## Hackathon challenges

In this document we present four different challenges you can choose to work on during the hackathon. The goal of these challenges is both for you to grow your knowledge about quantum networking, and to actually program quantum networking applications. While this document describes the basics of the required protocols, you will be expected to explore the relevant references to learn all the details necessary for the implementation.

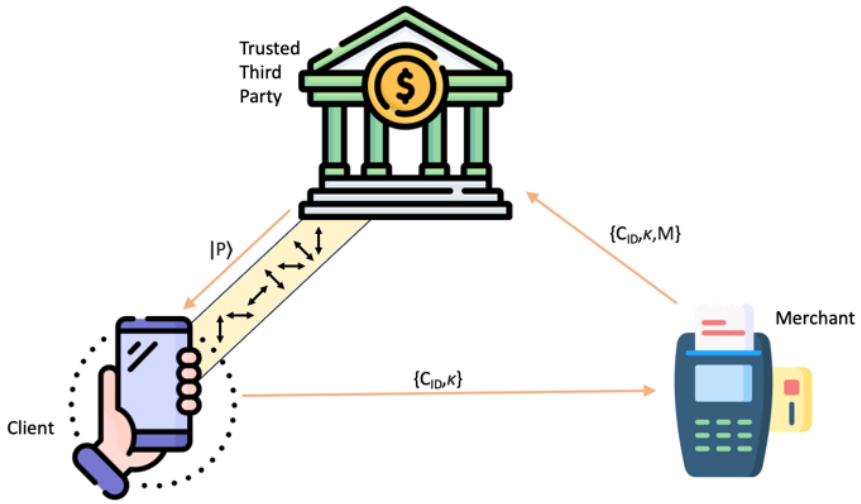
The simulation framework for the event will be the SquidASM quantum network simulator, which is also being used within the QIA Consortium to simulate applications and inform hardware design.

In addition to the four challenges described below, participants may also choose to work on a problem or application of their own choosing. For example, by implementing one of the many protocols in the Quantum Protocol Zoo.

### Secure Quantum Digital Payments

The team should design and implement, using SquidASM, a simulation of the quantum-digital payment protocol described in paper [1].

There are three parties: a Client, a Merchant, and a Bank/Creditcard institute (denoted as Trusted Third Party, TTP). We do not assume any quantum or classical communication channel to be trusted, except an initial prior step between the TTP and Client for an account creation (in which the Client receives a secret token C from the TTP).




The protocol:

1. During a payment, the TTP generates a random bitstring b and a random conjugate basis-string B, both of length $\lambda$. The j-th bit of b is encoded onto a quantum state prepared in the j-th element of B. For example, assuming $\lambda$ = 4 and Bj in {+/-;0/1}, choosing b = 0101 and B = 0011 would result in a 4-qubit quantum state $|P\rangle = |+\rangle|-\rangle|0\rangle|1\rangle$. Indeed, the first bit of B (i.e., 0) selects +/- and the first bit of b (i.e., 0) selects +. And so on. The TTP sends $|P\rangle$ to the Client.
2. Then, the Client calculates m=MAC(C,M), denoted as output tag, where M is the identifier of the Merchant and MAC is a Message Authentication Code. The client interprets m as a basis-

string and privately measures $|P\rangle$ according to m. The resulting string of measurements, denoted as $\kappa$, constitutes a cryptogram.

3. The Client sends its public identifier $C_{ID}$ (not to be confused with C, which is a secret between the Client and the TTP) to the Merchant, along with $\kappa$. The Merchant sends $\{C_{ID},\kappa,M\}$ to the TTP for verification.
4. To authorize the purchase, the TTP looks up C (the secret token corresponding to $C_{ID}$) and calculates m = MAC(C,M). The TTP accepts the transaction (and transfers the money from the Client's account to the Merchant's one) if and only if $\kappa_j = b_j$ when $m_j = B_j$. The TTP rejects otherwise.

It is suggested to start by simulating the execution of the protocol described above, assuming that the Client and the Merchant are both honest. For simplicity, assume that C is already shared between the Client and the TTP, when the protocol execution starts.

