Quantum Key Distribution

Google CTF 2019 Qualifiers

ilm0

Agenda

1. Understanding the challenge

2. Solving the challenge

3. How could it affect you?

The challenge

Crypto

134 solves

92/500 points

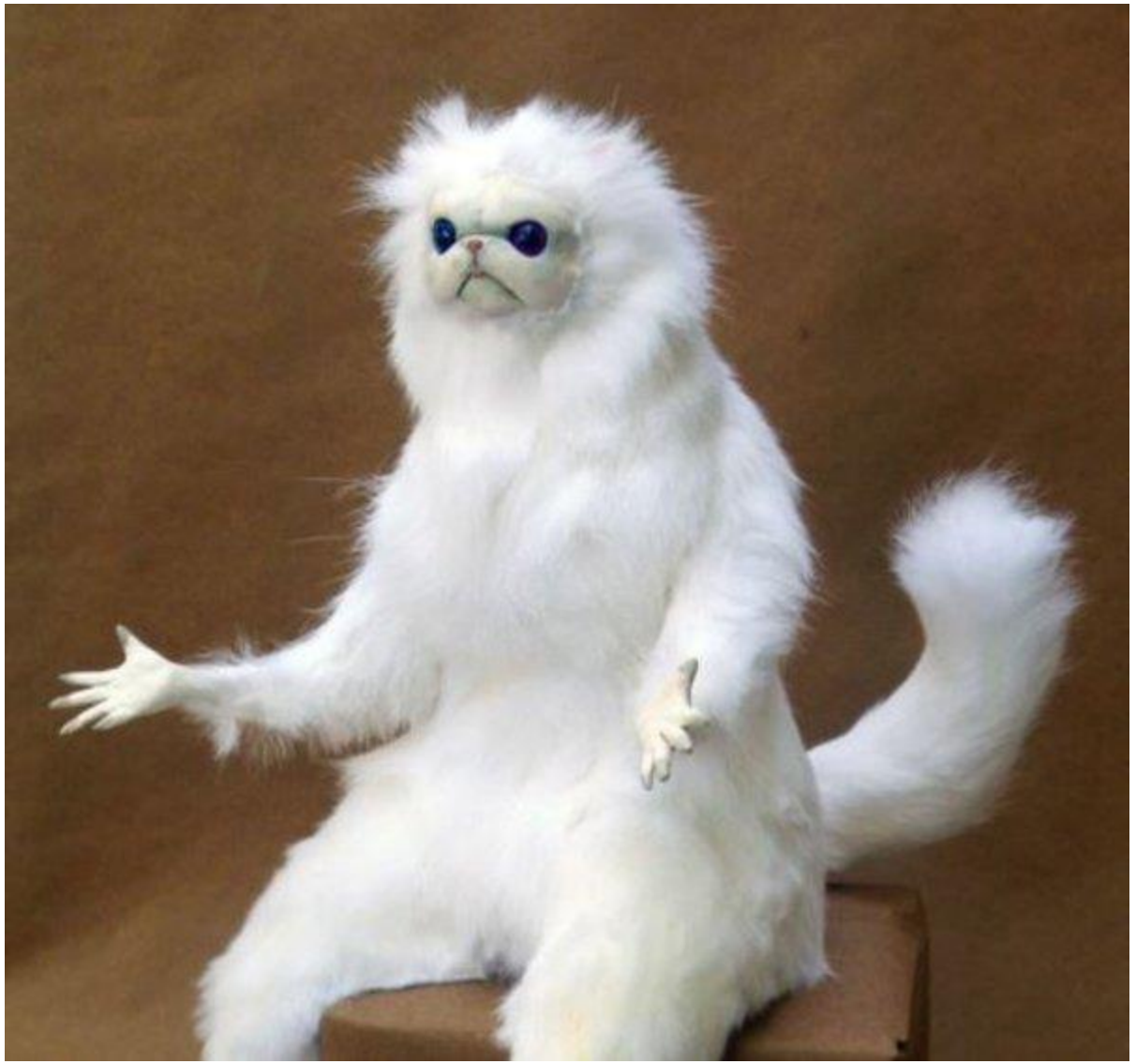
The challenge

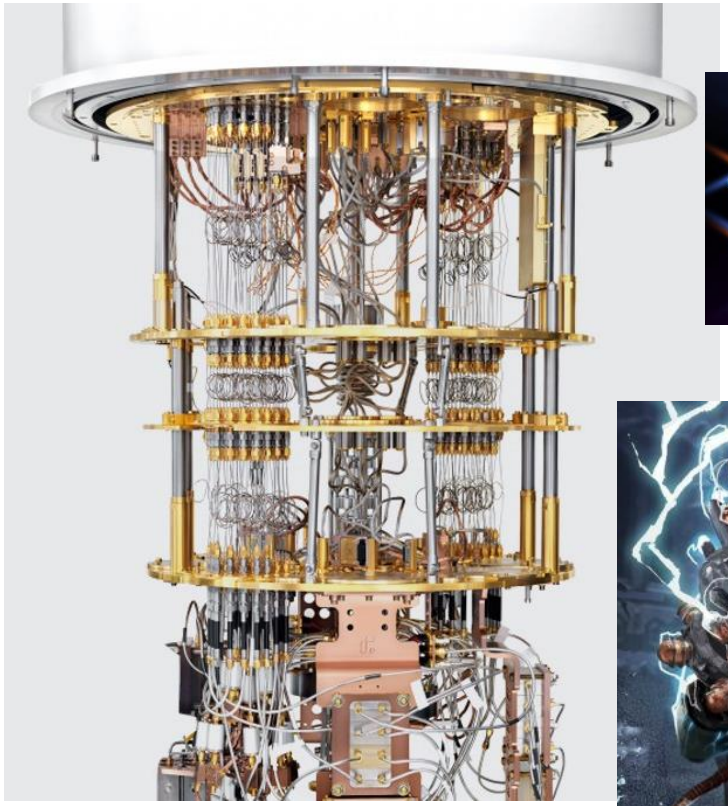
Crypto

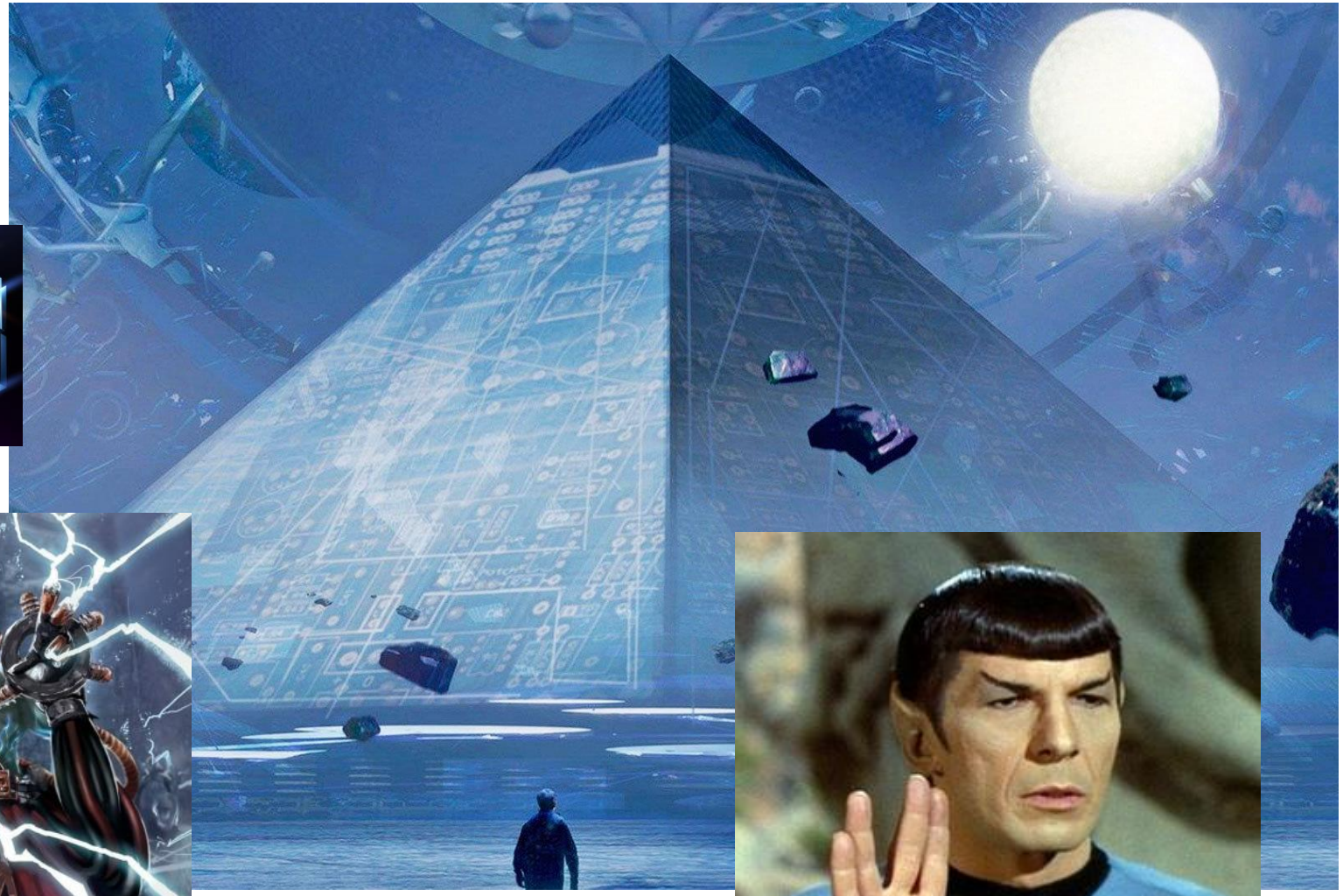
134 solves

92/500 points

But *quantum*?





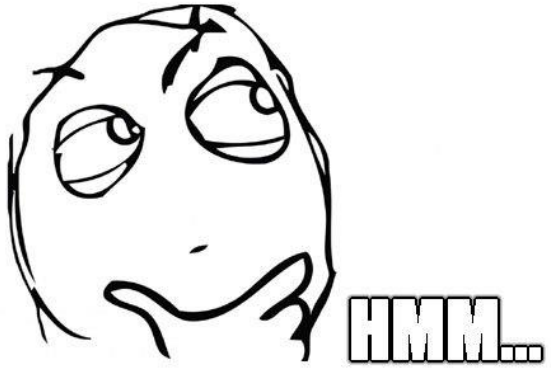


A closer look

We are simulating a quantum satellite that can exchange keys using qubits implementing BB84. You must POST the qubits and basis of measurement to '/qkd/qubits' and decode our satellite response, you can then derive the shared key and decrypt the flag. Send 512 qubits and basis to generate enough key bits.

A closer look

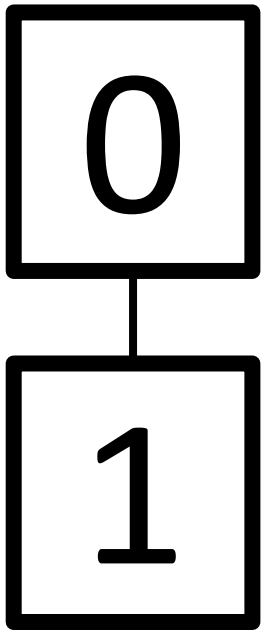
We are simulating a quantum satellite that can exchange keys using qubits implementing BB84. You must POST the qubits and basis of measurement to '/qkd/qubits' and decode our satellite response, you can then derive the shared key and decrypt the flag. Send 512 qubits and basis to generate enough key bits.



