UNIVERSITY OF CALIFORNIA

Los Angeles

Quantum Key Distribution

A report submitted in partial satisfaction of the requirements for the degree

Master of Science in Electrical and Computer Engineering

Jayanth Shreekumar

Acknowledgements

Foremost, I would like to express my sincere gratitude to Professor Lara Dolecek for her unwavering support of my study and research throughout my time at the Laboratory for Robust Information Systems (LORIS)., where I had the opportunity to conduct cutting edge research. I thank her for her guidance and I learnt a lot from her experience and expertise.

I would also like to thank Debarnab Mitra and Lev Tauz for collaborating with me and guiding my work. They were supportive throughout my time at LORIS and I benefited from the useful discussions we had on a weekly basis. I learnt a lot from observing them conduct research and picked up important skills, both technical and non-technical, from them.

Abstract

Quantum key distribution is a way for two users to communicate with each other securely and involves several features from quantum mechanics. Time entanglement QKD is popular due to its ability to generate high key rates while maintaining a high degree of privacy. In this project, I worked on several aspects of the QKD problem, which involved working with the multi-layered coding scheme, finding the best set of mappings to obtain the highest key rate, incorporating the progressive edge growth (PEG) algorithm to obtain LPDC codes with large girth, and modelling the QKD channel. We show that experimentally that using binary and gray mappings are not a bad choice and that the QKD channel can be modelled as a mixture of two Gaussians and a uniform distribution by comparing key rates obtained for the empirical channel with those obtained for the parameterized channel.

Introduction

Low Density Parity Check (LDPC) Codes

Low Density Parity Check (LDPC) codes are a class of linear block codes defined in terms of a structured sparse parity check matrix H. They were discovered by Gallagher in 1962[1], and rediscovered by McKay in 1996[2]. They offer a good trade-off between their error correction capabilities and their complexity in practical implementation. In the limit of long codeword lengths, binary LDPC codes can perform decoding close to the the Shannon limit (their capacity), using the message passing algorithm. Binary LDPC codes have been widely explored and documented.

Non-binary LDPC (NB-LDPC) codes are the general versions of LDPC codes that extend the concept of binary LDPC codes over finite fields GF(q) (or GF(2^q)[3]. The parity check matrix H of NB-LDPC codes is composed of the elements of a pre-defined Galois finite field GF(q). Clearly, binary-LDPC codes are special cases where $q = 2$. NB-LDPC codes are expected to perform much better than binary codes as they have more symbols as opposed to just 0 and 1, and can achieve much higher rates while avoiding short cycles in the bipartite graph and keeping the weights of codewords large.

However, the drawback is the increased complexity of implementing the decoding aspect: the message passing algorithm for the binary case where we simply flip bits (bit flipping algorithm) is no longer sufficient. Belief propagation is a general version of the message passing algorithm. It is built on an underlying graph called the factor graph or Tanner graph, which is the graphical representation of the parity check matrix H. The complexity of decoding of LDPC codes depend on the number of edges in the graph, as well as the number of bits sent between variable (data) nodes and check nodes. For NB-LDPC codes, complexity scales exponentially in the form a^{t_r} where t_r is the maximum number of non-zero entries per row in the parity check matrix.

Quantum Key Distribution

Time entanglement quantum key distribution [4, 5] is a specific way to generate entangled photon streams and send one of each pair to Alice and Bob securely. Quantum channels cannot be measured due to the property that the act of measuring the transmitted bits disturbs them and thus alerts them to external interference. In time entanglement QKD, both Alice and Bob receive a single photon from each pair, and they convert these photons into a sequence of symbols depending on their arrival times as shown in Fig. 1. The times of Alice are Bob, which are assumed to be synchronized, is divided up into intervals of time

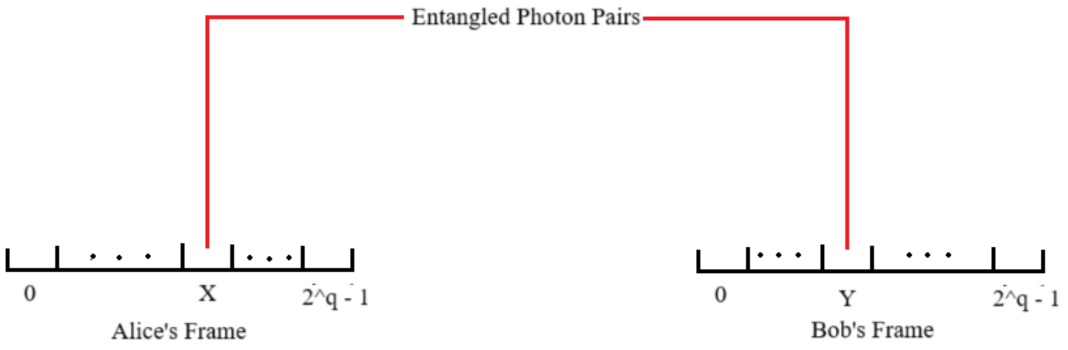


Figure 1: Key generation using time binning in Quantum key distribution

called frames. Each frame is in turn divided into $2^q - 1$ parts, called bins, with each bin assigned a length of time called binwidth, and the arriving photon is assumed to fall into one of these bins. These bins, which are numbered from 0 to $2^q - 1$, act as the symbol that Alice or Bob received based on which bin the photon was received in. To make synchronization easier, Alice and Bob both discard frames in which either one of them receive no photons or more than one photons, and only keep the frames in which both of them receive exactly one photon.

Ideally, the photons occupy the same bins for both Alice and Bob and thus they receive identical messages. But due to practical issues, these sequences are not necessarily identical. There are two major problems that are faced when dealing with the time entanglement QKD protocol: timing jitters, and mismatched photons. Timing jitters refer to the fact that entangled photons are not completely in sync and therefore, they occupy bins that are usually adjacent, but are not exactly the same, thereby causing mismatch. The other problem, called mismatched photon pairs, refers to a scenario where two photons that are from different entangled pairs occupy the same frame in both Alice and Bob but not the same bin. Since the photons are not entangled, they can occupy any pairs of bins in the users. Finally, the result of this time binning protocol is that we observe a sequence of pairs of symbols $(\mathbf{X}, \mathbf{Y})$ where $\mathbf{X} \in \mathbb{F}^N$ is a sequence of N symbols called a block that Alice observes, with each symbol in the Galois field of size 2^q, and $\mathbf{Y} \in \mathbb{F}^N$ is a sequence of N symbols that Bob observes, with each symbol in the same Galois field size.

The goal of secret key distribution protocols is to share a sequence of symbols between two individuals, Alice and Bob, through a standard classical communication channel that is being intercepted and listened to by an eavesdropper, Eve. This shared sequence, called a key, consists of a bits that are ideally meaningless to Eve, but are known to Alice and Bob. Therefore, Alice and Bob can communicate over the classical channel to reconcile their differences, while simultaneously not giving away much information to Eve. This step of the post-processing phase is called the information reconciliation (IR) step. Since the classical channel is not private, the shared key is not private, and to compensate for this, Alice and Bob perform the privacy amplification step to re-establish secrecy.