Then, also simulate malicious behaviours.

- A malicious merchant M' would try to forge an output tag such that MAC(C,M)=MAC(C,M') $\Leftrightarrow$ m=m' $\Leftrightarrow$ $\kappa=\kappa'$. In this way, merchant M' could receive the payment that should be sent to merchant M. The value of parameter $\lambda$ should be selected so that the probability of output tag forging is minimised (see [1] for details). Show that non-ideal choices for $\lambda$ may lead to wrong payments with high probability.
- A malicious Client would try to exploit the imperfection of real devices (inaccurate state preparation, lossy quantum channels, etc.) to circumvent the commitment or double-spend the cryptogram. In fact, some bits in step 4 will be unequal, although measured in the same basis, and the protocol would abort even though it was followed honestly. Suppose the TTP tolerates as many as 50% losses. A malicious Client could measure half of the quantum token $|P\rangle$ in the basis for $M_0$ and the other half in the basis for $M_1$ (being $M_0$ and $M_1$ two merchants), thus creating two successfully committed tokens. Try to figure out how to prevent this kind of attack (a possible solution is illustrated in [1]).

The team should feel free to choose the preferred qubit technology, among those available with SquidASM (generic hardware, NV centers, color centers). Plot the fidelity of the quantum states for different values of the physical parameters for qubits and quantum links.

## References

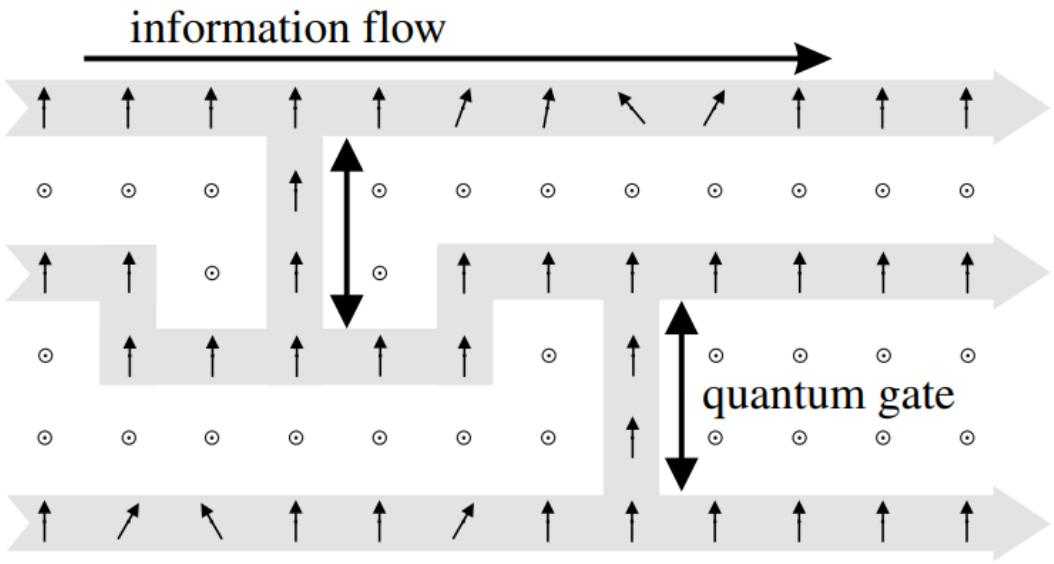
*[1] Peter Schiansky, Julia Kalb, Esther Sztatecsny, Marie-Christine Roehsner, Tobias Guggemos, Alessandro Trenti, Mathieu Bozzio, and Philip Walther. Demonstration of quantum-digital payments. Nature Communications, 14(1), Jun 2023.*

## Grover search using Blind Quantum Computation

In this challenge, the team should simulate the execution of Grover's algorithm in a secure, delegated way, using Blind Quantum Computation (BQC).

BQC is a protocol for carrying out a computation on a remote quantum server, in such a way that the inputs to the computation remain hidden from the server. The only requirement on the client is the ability to transmit single-qubit states to the server. In SquidASM this can be done by performing quantum teleportation of single qubits.