Quantum computing 101

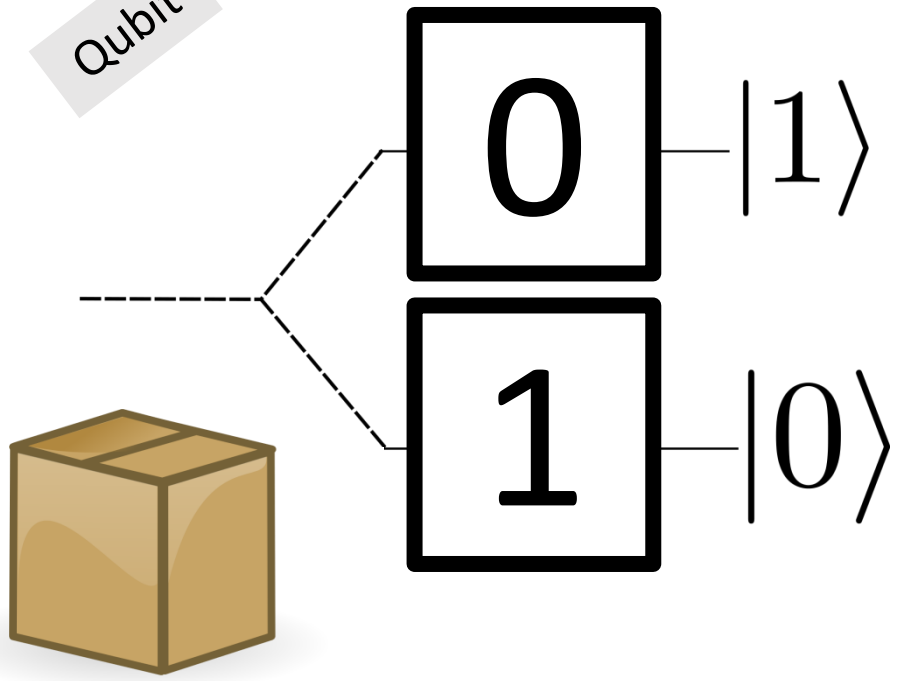
Classical CS

Bit



Quantum CS

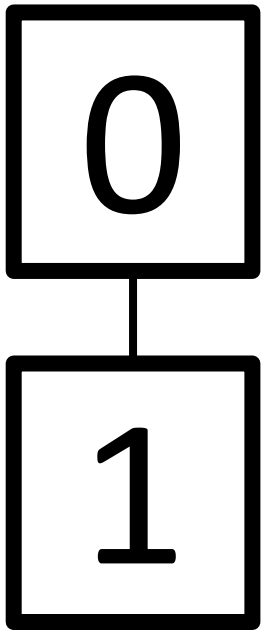
Qubit



Quantum computing 101

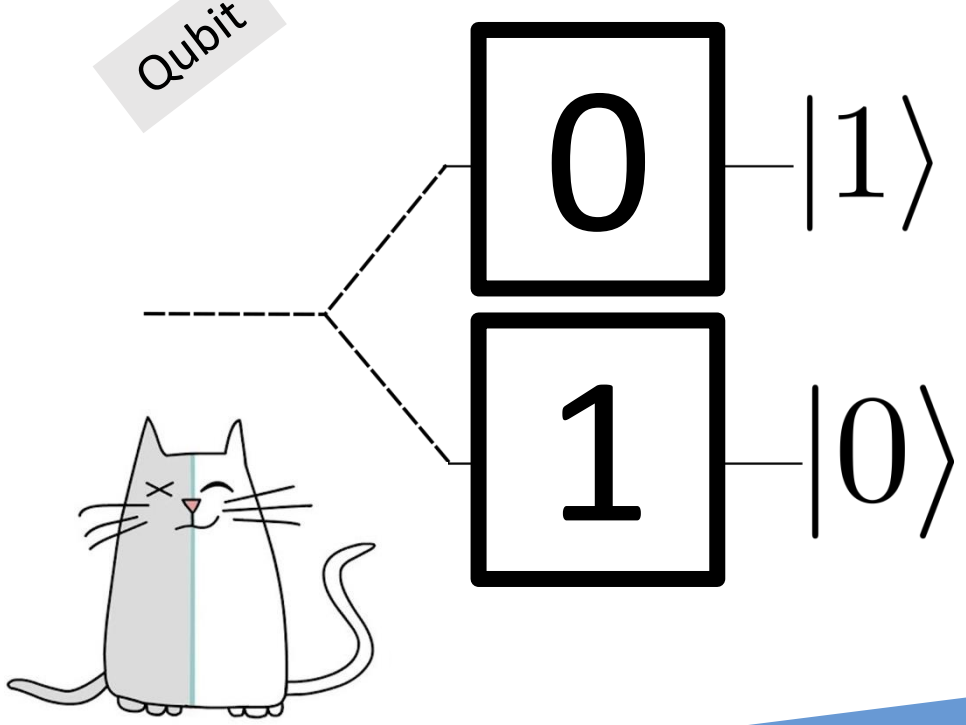
Classical CS

Bit



Quantum CS

Qubit



Qubits

$\alpha, \beta \in \mathbb{C}$

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