Finally, after information reconciliation, a new issue has to be addressed. Eve, who has been eavesdropping on the classical channel, has had access to the message exchanged between Alice and Bob, and therefore, the original sequence that Alice had may have been exposed during the IR stage where Bob utilized the information sent by Alice to estimate her sequence. Therefore, a necessary step, called the privacy amplification step, has to be performed in which both Alice and Bob extract secret keys S and $\hat{S}$ from their messages X and $\hat{X}$ by compressing them. Clearly, if $X = \hat{X}$, then $S = \hat{S}$. As standard techniques for privacy amplification already exist, in this project, I focused on the information reconciliation step.

Preliminaries

Channel Coding for Information Reconciliation

Channel coding methods are used for correcting errors that occur during the transmission of data over channels and recover the original message that was sent at the receiver. Thus, they lend themselves comfortably to the QKD problem for the information reconciliation phase. The IR phase is performed using non-binary Low density parity check (NB-LDPC) codes which are defined using a parity check matrix $H \in \mathbb{F}^{M \times N}$, where each element is from the pre-decided Galois field $\mathbb{F}$. Alice encodes her information by performing $\mathbf{R} = H\mathbf{X}$, and transmits this encoded message $\mathbf{R}$ over the public channel to Bob. Bob utilizes $\mathbf{R}$ as well as his own message $\mathbf{Y}$ to perform decoding and obtain $\hat{\mathbf{X}}$, his estimate of $\mathbf{X}$ which is Alice's message. This technique of utilizing side information $\mathbf{Y}$ to perform LDPC decoding of the message $\mathbf{R}$ is called the Slepian-Wolf coding scheme [6, 7]. To measure the performance of these techniques, the information reconciliation rate (IR rate) r is used:

$$r = q(1-E)\frac{N-M}{N}$$

where q is the number of bits used per symbol, E is the frame error rate, and $\frac{N-M}{N}$ is the code rate of H.

However, the utilization of NB-LDPC codes to higher order Galois fields is not scalable due to increased complexity of decoding for large alphabets of size 2^q (it scales as log-linear in field size [3]). Therefore, the utilization of a layered scheme for decoding is necessary as shown in [7]. In this multi-layered coding scheme, the encoding of a symbol $X \in \mathbb{F}$ is performed by mapping it to a binary sequence as follows: for each symbol $X = \{0, 1, ..., 2^q - 1\} \in \mathbf{X}$ in the sequence, assign it a binary sequence of k bits, denoted by $[X_1, X_2, ..., X_k]$, and then transmit them over the channel after performing the encoding $\mathbf{R}_j = H\mathbf{X}_j$ where $\mathbf{X}_j = [X_j^1, X_j^2, ..., X_j^N]$. Thus Bob receives $\mathbf{R} = [R^1, R^2, ..., R^M]$ where each R^i is made up of bits given by $R^i = [R_1^i, R_2^i, ..., R_k^i]$, and performs binary LDPC Slepian-Wolf decoding on these bits layer-wise to recover $\hat{\mathbf{X}}_j = [X_j^1, X_j^2, ..., X_j^N]$ for all j and thus also recover $\hat{\mathbf{X}}$.

Generalizing this layered coding scheme, we can also control the number of bits that contribute to every layer. Instead of performing a binary mapping and taking each bit for its own layer, we can split each symbol X into $b + 1$ symbols as follows. Let $ab + r = q$ where $b = \lfloor \frac{q}{a} \rfloor$ and r is the remainder. Then, we can always map a symbol that has q bits into symbols from a smaller Galois field $\mathbb{F}(2^a)$ where each symbol is made up of a binary sequence of a bits. As an example. the symbol 13 in $\mathbb{F}(2^4)$ with binary sequence 1101 can be split into two symbols $3 \in \mathbb{F}(2^2)$ with binary sequence 11 and $1 \in \mathbb{F}(2^2)$ with binary sequence

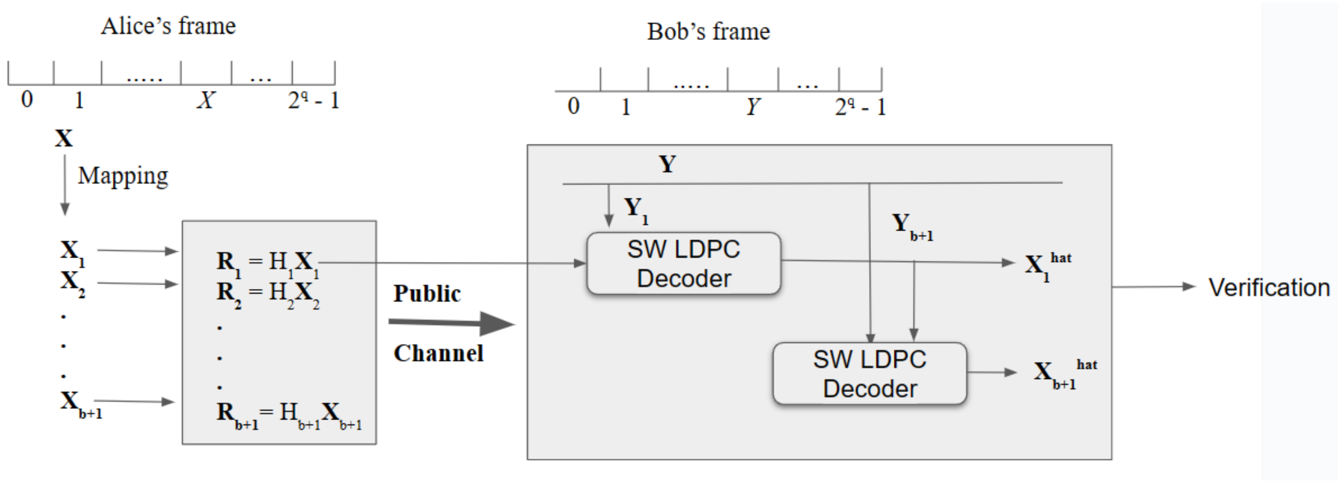


Figure 2: Non Binary MLC Protocol with Slepian Wolf Coding Scheme

01. Another example: the symbol $26 \in \mathbb{F}(2^5)$ that has the binary sequence 11010 can be split into three symbols $3 \in \mathbb{F}(2^2)$ with binary sequence 11, $1 \in \mathbb{F}(2^2)$ with binary sequence 01, and $0 \in \mathbb{F}(2^1)$ with binary sequence 0.