BQC is based on a form of quantum computation called one-way quantum computing. In this model, one initially creates a large entangled state (cluster state), and performs single qubit measurements on part of the state in sequence. Because of the entanglement in the state, the choice of single-qubit measurement angles has an effect on the nearby qubits. This therefore propagates information through the state, thereby doing a computation.

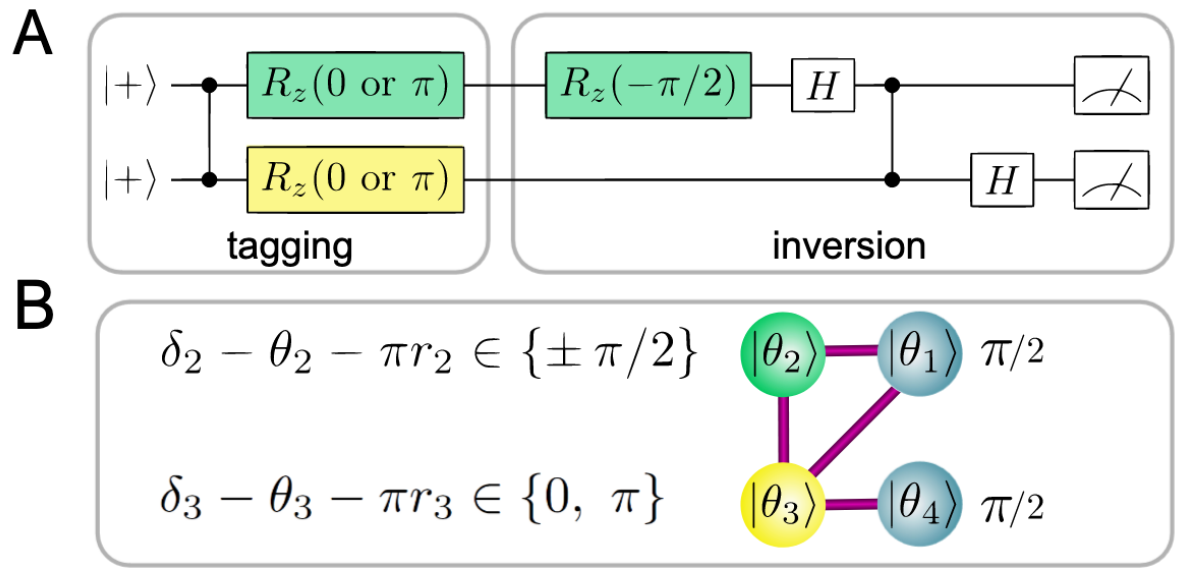



*In one-way quantum computation one initially prepares a large entangled cluster state. A computation can then be performed by measuring qubits at particular angles. In the image, the circles represent qubits that are measured in the Z basis, which has the effect of removing them from the entangled state. The arrows represent the measurement angle of qubits in the X-Y plane, and this propagates information through the entangled cluster state. Image from [1]*

In BQC the measurement angles that determine the computation are encoded in the single-qubit states sent by the client and are teleported into the cluster state [2].

BQC can be used to run for example Grover's algorithm. This is a quantum search algorithm that can find an element in an unstructured list in $O(\sqrt{N})$ steps instead of the $O(N)$ steps of a classical algorithm that iterates through the list. An introduction to this algorithm can be found at [3].

An optimized version of BQC can be used to run a short Grover circuit using a four-qubit cluster-state [4], and your challenge will be to implement this in SquidASM.




*In **A** an example of a two-qubit grover search circuit is shown. The first two gates act as an oracle, applying a phase shift to one of the four computational basis states that should be tagged. The second part of the circuit performs the amplitude amplification step to boost the probability amplitude of the correct term in the superposition state. **B** shows an implementation of this circuit using BQC, with a four-qubit cluster state. The angles θ are the measurement angles of the qubits. Image from [4]*

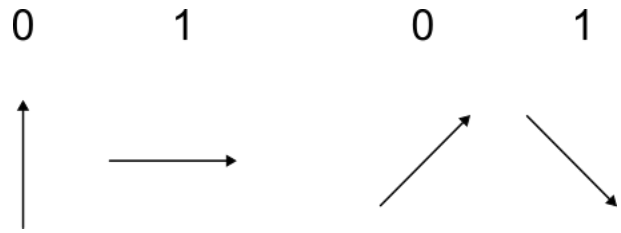
Steps of the challenge:

1. Create the four-qubit cluster state by initializing all qubits on the server in the |+> state and performing local complementation on qubit 2 [5]
2. Verify that you can perform a single-step Grover search on this state using the method from [4]. Compare this to the equivalent quantum circuit.
3. Send a qubit from a client and use this qubit to control the measurement angle of a qubit in the resource state [2,4]
4. Execute the Grover circuit by teleporting two client qubits, and sending the necessary classical information to the server


[1] https://pennylane.ai/qml/demos/tutorial_mbqc/
[2] https://www.scottaaronson.com/showcase2/report/charles-herder.pdf
[3] https://pennylane.ai/qml/demos/tutorial_grovers_algorithm/
[4] https://arxiv.org/pdf/1110.1381.pdf
[5] https://peterrohde.org/an-introduction-to-graph-states/#Local_complementation

## Extending QKD implementation

The goal of this challenge will be to build on an existing implementation of quantum key distribution (QKD) within SquidASM [1]. QKD is a method of building a secure cryptographic key between two parties by exchanging single-qubit quantum states. In the most basic QKD protocol, called BB84, one party (Alice) sends single photons to a second party (Bob) through an authenticated by not necessarily trusted channel. Alice randomly choses to encode a bit (0 or 1) in the polarisation of the each photon, and additionally randomly choses to either use the horizontal/vertical polarisations, or the diagonal/anti-diagonal polarisations:




Both pairs of states are said to form a basis. Upon receiving a photon, Bob randomly chooses which basis in which to perform a measurement of the polarisation. If he choses the same basis as Alice used for the encoding, he will recover the bit encoded by Alice with 100% probability (in an ideal scenario). However, if he choses the opposite basis to measure in, the outcome of this measurement will be completely random. A series of state preparations / measurements in the protocol might look like the following:

| Alice bit | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Alice basis | x | + | + | x | x | + | + | x | x | x | + | + | x | + | x | + | x |
| Bob basis | x | x | + | + | + | x | + | x | + | x | x | + | x | X | + | + | x |
| Bob bit | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |

Here “+” refers to the horizontal/vertical basis and “x” to the diagonal basis. Approximately half the time Alice and Bob pick the same measurement basis, and share correlated information. Since this information is generated at the time of measurement it is not known to any eavesdropper, and can be used as a secure cryptographic key. In order to establish this key, Alice and Bob publicly announce what measurement settings they used in each round, and disregards the measurement rounds in which they measured in different bases.

One potential attack against QKD is that an eavesdropper intercepts the photons in the channel, measures them and retransmits them to Bob. However, since the attacker does not know what basis Alice used (this is only announced later), they cannot always recover Alice’s bit. When they prepare a new state to send to Bob, they will therefore make mistakes, by preparing a state in a different basis than Alice. For example, Alice might prepare a state in the “+” basis, an attacker Eve randomly measures it in the “x” basis and prepares a new state in this basis. Then, if Bob measures the state he receives from Eve in the “+” basis he will find, half of the time, that it does not agree with Alice’s bit, even though they used the same basis. Errors in the cryptographic key can therefore be used to infer the presence of an eavesdropper.

As part of the protocol, Alice and Bob therefore need to use part of their cryptographic key to estimate the quantum bit error rate – the fraction of faulty bits in their key. However, errors may also be caused by experimental sources of noise in the protocol, and are expected even during normal

operation. To obtain a useable key, Alice and Bob therefore need to remove any errors in their key, without leaking the key itself [2]. This is called key reconciliation. One way to do this is by performing an XOR operation between neighboring bit values in the key and comparing the results. The XOR gate returns “0” if two bits are the same and “1” if they are different. It has the following truth table:

| Input 1 | Input 2 | Output |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

The XOR value does not reveal any information about the two input bits themselves and can be announced publicly. If a given pair of bits in the secret key have the same XOR value, Alice and Bob can both discard for example the second bit in the pair and know that the first bit agrees. However, for low bit error rates this approach is inefficient.

