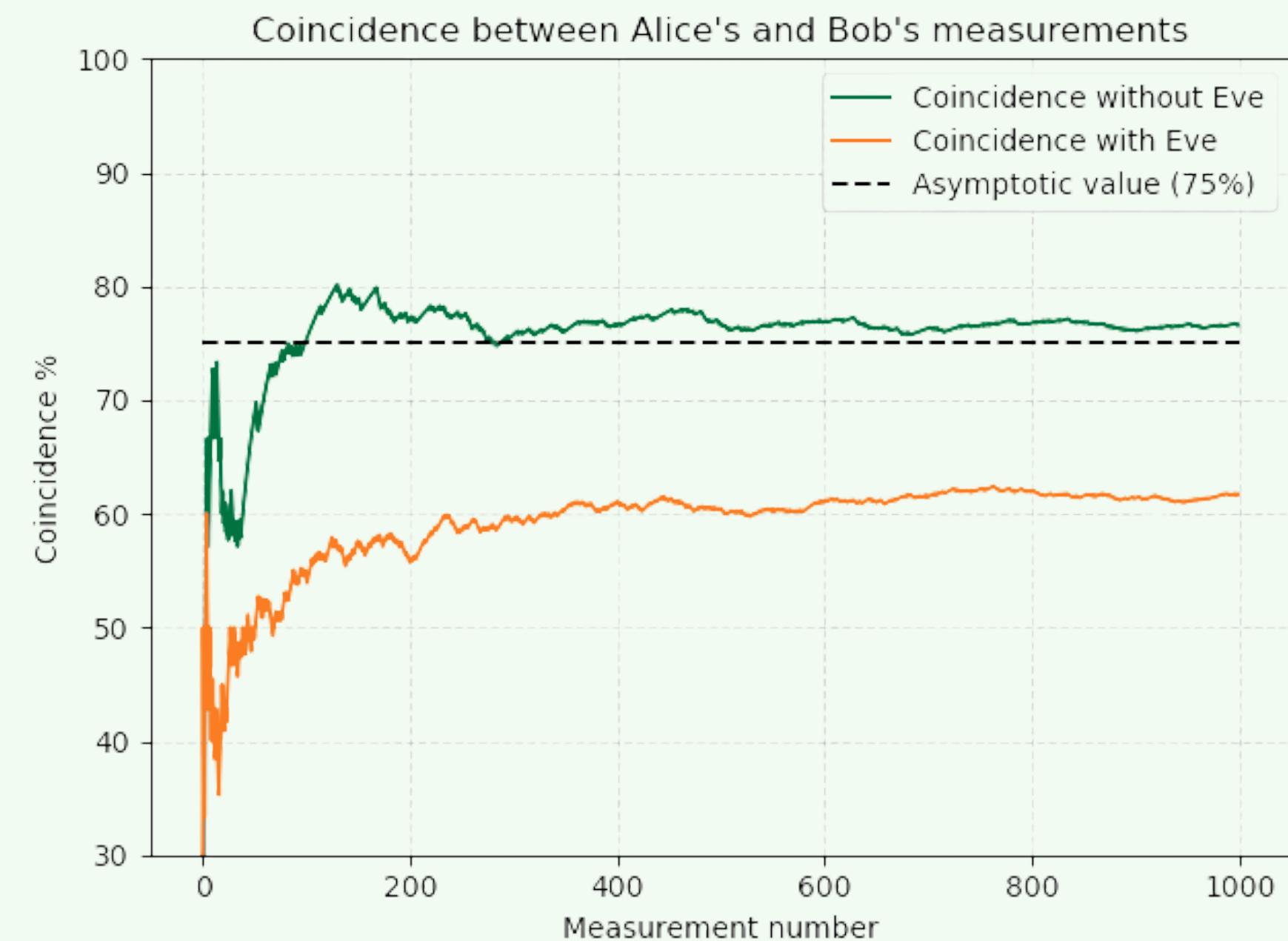
# BB84 básico

- Código Qiskit
  - Aleatoriedad



# BB84 básico

- Código Qiskit
  - Aleatoriedad
  - Presencia Eva



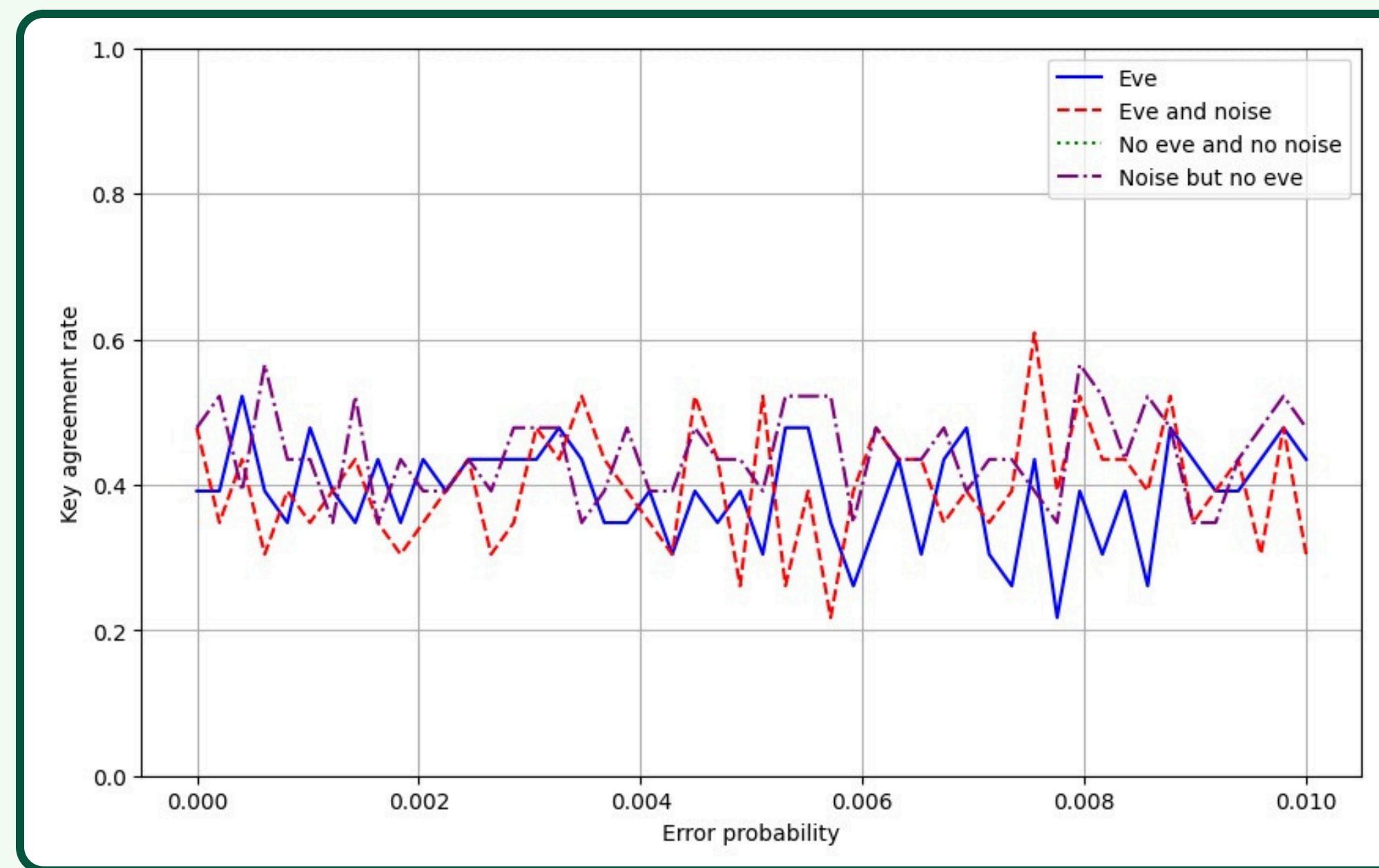
# BB84 en canal ruidoso

- Código Qiskit:
  - Añadimos modelo **sencillo** y **parametrizado** de ruido

```
def noise_model(p):  
    noise_model = NoiseModel()  
  
    amp_damp = amplitude_damping_error(p)  
    noise_model.add_all_qubit_quantum_error(amp_damp, ['x', 'h'])  
  
    depol = depolarizing_error(p, 1)  
    noise_model.add_all_qubit_quantum_error(depol, ['x', 'h'])  
  
    return noise_model
```