This can be written generally as converting every symbol $X \in \mathbf{X}$ into a sequence of symbols $[X_1, X_2, ..., X_b, X_{b+1}]$ where the first b symbols belong to the Galois field $\mathbb{F}(2^a)$ and the final symbol, if there is a reminder present, belongs to the Galois field $\mathbb{F}(2^r)$. Therefore, we can now split $\mathbf{X} \in \mathbb{F}(2^q)^N$ into layers $[\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_{b+1}]$ where $\mathbf{X}_i \in \mathbb{F}(2^a)^N, 1 \leq i \leq b$ and $\mathbf{X}_{b+1} \in \mathbb{F}(2^r)^N$. To correspond to this, we also use parity check matrices whose elements belong to the same Galois field, i.e., $H_i \in \mathbb{F}(2^a)^{m_i \times N}, 1 \leq i \leq b$ and $H_{b+1} \in \mathbb{F}(2^r)^{m_{b+1} \times N}$. Then Alice sends the message $\mathbf{R} = [\mathbf{R}_1, \mathbf{R}_2, ..., \mathbf{R}_{b+1}]$ over to Bob who then performs layer-wise decoding to obtain $[\hat{\mathbf{X}}_1, \hat{\mathbf{X}}_2, ..., \hat{\mathbf{X}}_{b+1}]$ and thus recovers $\hat{\mathbf{X}}$. Using this new NB-LDPC MLC coding scheme, the total IR rate is calculated as the sum of the IR rates per layer, and is given by:

$$r = \sum_{i=1}^{b} a(1 - E_i)\frac{N - m_i}{N} + r(1 - E_{b+1})\frac{N - m_{b+1}}{N}$$

where E_i are the frame error rates for layer i. Clearly, the total IR rate depends on the rates $\frac{N-m_i}{N}$ used for every layer in the MLC scheme.

Public Channel Model

The public channel can be modelled as the a conditional probability of Bob receiving a symbol Y given that Alice transmitted a symbol X where $X, Y \in \{0, 1, 2^q - 1\}$ in a Galois field of size 2^q. As observed from the QKD dataset that I worked on, the public channel could be modelled as a combination of local and global errors. Local errors are caused due to jitters, while global errors are caused due to mismatched photons as described previously. Experimental results and previous work [8, 9] show that this public channel could be modeled as a mixture of two Gaussians and a uniform distribution, given by:

$$P_{Y|X}(y|x) = c\left(e^{\frac{(y-x-\mu_1)^2}{2\sigma_1^2}} + \alpha e^{(\frac{y-x-\mu_2)^2}{2\sigma_2^2}}\right) + \beta$$

where $x, y \in \mathbb{F}(2^q)$ and α is the scaling parameter for the second Gaussian and β is the uniform distribution.

Therefore, using the standard AWGN or BSC channels are not the best ones to model the public channel for the QKD problem, and utilizing this model will be helpful for LDPC codes in performing decoding.

Non-Binary LDPC Code Decoding

NB-LDPC codes are defined using a sparse parity check matrix $H \in \mathbb{F}^{M \times N}$, where each element in H is taken from the Galois field of size $\mathbb{F}(2^q)$. Equivalently, they can be represented using a bipartite Tanner graph in which each row represents a check node and each column represents a variable node. Edge connections between these variable nodes and check nodes are simply made based on whether the corresponding position in H is non-zero. If it is non-zero, an edge is created and is given the corresponding weight in H. The rate of the code is simply given by $\frac{dv}{dc}$ if the code is uniform where dv is the number of edges emanating from each variable node and dc is the number of edges emanating from each check node. If not, the rate of the code is calculated by utilizing the variable and check node degree distributions.

The general decoding algorithm for LDPC codes is called the belief propagation (BP) algorithm or the sum-product algorithm (SP) [10]. For binary LDPC codes, messages passed in the SP algorithm are simply a single probability value. However, since there are 2^q elements in the Galois field in an NB-LDPC code, we must use a vector of length 2^q to store the probability of each symbol for every variable node. The initialization of the probabilities at variable node n is given by:

$$\gamma_n^{(0)}(g) = P(X_n = g|Y = y_n)$$

where the subscript (0) represents the current iteration step and the channel probabilities from the public channel model described above is used.

At iteration (k), the check node processing step in which check nodes aggregate information and send messages is as follows. For each check node m and each variable node $n \in S_v(m)$ which connects to check node m, perform the following for every position $\beta \in 2^q$ in the LLR vector to compute the next message:

$$c_{m,n}^{(k)}(\beta) = \sum_{a_j \in \mathcal{L}(m|a_n=\beta)} \left(\prod_{j \in S_v(m) \setminus n} v_{m,j}^{(k-1)}(a_j) \right)$$

where $S_v(m)$ is the set of all variable nodes connected to check m and $\mathcal{L}(m|a_n = \beta)$ is the set of sequences of finite field elements $(a_j)(j \in S_v(m) \setminus n)$ such that they satisfy the parity check equation $\sum_{j \in S_v(m) \setminus n} h_{m,j} a_j = h_{m,n}\beta$.

The variable node update for a variable node n, each $m \in S_c(n)$ for every symbol $\beta \in \mathbb{F}(2^q)$ is given as:

$$v_{m,n}^{(k)}(\beta) = \gamma_n(\beta) \prod_{i \in S_c(n) \setminus m} c_{i,n}^{(k)}(\beta)$$

where all the messages that is received, apart from the check node that it is sending the message to, is multiplied. Finally, aposteriori information is computed using:

$$z_n^{(k)} = argmax_\beta \left(\gamma_n(\beta) \prod_{i \in S_c(n)} v_{i,n}^{(k)}(\beta) \right)$$

If $z_n^{(k)} H^T = 0$, then decoding is successful as a codeword has been found, otherwise continue decoding until a pre-set maximum number of iterations is achieved, at which point decoding is declared unsuccessful. A minor change in the implementation that I worked with is that instead of passing around probabilities, LLRs are computed and the messages are also computed using these LLRs. This helps reduce quantization errors as multiplications are converted to additions.

Contributions

NB-LDPC codes using Progressive Edge Growth Construction

NB-LDPC codes are powerful codes because they are capacity approaching, but easily lend themselves to low complexity iterative decoding for optimal decoding. While random graphs have been used to construct high performance NB-LDPC codes, they only rely on the sparsity of the parity check matrix to avoid short cycles of length 4 or 6 which are highly detrimental to the decoding process. Therefore, while random LDPC codes designed for long block lengths work fairly well due to the large size and sparsity of the parity check matrix, it becomes much less effective for shorter block lengths which are much more prone to graphs with smaller girth. The Progressive Edge Growth (PEG) algorithm [11] is a deterministic algorithm that works to construct large girth graphs by placing edges iteratively as best as possible by maximizing the local girth of the sub-graph at that stage. One of my contributions was incorporating the PEG construction for LDPC codes which is open-source in https://github.com/Lcrypto/classic-PEG-/tree/master/classic_PEG/.