$$|\alpha|^2 + |\beta|^2 = 1$$

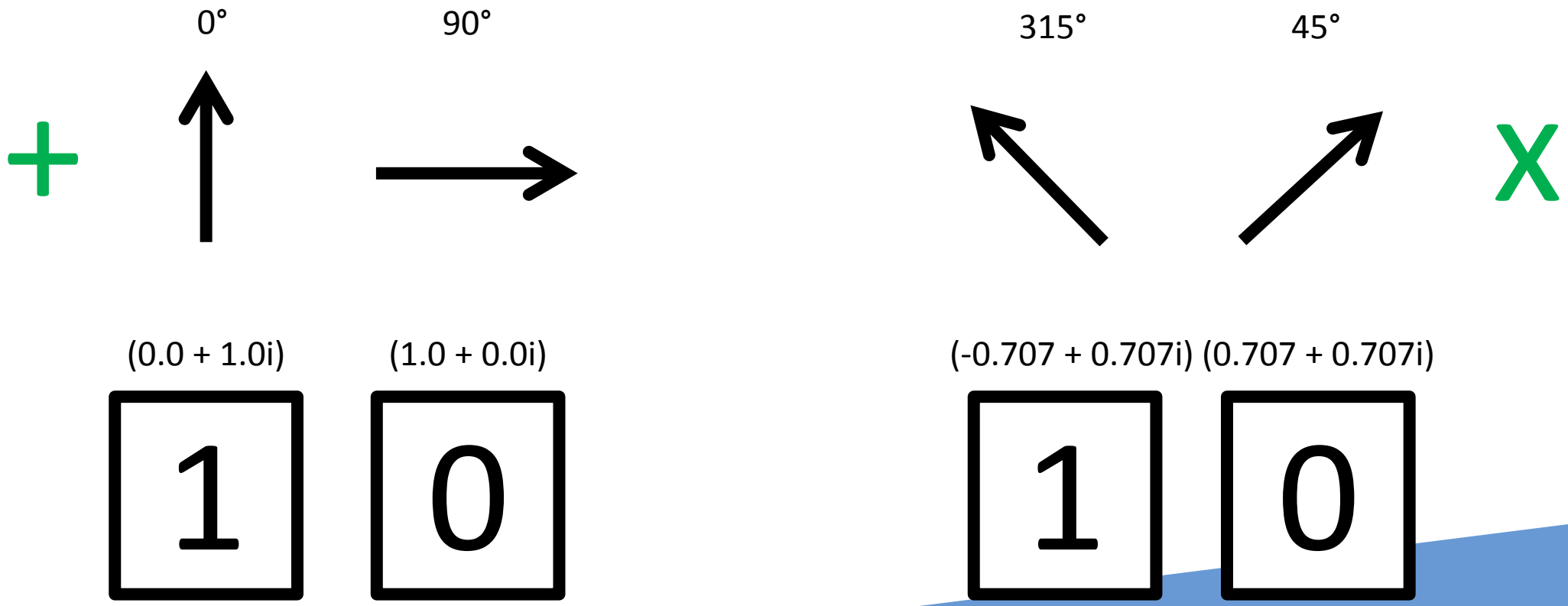
BB84

BB84

- Key distribution scheme cf. Diffie-Hellman
- Derive private key for one-time pad
- Works with 2 channels – 1 quantum, 1 traditional (authenticated)
- Uses photon polarization states

BB84

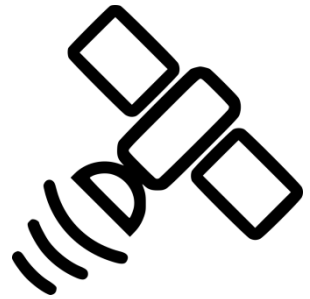
Photons → qubits
Information → polarization