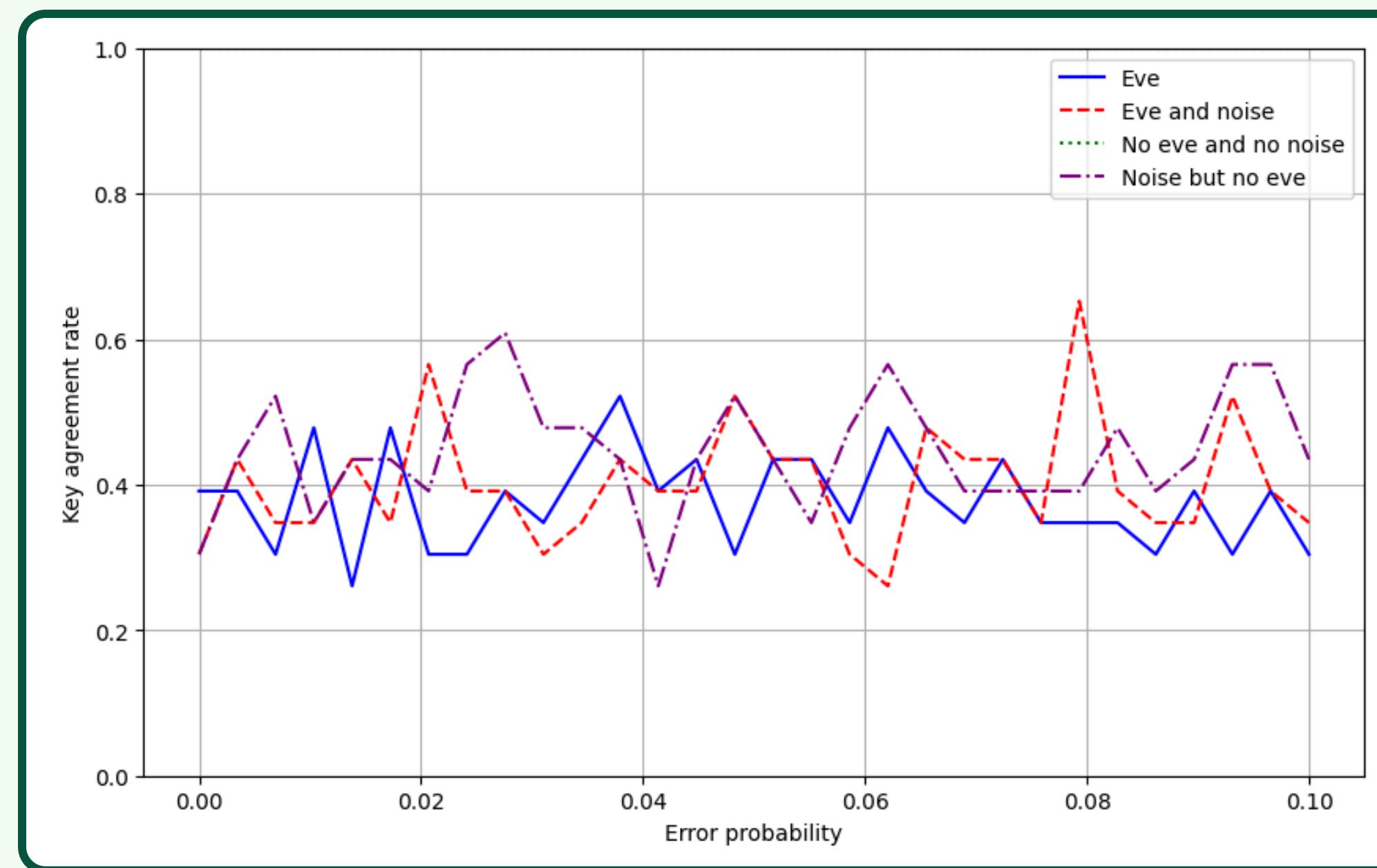
# BB84 en canal ruidoso

- Comparativa ruido vs. ideal: modelo de suite de testeo (umbral del 1%)



# BB84 en canal ruidoso

- Comparativa ruido vs. ideal: modelo de suite de testeo (umbral del 10%)



# Ampliación de privacidad

- Si el *mismatch* es superior al ruido → no todo está perdido
- **Ampliación de privacidad:** reducción de la información que una posible Eve haya obtenido

Cadena A	Cadena B	Ejemplo
hola	hola!	0110101001000111
↓ SHA-256	↓ SHA-256	↓ SHA-256
b221d9dbb083a7f33428d7 c2a3c3198ae925614d7021 0e28716ccaa7cd4ddb79	a80897215bfeca30c04e77 96c593f66024daf5a1d181 07671fd6f759cfa8ddd8	b8f5684a24a3f849d93d67 227bcf3f1c4969fc11c56b a8042e7c3efa97448fbc

# Limitaciones y próximos pasos

- Análisis de seguridad
- Certificaciones (Nostradamus, Vigo Quantum...)
- Caracterización y detección exhaustiva de fuentes de ruido
- Implementación de qubits lógicos (vs. qubits físicos)

# Conclusiones

- Relevancia actual protocolo BB84
  - Aplicaciones de todo tipo
- Vías de mejora: caracterización del ruido + ampliación de privacidad
- Queda mucho por hacer

Quantum Monkeys

*Jorge Giménez, Miguel Hernández, Andreu Moreno y Rubén Piles*

# Protocolo BB84

Modelo de *testeo* de sistemas QKD

QisFall  
kitFest