The PEG algorithm works on a Tanner graph that has been unfolded in a breath-first way. Let the vertices in the graph be given by $V = V_c \cap V_s$ where V_s is the set of all variable nodes and V_c is the set of all check nodes. Let $D_s = \{d_{s_0}, d_{s_1}, d_{s_2}, ..., d_{s_{n-1}}\}$ be the set of degrees for the variable nodes and let $D_c = \{d_{c_0}, d_{c_1}, d_{c_2}, ..., d_{c_{n-1}}\}$ be the set of degrees for check nodes that form their respective degree distributions. Without loss of generality, consider a graph that is being constructed and it already has some edges placed in it. For every variable node s_j, we can find its neighbourhood within depth l by traversing the edges and creating a tree of depth l. Let $\mathcal{N}^l_{s_j}$ be the set of all check nodes that are reached by s_j and let $\tilde{\mathcal{N}}^l_{s_j}$ be all the unreached nodes so that it forms a partition with $\mathcal{N}^l_{s_j}$ over all check nodes. Clearly, there may be duplicate check nodes due to cycles. The algorithm iterates over all variable nodes and over all degrees of variable nodes and tries to place an edge between the variable node in question and the farthest check node. There are two cases that need to be addressed:

1. Not all check nodes can be reached by the variable node i.e $\tilde{\mathcal{N}}^l_{s_j}$ is not empty. In this case, we can simply select a check node from this set and connect an edge to it. This avoids the creation of any cycle. Note that this happens frequently at the start of the algorithm.

2. All check nodes can be reached by the variable node. In this case, the algorithm chooses a check node that is at the largest possible depth from the current variable node while also simultaneously not violating the check node degree distribution. If this depth is given by $d + 1$, then the largest possible cycle at this point is given by $2(d + 2)$, If there are multiple choices, choose the check node that has the smallest degree to maintain uniform check node degree distribution.

Mappings

In the NB-LDPC MLC decoding scheme discussed previously, the binary representation of a symbol was used to convert a symbol in $\mathbb{F}(2^q)$ into a binary string, and this was used later to create several symbols from a smaller field $\mathbb{F}(2^a)$. The binary representation of a symbol is the simplest mapping to consider because it is straightforward to map a symbol to a binary string - simply find the symbol's binary representation: $GF(2^q) \rightarrow GF(2)^q$. Gray mapping is another such mapping in which two successive symbols in a field have binary strings that only differ in one position.

However, is there a better arbitrary mapping that can be used to convert a symbol into a binary string? For a Galois field of size 2^q, there are $2^q!$ number of different permutations that have to be tried out in a brute force approach to find the best mapping, which is clearly infeasible. Hence, a heuristic algorithm to find the best mapping was employed. Simulated annealing [12] is a search algorithm that makes local changes to the state of the system to escape the local optima and approximate the global optima in discrete optimization problems.

We use a modified version of simulated annealing for our problem. The different possible permutations of mappings in this problem correspond to the states of the system and local search is performed by swapping the binary representation of any two symbols in a mapping to find a neighbour. The entire modified SA algorithm is presented in algorithm 0. The input to the algorithm are:

1. S_{init}: This is the initial mapping that is used as the baseline to start the algorithm. It is typically the straightforward binary mapping.

2. q: This is the number of bits used to represent a symbol. For a Galois field of size 2^q, the number of bits required to represent it is q.

3. f_{th}: Threshold value that accounts for noise and also helps to decide if the IR rate of the new mapping is better than that of the previous one.

4. f_{de}: Normalizes the exploration formula so that it always yields a value in $[0, 1]$.

5. $iter$: number of iterations to perform the simulated annealing algorithm.

The algorithm starts off by finding the IR rate of an initial mapping, which is typically binary or gray. For the second epoch, any two binary representations are swapped, taking care not to repeat any permutations that were computed previously by using a set to keep track as shown in 8. Once a mapping whose IR rate has not yet been established is found, its IR rate is found and the difference between this value and the IR rate of the previous mapping is calculated as shown on line 14. If this difference is a value greater than the user input value f_{th} which accounts for random noise, then this mapping is objectively better than the previous one, and it is always used to generate a new mapping in the next epoch. If the difference is lesser than f_{th}, then only a fraction of these inferior mappings are used to find the next mapping in the hope of breaking out of the local optima through exploration as shown by the condition in line 22. The output of the algorithm is the best mapping and its corresponding IR rate.

Algorithm 1 Simulated Annealing Algorithm

```
1: procedure SIMULATED ANNEALING($S_{init}, q, f_{th}, f_{de}, iter$)    ▷ To find the approx. global optimum mapping
2:     $S \leftarrow S_{init}$                                  ▷ Initial mapping (typically binary or gray)
3:     $K \leftarrow 0$
4:     $V \leftarrow \varnothing$
5:     for $T \leftarrow iter$ to 0 do                          ▷ loop over epoch
6:         if $T \neq iter$ then                                ▷ If this is not the 1st epoch
7:             $S_{new} \leftarrow$ SWAP($S, q$)                ▷ Find new mapping permutation
8:             while $S_{new} \in V$ do
9:                 $S_{new} \leftarrow$ SWAP($S_{new}, q$)
10:            end while
11:        end if
12:        $V.insert(S_{new})$                                  ▷ To make sure we do not repeat mappings
13:        $K_{new} \leftarrow$ GET IR RATE($S_{new}$)
14:        $\Delta K = K_{new} - K$
15:        if $\Delta K \geq f_{th}$ then                       ▷ Better mapping found than current one
16:            $K \leftarrow K_{new}$
17:            $S \leftarrow S_{new}$
18:            if $K_{new} > K_{best}$ then                     ▷ Best mapping at this point found
19:                $K_{best} = K_{new}$
20:                $S_{best} = S_{new}$
21:            end if
22:        else if $e^{\frac{\Delta K - f_{th}}{T \times f_{de}}} > rand(0,1)$ then    ▷ Exploration
23:            $K \leftarrow K_{new}$
24:            $S \leftarrow S_{new}$
25:        end if
26:    end for
27:    return $S_{best}, K_{best}$                              ▷ Return the best mapping and its IR rate
28: end procedure
29:
30: procedure SWAP($S, q$)                                      ▷ Find neighbour of current mapping state
31:    $r1 \leftarrow rand(0, 2^q)$
32:    $r2 \leftarrow rand(0, 2^q)$
33:    while $r2 = r1$ do
34:        $r2 \leftarrow rand(0, 2^q)$
35:    end while
36:    $S[r1], S[r2] = S[r2], S[r1]$
37:    return $S$
38: end procedure
```

QKD Channel Modelling

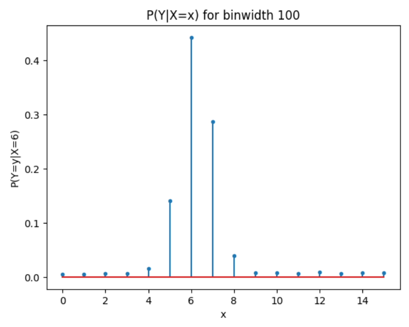
So far, QKD experiments relied on working with standard channels such as the AWGN channel or the BSC channel for the public channel model. As stated before, these channels are not good approximations for the QKD public channel. A slight improvement is using empirical probabilities $P(Y|X)$ created from a dataset of symbols that Alice and Bob received at their end when they performed time binning. However, there are two drawbacks for this approach. Firstly, the channel probabilities depend on a specific dataset that can change slightly on average. Also, not having a general model limits experiments to only those people who have access to the dataset. Secondly, the dataset in question had bins of width ranging from $100ps$ to $1000ps$ in steps of $100ps$ and there was no easy way to work with other binwidths, like $250ps$. Therefore, having a general model that can be used to extrapolate to different binwidths as required, will be very useful. Further, it can also be used more widely as the QKD channel can then be created locally to work with.