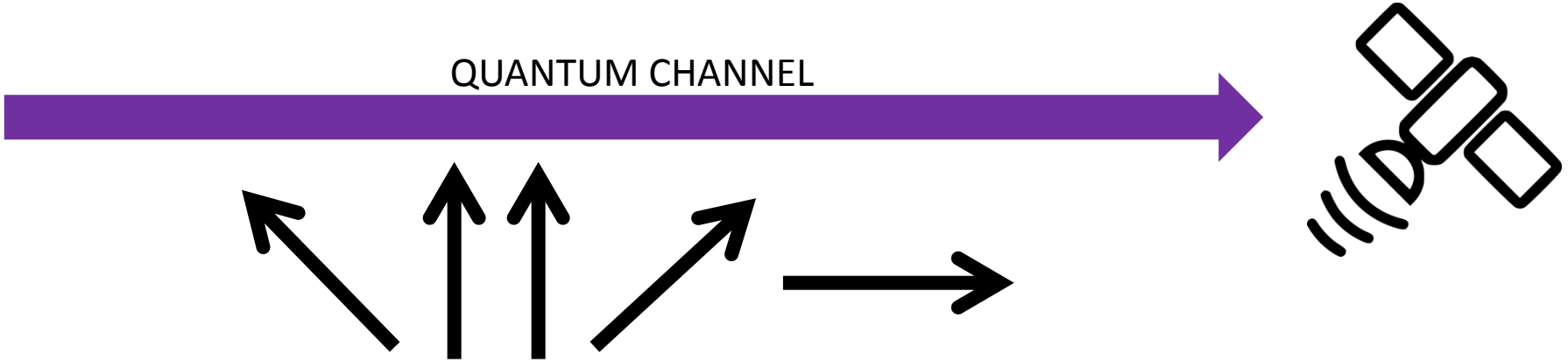
BB84

11100

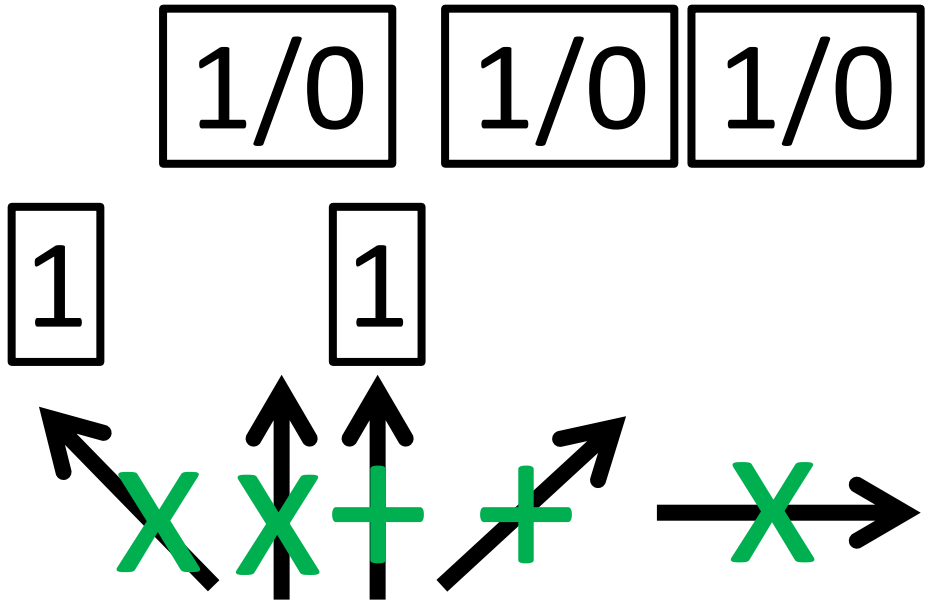
x+-x+



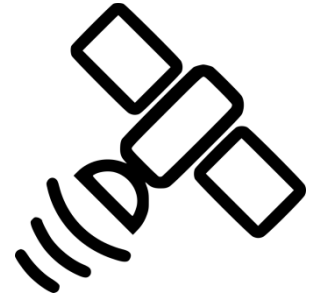
BB84



BB84



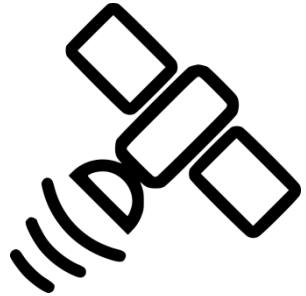
QUANTUM CHANNEL



BB84

TRADITIONAL AUTHENTICATED CHANNEL

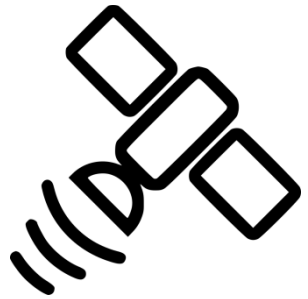
X X + + X



BB84

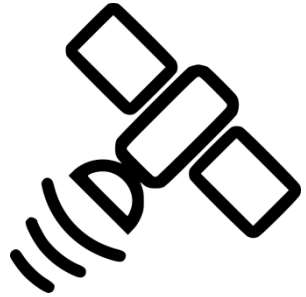
TRADITIONAL AUTHENTICATED CHANNEL

1st:OK 3rd:OK



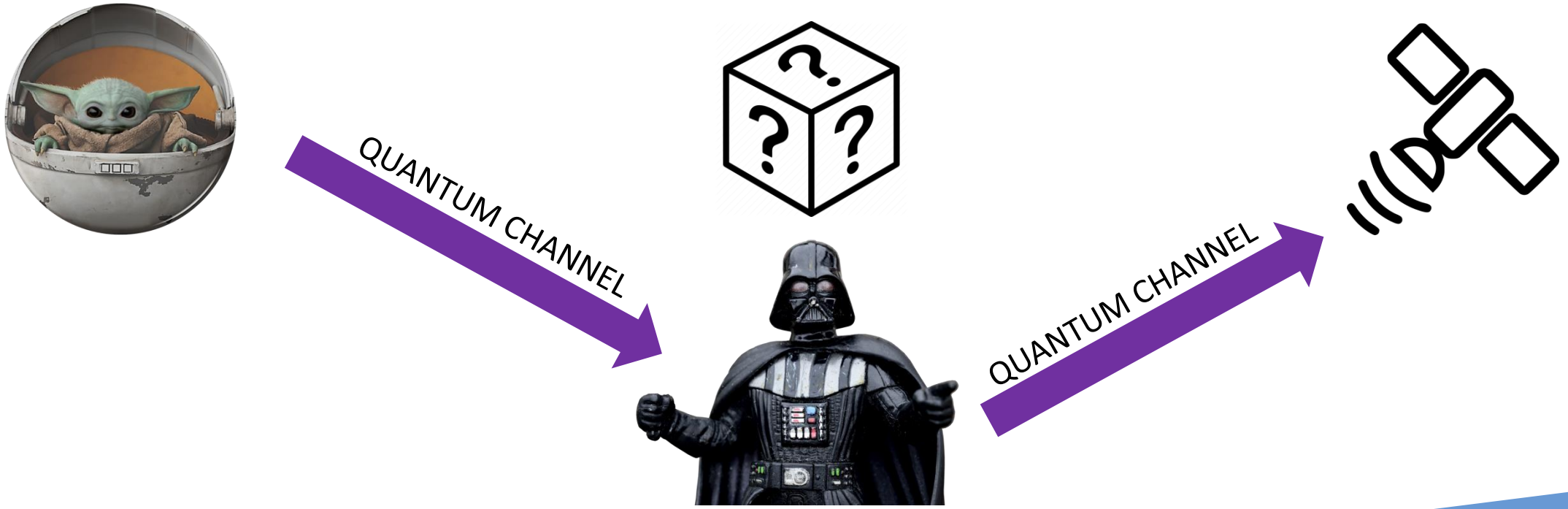
BB84

11



11

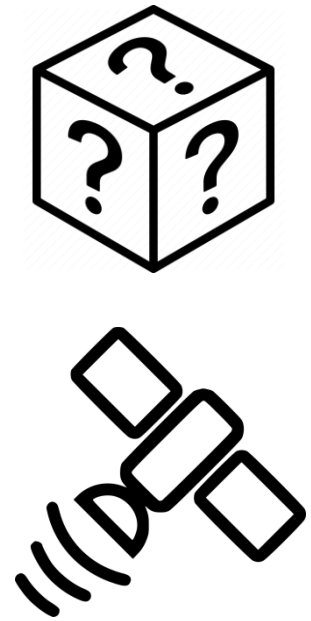
BB84 with eavesdropper



BB84 with eavesdropper

TRADITIONAL AUTHENTICATED CHANNEL

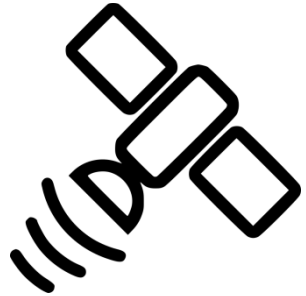
X X + + X



BB84 with eavesdropper

TRADITIONAL AUTHENTICATED CHANNEL

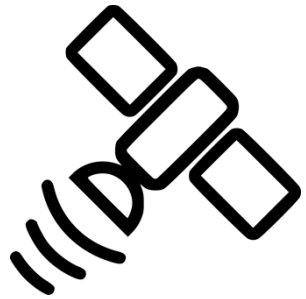
2nd:OK 3rd:OK



BB84

TRADITIONAL AUTHENTICATED CHANNEL

00



BB84

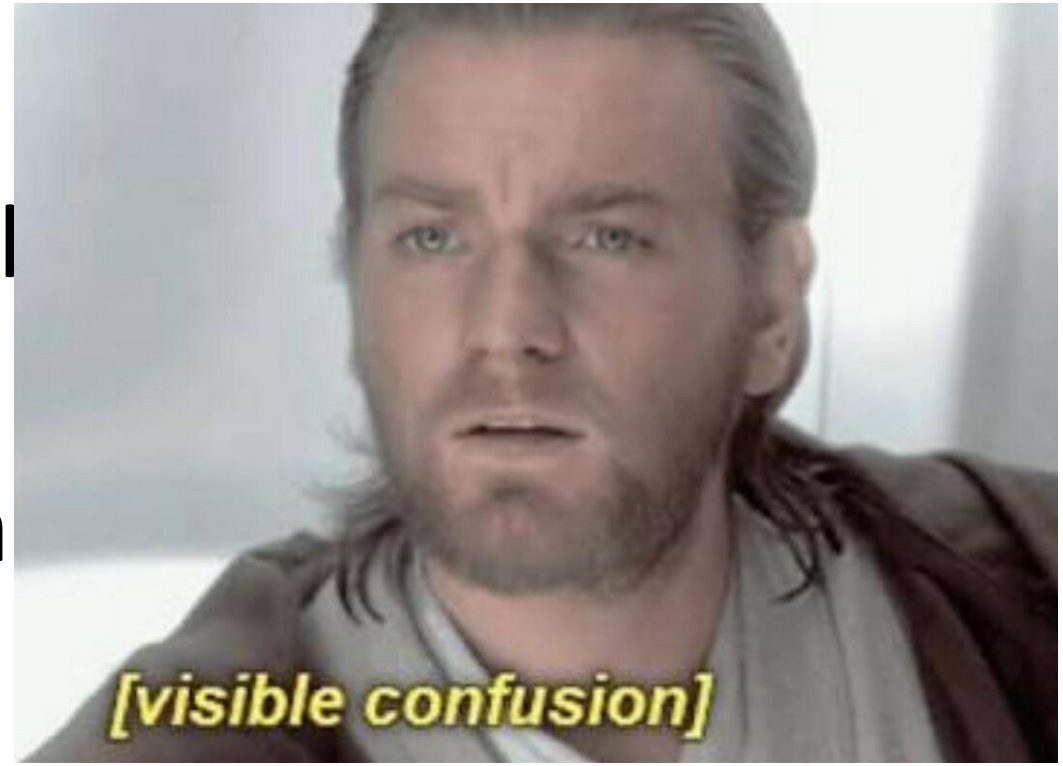
How do I make qubits?

I dont have any quantum channels!

BB84

How do I mal

I dont have a



A closer look /2

We are simulating a quantum satellite that can exchange keys using qubits implementing BB84. You must POST the qubits and basis of measurement to '/qkd/qubits' and decode our satellite response, you can then derive the shared key and decrypt the flag. Send 512 qubits and basis to generate enough key bits.

Cyphertext

Flag (base64)

```
U2FsdGVkX19OI2T2J9zJbjMrmI0YSTS+zJ7fnxu1YcGftgkeyVMMwa+NNMG6fGgjROM/hUvvUxUGhctU8fqH4titwti7HbwNMxFxfIR+lR4=
```

Example decryption with hex key 404c368bf890dd10abc3f4209437fcbb:

```
echo "404c368bf890dd10abc3f4209437fcbb" > /tmp/plain.key; xxd -r -p
/tmp/plain.key > /tmp/enc.key echo


"U2FsdGVkX182ynnLNxv9RdNdB44BtwkjHJpTcsWU+NFj2RfQIOpHKYk1RX5i+jKO" |
openssl enc -d -aes-256-cbc -pbkdf2 -md sha1 -base64 --pass file:/tmp/enc.key
```