If an experimentally obtained key has some finite error rate, Alice and Bob have to assume that Eve intercepted and retransmitted at least some fraction of the photons, since this is the worst-case scenario. It is then in their interest to minimize the amount of information that Eve could have obtained about the key. This process is called privacy amplification and is a way of distilling the key material from a partially compromised key into a more secure key. In the simplest case, this can also be done using an XOR operation, but in a slightly different way to the key reconciliation.

To do the privacy amplification, Alice randomly picks two bits in the key and replaces them with their XOR value. She then announces the indices of the bits she replaced, but not the XOR value itself. Bob then performs the same operation on his key. If Eve knew one bit in the key, then after this step she on average knows ½ bits.

Steps of the challenge:

1. Implement key reconciliation in QKD application. Use a hash function to allow Alice and Bob to verify that their keys are identical after this step.
2. Estimate the QBER and perform an appropriate level of privacy reconciliation to decrease the amount of information leaked to an eavesdropper.
3. Public channels in QKD need to be authenticated. Add this as a layer on top of the classical communication in the protocol.
4. The XOR-based methods for key reconciliation and privacy amplification described above are quite inefficient. Try to implement more sophisticated methods, such as the key reconciliation method in [3] or the Toeplitz matrix hashing method for privacy amplification [4]

[1] https://github.com/QuTech-Delft/squidasm/tree/develop/examples/applications/qkd
[2] https://arxiv.org/pdf/quant-ph/0101098.pdf
[3] https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=cd81f9ed8a3adfecf6a69fdf5e1dbf92d7844dcc
[4] https://iopscience.iop.org/article/10.1088/1742-6596/741/1/012081/pdf

## Quantum leader election

An important task in distributed systems is that of leader election. This amounts to deciding for example which of a set of threads, processes or nodes gets to modify certain data, assign work to the other members of the system and so on. Another example of leader election is in blockchain, where the task of electing a leader is equivalent to deciding who gets to create the new block [1].

In some scenarios, the parties attempting to elect a leader might not trust each other and want to agree on a leader in a fair way. That is, without an individual, or group of individuals, being able to bias the outcome of the election. While this is impossible to do using just classical information processing, there exists quantum protocols for this task [2].

Quantum leader election is based on a protocol call weak coin flipping [3]. In this scenario, two parties Alice and Bob want to agree on a random outcome, 0 or 1, without trusting each other. It can be thought of as if the coin is 0 then Alice is elected, and if the coin is 1 Bob is elected. The protocol is said to be unbiased if a player attempting to cheat still cannot be elected with a probability more than ½ + ε, where ε is some number that can be made arbitrarily small.

For this challenge, however, you will implement a simpler version of a coin flipping protocol, that does have a finite bias but is nevertheless better than any classical protocol [4]. For details on this protocol, please consult Section 2 of Ref. [5], where it is described in a detailed step by step manner.

Note that the protocol requires Alice to prepare a qutrit state – that is, a quantum state of dimension three (that can take on values {0,1,2} instead of just {0,1} in the case of a qubit).

The steps of the challenge are the following:

1. Find a way to encode a qutrit using qubits in SquidASM (for example using the symmetric subspace of two-qubit states).
2. Implement a circuit for Alice to prepare the state $|\psi_a\rangle$ as defined in Ref. [5].
3. Send the qutrit part of the state to Bob using quantum teleportation.
4. Implement the classical rounds of communication, simulate the protocol and check that it is fair (gives an equal winning probability) when neither player cheats.
5. Try to think of a way for Alice or Bob to cheat, implement this strategy and simulate the winning probability they can achieve (see Ref. [5] for an example of a cheating strategy).

[1] https://eprint.iacr.org/2022/993.pdf
[2] 0910.4952.pdf (arxiv.org)
[3] Coin Flipping - Quantum Protocol Zoo (veriqloud.fr)
[4] 1811.02984.pdf (arxiv.org)
[5] arXiv:quant-ph/0206121v2 12 Jul 2002