For a Galois field $\mathbb{F}(2^q)$, each symbol that Alice and Bob receive is in the set $\{0, 1, ..., 2^q - 1\}$. The channel conditional probability $P(Y|X = x)$ where X denotes Alice and Y denotes Bob is then a PMF of length 2^q and sums up to 1, and this is true for every $X \in \{0, 1, 2, ..., 2^q - 1\}$. To model the QKD channel was a two-step process. Using a discriminative model that involves fitting a curve to the 2^q probability values in $P(Y|X = x)$ is easier to perform, but it does not work well as it simply fits a linear regression curve to the points without utilizing any information of the underlying probability distribution. A better approach is to use a generative modelling approach where the idea is to create a probability distribution that models the empirical probabilities as closely as possible. I used the generative modelling approach, where I created a continuous distribution whose inputs were the parameters of the distribution and then discretized it to create a corresponding discrete probability distribution. Discretization was based on the idea that for any pre-defined binwidth b, a photon that falls into a bin is converted to the same symbol s as long as it is in the range $[0, b)$. Once the discrete distribution was found, the parameters of the continuous distribution were tweaked to obtain a discrete distribution that was as close to the empirical distribution as possible.

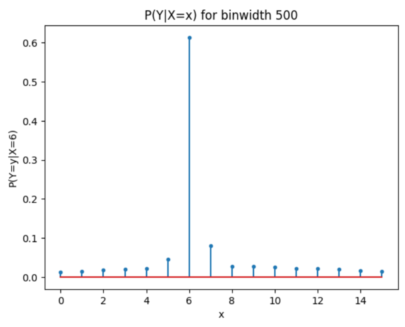
I used the mean absolute distance to calculate the error between the empirical PMF $P(Y|X = x)$ obtained from the dataset and the generated PMF $P(\tilde{Y}|X = x)$. Since $Y \in \{0, 1, 2, ..., 2^q - 1\}$, the average of the mean absolute differences $\tilde{M}$ over all values of Y is calculated as:

$$\tilde{M} = \frac{1}{2^q} \sum_{y=0}^{2^q-1} |P(\hat{Y} = y|X = x, A) - P(Y = y|X = x)|$$

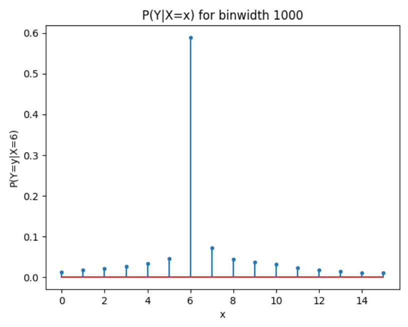
where A is the set of parameters of the distribution that was used to generate the continuous distribution. I worked with the average of the mean absolute differences rather than working with 2^q different sets of parameters, one for each $P(Y|X = x)$, for a particular binwidth. This would have added more complexity as I would need to find the function that related the 2^q sets of parameters in a given $P(Y = y|X)$ corresponding to a binwidth before finding the relationship between parameters across binwidths, which



(a) $P(Y|X = 6)$ for Binwidth 100 ps



(b) $P(Y|X = 6)$ for Binwidth 500 ps



(c) $P(Y|X = 6)$ for Binwidth 1000 ps

Figure 3: Sample PMFs $P(Y|X = 6)$ for binwidth 100 ps, 500 ps, and 1000 ps

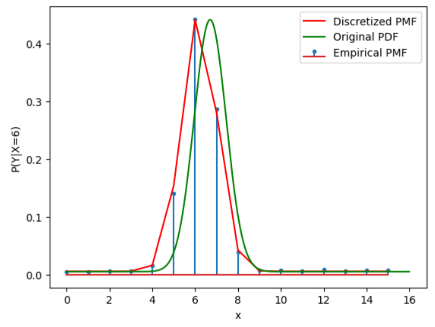
would have been a complex and unwieldy model to work with due to the large number of parameters involved.

The figure above shows $P(Y|X = 6)$ for 3 different binwidths - 100 ps, 500 ps, and 1000 ps binwidths for a Galois field of size 2^4. We see that the PMF for binwidths 1000 and 500 are much more concentrated at $Y = 6$ when compared to that of binwidth 100. This is as expected because the larger the binwidth, the lesser the chance of a photon occupying an adjacent bin in error i.e., the probability that a photon is delayed longer than the width of the bin is much smaller. Also, we see that the PMF for binwith 100 has a much larger variance, which results from the fact the even slight delays in photon arrival times caused it to be put into an adjacent bin, therefore leading to errors. A similar distribution in the PMF $P(Y|X = x)$ can be seen for all different $x \in \{1, 2, ..., 15\}$. We also observe that the PMF is not completely symmetric: it is slightly skewed to the right for all binwidths. This is another reason why a generative modelling approach is superior to a discriminative approach. The generative modelling approach can easily model skewed distributions such as this one by simply shifting the mean and the performing discretization.

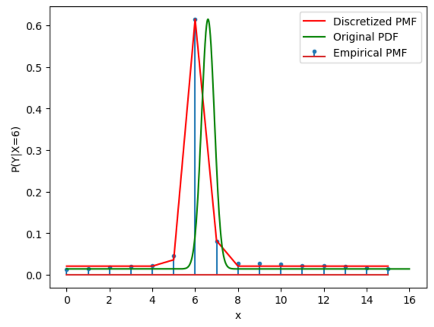
I worked with several different probability distributions to model the above empirical distribution.A reasonable starting point to modelling is the basic assumption that the distribution is a combination of a uniform distribution, that models system noise, and another distribution (or a combination of distributions) that models photon jitter. To find the best parameters for any model, I performed a broad grid search to find the parameters that modelled the PMF as closely as possible. I used the mean absolute difference metric as stated above to compare different distributions. Once this was established, I performed a finer grid search over a smaller space around this set of parameters to fine tune the model.

The figure below in 4 shows the results of modelling using a uniform distribution for noise and a single Gaussian distribution for photon jitter. We see that the discretized model fits pretty well to the empirical distribution at binwidth 100, but fails to do so for the others. For binwidth 1000, we clearly see that the discretized model overshoots the empirical distribution at $Y = 6$ and undershoots for Y around 6. The parameters for the underlying PDFs are as follows: Binwidth 100 ps $(\mu, \sigma, \beta) = (0.7, 0.75, 0.011)$, Binwidth 500 ps $(\mu, \sigma, \beta) = (0.6, 0.3, 0.0155)$, Binwidth 1000 ps $(\mu, \sigma, \beta) = (0.55, 0.3, 0.0166)$. Here, μ is the mean of the underlying PDF, shifted from the value that Y took, in this case, 6, σ is the standard deviation, and

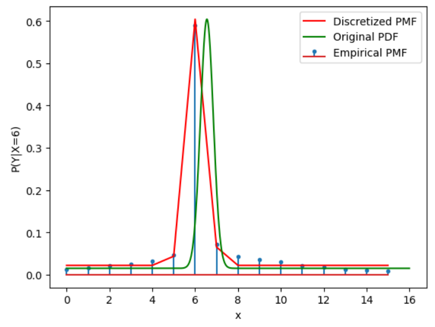
β is the magnitude of the uniform distribution.