Satellite server

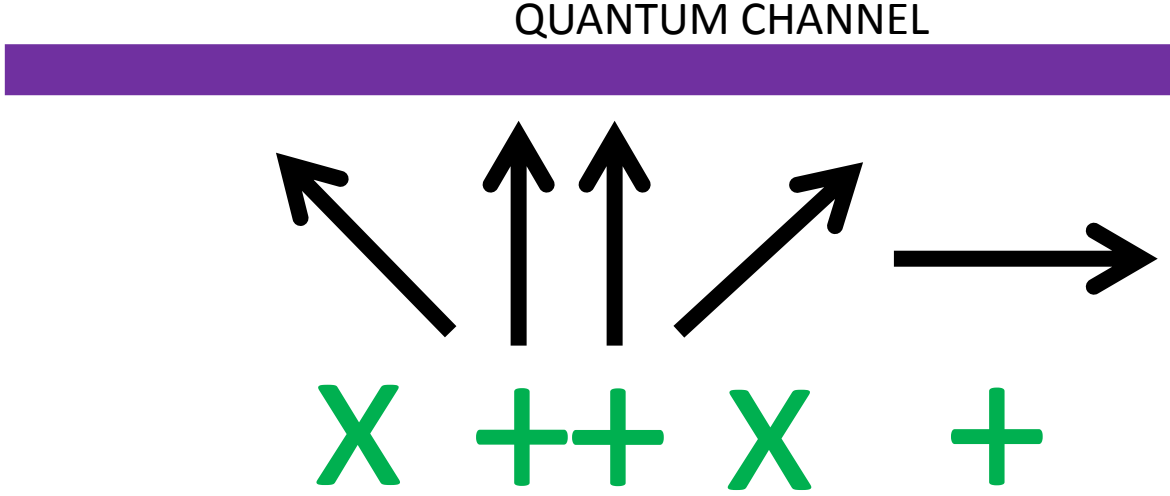
```
# Receive user's qubits and basis, return the derived key and our basis.
def perform(rx_qubits, rx_basis):
  random.seed()
  # Multiply the amount of bits in the encryption key by 4 to obtain the amount of basis.
  sat_basis = [random.choice('+x') for _ in range(len(current_app.config['ENCRYPTION_KEY'])*16)]
  measured_bits = measure(unmarshal(rx_qubits), sat_basis)
  binary_key, err = compare_bases_and_generate_key(rx_basis, sat_basis, measured_bits)
  if err:
    return None, None, err
  # ENCRYPTION_KEY is in hex, so multiply by 4 to get the bit length.
  binary_key = binary_key[:len(current_app.config['ENCRYPTION_KEY'])*4]
  if len(binary_key) < (len(current_app.config['ENCRYPTION_KEY'])*4):
    return None, sat_basis, "not enough bits to create shared key: %d want: %d" % (len(binary_key),
      len(current_app.config['ENCRYPTION_KEY']))
  return binary_key, sat_basis, None
```

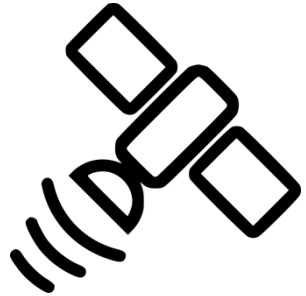
Satellite server

```
# Receive user's qubits and basis, return the derived key and our basis.
def perform(rx_qubits, rx_basis):
  random.seed()
  # Multiply the amount of bits in the encryption key by 4 to obtain the amount of basis.
  sat_basis = [random.choice('+x') for _ in range(len(current_app.config['ENCRYPTION_KEY'])*16)]
  measured_bits = measure(unmarshal(rx_qubits), sat_basis)
  binary_key, err = compare_bases_and_generate_key(rx_basis, sat_basis, measured_bits)
  if err:
    return None, None, err
  # ENCRYPTION_KEY is in hex, so multiply by 4 to get the bit length.
  binary_key = binary_key[:len(current_app.config['ENCRYPTION_KEY'])*4]
  if len(binary_key) < (len(current_app.config['ENCRYPTION_KEY'])*4):
    return None, sat_basis, "not enough bits to create shared key: %d want: %d" % (len(binary_key),
      len(current_app.config['ENCRYPTION_KEY']))
  return binary_key, sat_basis, None
```

Solving the challenge

First try





Generate key

```
def compare_bases_and_generate_key(tx_bases, rx_bases, measure):
  """Compares TX and RX bases and return the selected bits."""
  if not (len(tx_bases) == len(rx_bases) == len(measure)):
    return None, "tx_bases(%d), rx_bases(%d) and measure(%d) must have the same length." %
(len(tx_bases), len(rx_bases), len(measure))
  ret = ''
  for bit, tx_base, rx_base in zip(measure, tx_bases, rx_bases):
    if tx_base == rx_base:
      ret += bit
  return ret, None
```

Mighty XOR

- One time pad uses XOR
- But what should I XOR?
- Involutory function
  XOR(k, XOR(k, x)) = x

Can all this affect you?
Should you care?

3

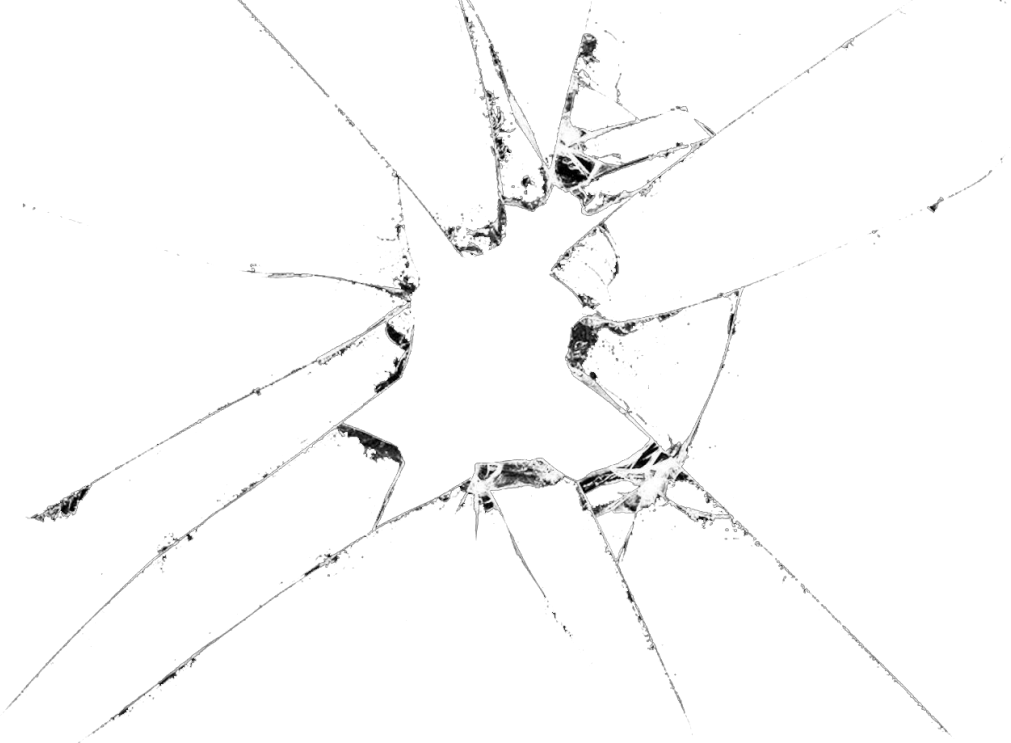
What can quantum computers do?

