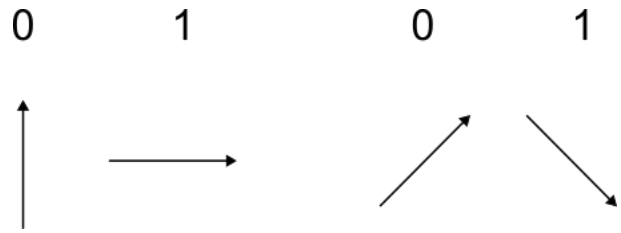
Extending QKD implementation

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 chooses to encode a bit (0 or 1) in the polarisation of the each photon, and additionally randomly chooses 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 chooses 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 chooses 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