(a) $P(Y|X = 6)$ for Binwidth 100 ps

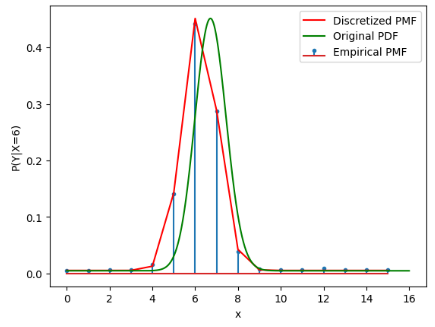


(b) $P(Y|X = 6)$ for Binwidth 500 ps

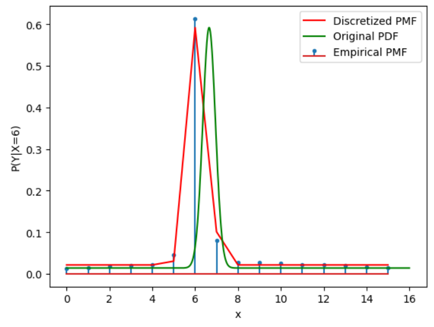


(c) $P(Y|X = 6)$ for Binwidth 1000 ps

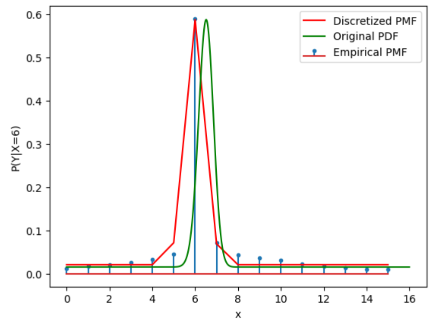
Figure 4: Sample PMFs $P(Y|X = 6)$ for binwidth 100 ps, 500 ps, and 1000 ps along with a discretized model fit to the empirical PMF (in red) and the underlying continuous distribution (in green). The underlying distribution is made up of a single Gaussian along with a uniform distribution.



(a) $P(Y|X = 6)$ for Binwidth 100 ps



(b) $P(Y|X = 6)$ for Binwidth 500 ps

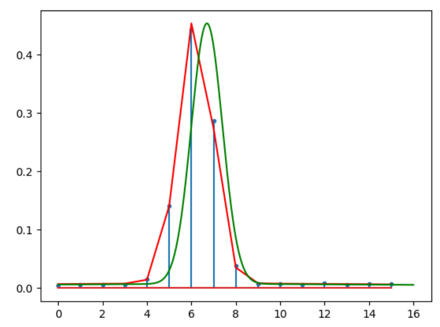


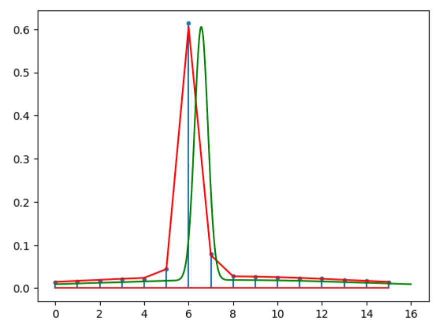
(c) $P(Y|X = 6)$ for Binwidth 1000 ps

Figure 5: Sample PMFs $P(Y|X = 6)$ for binwidth 100 ps, 500 ps, and 1000 ps along with a discretized model fit to the empirical PMF (in red) and the underlying continuous distribution (in green). The underlying distribution is made up of a single skewed Gaussian along with a uniform distribution.

To account for the modelling errors described above, I tried to fit a skewed normal distribution + uniform distribution, the results of which are shown above in figure 5. The parameters for the underlying PDFs are as follows: Binwidth 100 ps $(\mu, \sigma, a, \beta) = (0.4, 0.8, 0.6, 0.011)$, Binwidth 500 ps $(\mu, \sigma, a, \beta) = (.7, 0.3, -0.2, 0.016)$, Binwidth 1000 ps $(\mu, \sigma, a, \beta) = (.7, 0.4, -0.8, 0.021)$. Here, μ is the mean of the underlying Gaussian, shifted from the value that Y took, in this case, 6, σ is the standard deviation, a is the skewness factor of the normal distribution, and β is the magnitude of the uniform distribution. Again, we see that using a skew does not help to alleviate the problems that were present without skew. This is evident in the fact that the best parameter for the skew that was obtained by performing grid search was very small, and the skewness factor did not help much.

We also observe from figure 3 that there is a distinct second Gaussian distribution that can be used to model the low lying distribution for the higher binwidths. The peak can be modeled using a Gaussian with a small standard deviation, and the low lying probabilities can be modelled more effectively using a second Gaussian.


(a) $P(Y|X=6)$ for Binwidth 100 ps


(b) $P(Y|X=6)$ for Binwidth 500 ps

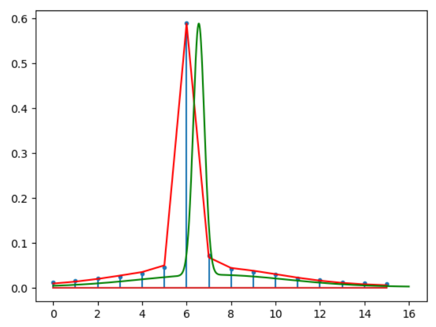

(c) $P(Y|X=6)$ for Binwidth 1000 ps

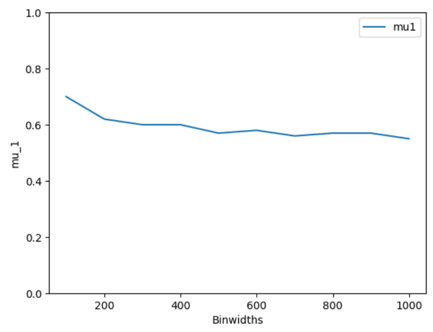
Figure 6: Sample PMFs $P(Y|X=6)$ for binwidth 100 ps, 500 ps, and 1000 ps along with a discretized model fit to the empirical PMF (in red) and the underlying continuous distribution (in green). The underlying distribution is made up of a two Gaussians along with a uniform distribution.

Therefore, the model that I used is given by:

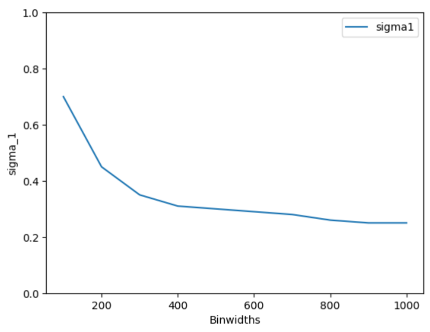
$$P_{Y|X}(y|x) = c\left(e^{\frac{(y-x-\mu_1)^2}{2\sigma_1^2}} + \alpha e^{(\frac{(y-x-\mu_2)^2}{2\sigma_2^2}}\right) + \beta$$

Here, μ_1 is the mean of the first Gaussian, shifted from the value that Y took, in this case, 6, σ is the standard deviation of the first Gaussian, μ_2 is the mean of the second Gaussian, σ_2 is the standard deviation of the second Gaussian, α is the magnitude of the second Gaussian, and β is the magnitude of the uniform distribution. We see that by using a two Gaussian model, the discretized PMF obtained from the underlying PDF closely matches the empirical PMF of the channel.