- Only run probabilistic algorithms
- Physical and financial modelling
  ↳ Predictions
- Searching an unstructured database
  - Grover's algorithm

What can quantum computers do?

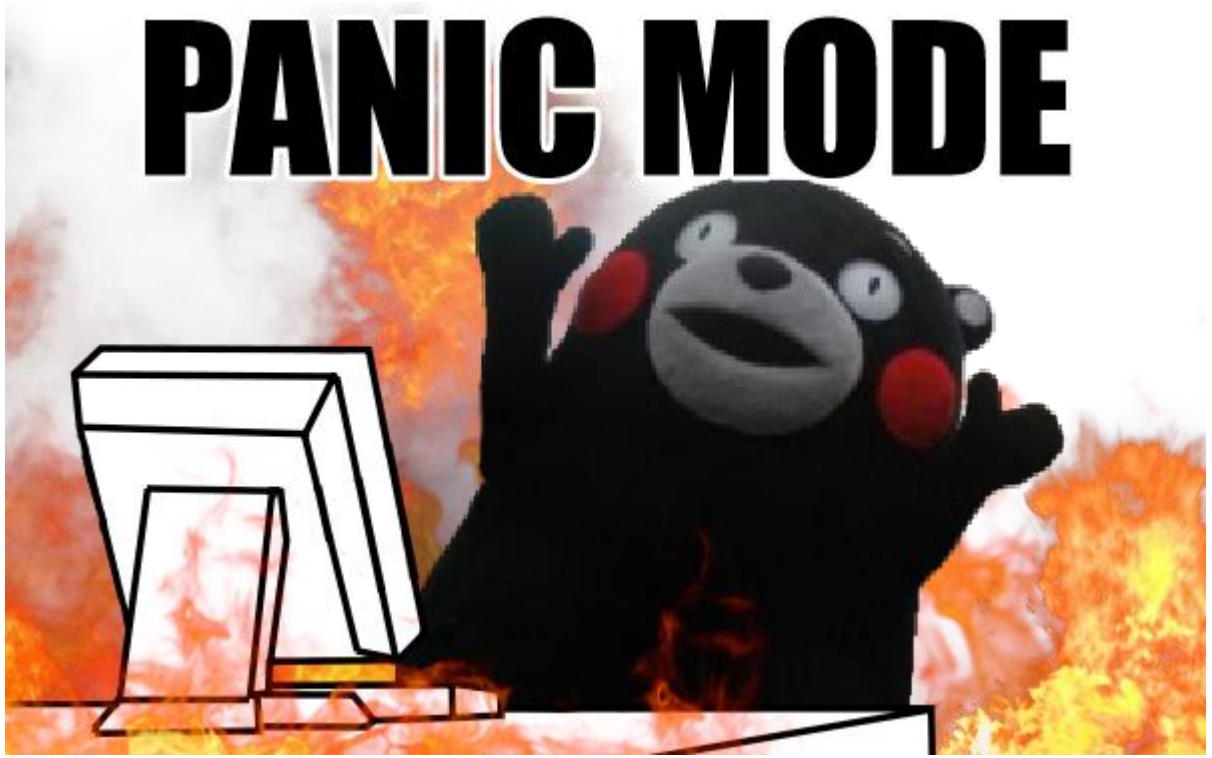
- Only run probabilistic algorithms
- Physical and financial modelling
  ↳ Predictions
- Searching an unstructured database
  - Grover's algorithm
- Integer factorization → Shor's algorithm

Post quantum cryptography



Integer factorization
Discrete log problem

RSA, Diffie-Hellman and ECDSA



PANIC?!

How to prepare

Table 1: Security Comparison						
Type of Attack	Symmetric Encryption			Public Key Encryption		
		Key Length	Bits of Security		Key Length	Bits of Security
Classical Computers	AES-128	128	128	RSA-2048	2048	112
	AES-256	256	256	RSA-15360	15,360	256
Quantum Computers	AES-128	128	64	RSA-2048	2048	25
	AES-256	256	128	RSA-15360	15,360	31

- Several families of quantum-resistant PKC
  - ↳ Lattice-based, code-based, hash-based and multivariate

- Use strong symmetric crypto

Where is quantum now?

- Big claims, little results 
- Quantum supremacy
- Many sceptics, even more obstacles
- Use cases very targeted → not a replacement, but complement

Lessons learned

- Understanding key concepts of a challenge is essential before 'getting dirty'
- Crypto basics aswell
- Possibility of post-quantum crypto

Thank you for your attention!


OH ALICE... YOU'RE THE ONE FOR ME
BUT BOB... IN A QUANTUM WORLD HOW CAN WE BE SURE
ψ^+ or ψ^- ?

Questions?