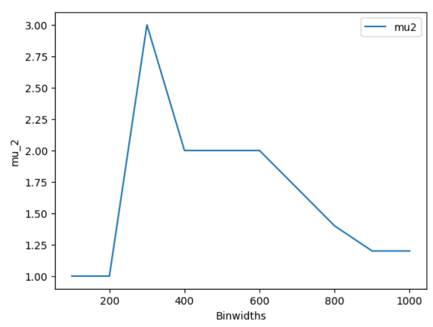
Now that the two Gaussian model has been established, I wanted to find a function that could be employed to generalize the model across binwidths. To do so, I plotted the optimal parameters that I obtained for each binwidth, which is as follows:



(a) Trend in μ_1

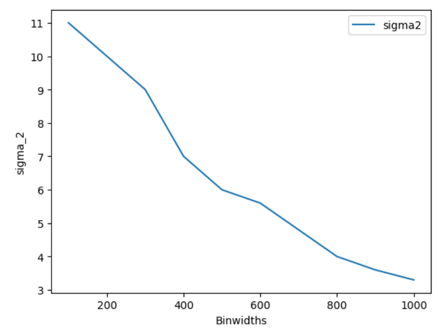


(b) Trend in σ_1

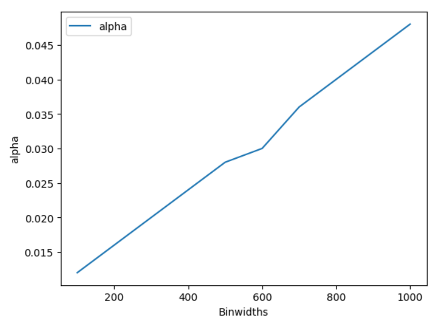


(c) Trend in μ_2

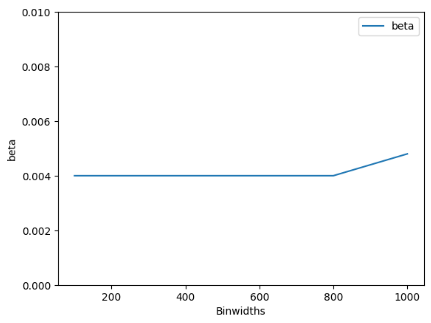
Figure 7: Trends in μ_1, σ_1, and μ_2 for the Double Gaussian model



(a) Trend in σ_2



(b) Trend in α



(c) Trend in β

Figure 8: Trends in σ_2, α, and β for the Double Gaussian model

There are three things of interest in this plot:

1. The plots for μ_1, σ_2, α, and β all have a linear trend.
2. The plot for σ_1 has an exponential trend. This might be due to the fact that the first Gaussian gets thinner as the binwidth increases - the number of errors due to jitter decreases.
3. The plot for μ_2 does not seem to have any trend. However, upon further inspection, I noticed that the parameter μ_2 took a wide range of values while causing only a negligible change to the mean absolute difference. This is because it is the mean of the second Gaussian which is scaled by α. Therefore, I modelled it as a linear trend since it did not have much impact on the overall model by itself.

To obtain a general model for the channel that follows these trends, I performed leave one out cross validation (LOOCV) for each parameter. For example, for μ_1, I performed LOOCV along with a linear model using 9 binwidths for training to obtain the parameters for the remaining binwidth. I performed this to obtain the parameters for each binwidth for every parameter. The results of LOOCV for the 2 Gaussian model are described in the below table in 1. I used these parameters to obtain the key rates of all binwidths for the QKD parameterized model as discussed in the results section.

Binwidth	μ_1	σ_1	μ_2	σ_2	α	β
100	0.616667	0.586812	2.230556	10.105556	0.011778	0.003767
200	0.638065	0.478297	2.025806	9.373387	0.015790	0.003877
300	0.626324	0.355109	1.466912	8.567647	0.019794	0.003962
400	0.611250	0.303912	1.658333	7.868750	0.023792	0.004033
500	0.601149	0.281018	1.629054	6.972973	0.027784	0.004100
600	0.586757	0.270975	1.590541	6.031757	0.032000	0.004168
700	0.576250	0.267032	1.583333	5.145833	0.035750	0.004242
800	0.560662	0.268927	1.598529	4.262500	0.039721	0.004329
900	0.543952	0.270475	1.637903	3.241935	0.043677	0.004310
1000	0.532500	0.270145	1.650000	1.994444	0.047611	0.004178

Table 1: Results of Leave One Out Cross Validation to Obtain the Parameters for each Binwidth when using 2 Gaussians + Uniform

Binwidth	μ	σ	β
100	0.7199252441	0.7199774927	0.01202474252
200	0.6218808582	0.4674111033	0.01306035776
300	0.5809318614	0.3703271629	0.01389997152
400	0.5638291958	0.3330090879	0.01458067922
500	0.5566861351	0.318664401	0.01513255555
600	0.5537027805	0.3131504499	0.01557998327
700	0.5524567594	0.3110309434	0.01594273041
800	0.551936349	0.3102162267	0.0162368237
900	0.5517189956	0.309903058	0.01647525662
1000	0.5516282163	0.3097826791	0.01666856351

Table 2: Results of Leave One Out Cross Validation to Obtain the Parameters for each Binwidth when using a Single Gaussian + Uniform

To validate my reasoning on the single Gaussian being insufficient, I performed the same experiments to find the 3 parameters for each binwidth, the results of which are described above.

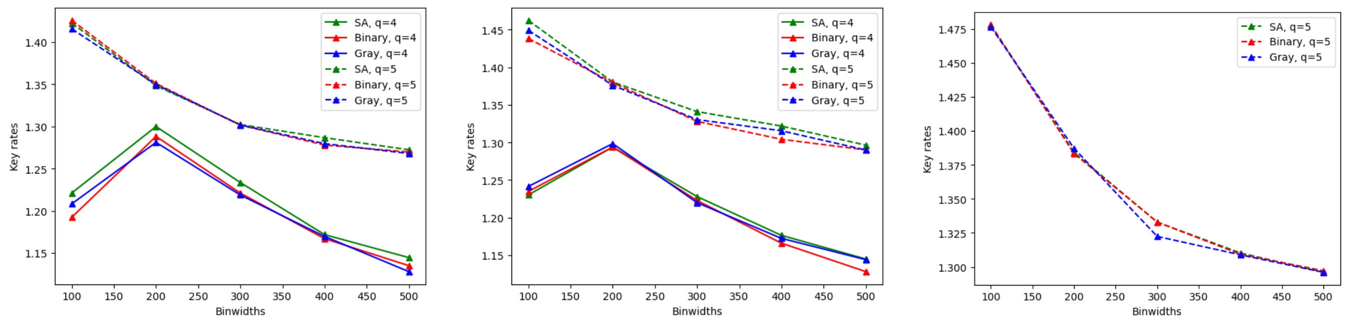
Results

This section covers the simulation results for the experiments conducted above. For all simulations, I have used a regular LDPC code with variable node degree of 3, and a codeword length of 2000. All LDPC codes were constructed using the PEG algorithm [11] as described above. The rate of the code was chosen dynamically with a step size of 0.01 and a maximum rate value of $H(X|Y) + 0.1$ to maximize the key rate for each layer and thus the overall key rate as well. $H(X|Y)$ is the entropy function.

Mappings

This section provides a comparison of the IR rates obtained by using the best mapping obtained using the simulated algorithm to run the LDPC decoding. The parameter a is the NB-MLC protocol parameter that controls the number of layers, or alternatively, the number of bits used per layer in the MLC coding scheme as outlined in the section on channel coding for information reconciliation. For fair comparison, the three different mappings were run on the same sets of codewords and the decoding algorithm was exactly the same. The rates were still dynamically selected based on decoding performance. I tested it on binwidths until binwidth 500.

The results observed are in figure 9. We see that the best mapping obtained using the simulated algorithm performs slightly better for $q = 4, a = 2$, but it does not perform any better for any of the others. Therefore, we conclude that binary and gray mappings are reasonable mappings to use for the NB-MLC scheme for decoding.

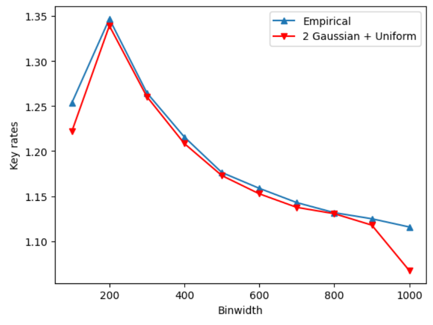


(a) a = 2 (b) a = 3 (c) a = 4

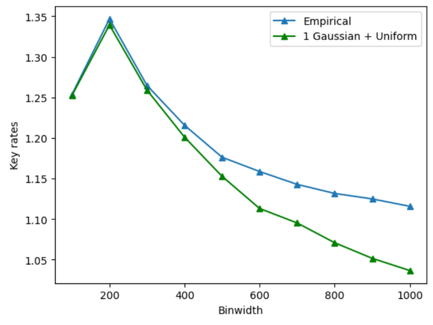
Figure 9: Simulation results of best mapping decoding results obtained using SA search

QKD Channel Modelling

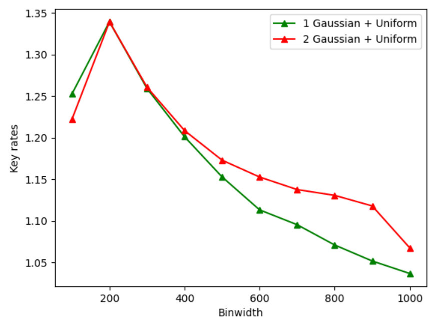
All simulations were performed on a Galois Field of size 2^4 in this section. After finding the parameters of the model using LOOCV as described above, I found the channel probability matrix using these parameters and used this parameterized channel to perform decoding simulations. The results of these simulations are shown in the figure 10. The left panel depicts the key rates that were obtained when using the empirical channel probabilities and compares it with the key rates obtained with the 2 Gaussian + uniform model that has 6 parameters. We see that the parameterized channel performs almost as well as when using the empirical channel probabilities, except for binwidth 1000. The second panel compares the key rates of the empirical channel probabilities with the key rates obtained with the 1 Gaussian + uniform model that has 3 parameters. We see that while this channel performs well for lower binwidths, it performs significantly worse on higher binwidths. This is expected, because the second Gaussian is prominent in the higher binwidths, which cannot be modelled using a single Gaussian model. Finally, the rightmost panel compares the key rate obtained using the two different parameterized models. We see that the 2 Gaussian model significantly outperforms the 1 Gaussian model at all binwidths except for binwidth 100 where it performs slightly worse. I believe this is because binwidth 100 does not require a second Gaussian, and so, the single Gaussian model performs just as well as the 2 Gaussian one.



(a) Empirical vs 2 Gaussian + Uniform



(b) Empirical vs 1 Gaussian + Uniform



(c) 2 Gaussian vs 1 Gaussian

Figure 10: Simulation results of QKD Channel Modelling

Conclusion

In this project, I worked on several aspects of the quantum key distribution problem to find ways to improve the decoding success using LDPC codes. I also worked on finding the optimal QKD public channel that will allow experiments to be performed by the public that do not have access to data from a QKD setup.

Bibliography

[1] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, 1962.

[2] D. J. MacKay and R. M. Neal, "Near shannon limit performance of low density parity check codes," *Electronics letters*, vol. 33, no. 6, pp. 457–458, 1997.

[3] M. Davey and D. MacKay, "Low density parity check codes over gf(q)," in *1998 Information Theory Workshop (Cat. No.98EX131)*, pp. 70–71, 1998.

[4] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," *Theoretical Computer Science*, vol. 560, p. 7–11, Dec. 2014.

[5] T. Zhong, H. Zhou, R. D. Horansky, C. Lee, V. B. Verma, A. E. Lita, A. Restelli, J. C. Bienfang, R. P. Mirin, T. Gerrits, *et al.*, "Photon-efficient quantum key distribution using time–energy entanglement with high-dimensional encoding," *New Journal of Physics*, vol. 17, no. 2, p. 022002, 2015.

[6] D. Slepian and J. Wolf, "Noiseless coding of correlated information sources," *IEEE Transactions on Information Theory*, vol. 19, no. 4, pp. 471–480, 1973.

[7] H. Zhou, L. Wang, and G. Wornell, "Layered schemes for large-alphabet secret key distribution," in *2013 Information Theory and Applications Workshop (ITA)*, pp. 1–10, 2013.

[8] S. Yang, M. C. Sarihan, K.-C. Chang, C. W. Wong, and L. Dolecek, "Efficient information reconciliation for energy-time entanglement quantum key distribution," in *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pp. 1364–1368, IEEE, 2019.

[9] S. Yang, *Application-Driven Coding Techniques: From Cloud Storage to Quantum Communications*. University of California, Los Angeles, 2021.

[10] O. Ferraz, S. Subramaniyan, R. Chinthala, J. Andrade, J. R. Cavallaro, S. K. Nandy, V. Silva, X. Zhang, M. Purnaprajna, and G. Falcao, "A survey on high-throughput non-binary ldpc decoders: Asic, fpga, and gpu architectures," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 1, pp. 524–556, 2022.

[11] X.-Y. Hu, E. Eleftheriou, and D. Arnold, "Regular and irregular progressive edge-growth tanner graphs," *IEEE Transactions on Information Theory*, vol. 51, no. 1, pp. 386–398, 2005.

[12] F. W. Glover and G. A. Kochenberger, *Handbook of metaheuristics*, vol. 57. Springer Science & Business Media, 2006.