Efficient Quantum Conference Key Agreement over Quantum Networks

Samuel Oslovich*, Md Zakir Hossain†, Trevor Thomas†, Bing Wang†, Walter O. Krawec†, Kenneth Goodenough‡
*QuTech, Delft University of Technology, Delft, The Netherlands
†School of Computing, University of Connecticut, Storrs, CT, USA
‡Manning College of Information & Computer Sciences, University of Massachusetts, Amherst, MA, USA

***Abstract*— Quantum conference key agreement (CKA) is useful for many applications that involve secure communication or collaboration among multiple parties. While CKA over quantum networks can be achieved using pairwise quantum key distribution, a more efficient approach is to establish keys among the parties directly through multipartite entanglement distribution. Existing studies on multipartite entanglement distribution, however, are not designed for CKA, and hence do not aim to optimize key rate. In this paper, we first develop an efficient 3-party CKA strategy based on a closed-form expression that we derive for estimating errors. We then develop a general strategy for N-party CKA that accounts for estimated key rates on individual network paths. For both cases, we use multipath routing to improve key rate. We evaluate our approach in a wide range of settings and demonstrate that it achieves high key rate and degrades gracefully when increasing the number of parties.**

I. Introduction

Quantum conference key agreement (CKA), or multiparty quantum key distribution (QKD), extends pairwise QKD to allow multiple parties to establish a common information-theoretic secret key. It is useful for many applications, including secure multiparty communication, distributed cryptographic applications, and quantum secure multi-party computation [1]. When the multiple parties are at geographically distributed locations, long-distance quantum networks provide a mechanism to connect them using intermediate quantum repeaters. While CKA can be achieved through pairwise QKD, which first establishes pairwise secret keys and then uses secure communication to establish a common key among all the parties, this process is inefficient [2]. A more efficient alternative is distributing multipartite entanglement directly to the multiple parties over a quantum network to establish keys among these parties.

Existing experimental studies [3], [4] have demonstrated the feasibility of CKA through multipartite entanglement distribution over small-scale quantum networks. Efficient CKA over large-scale quantum networks, however, faces many challenges due to limitations of near-term technologies such as lossy and noisy quantum channels, limited quantum memory with short lifetime, and probabilistic entanglement swapping operations. To improve key rate, an important building block is efficient multipartite entanglement distribution over quantum networks. While this problem has been explored in recent studies [5]–[9], their designs do not target CKA, and hence they do not optimize key rate. Specifically, the studies in [5], [6] aim to minimize the number of consumed entangled pairs under idealized conditions, with no consideration of loss and noise. The study in [7] considers a more realistic noisy quantum network setting, and develops a multi-objective routing algorithm considering both entanglement generation rate and fidelity. The study in [8] develops multipath routing techniques to improve entanglement throughput, without taking quantum link fidelity into account, however fidelity is important for CKA since key rate is affected heavily by fidelity. The study in [9] maximizes the expected multipartite entanglement generation rate under given fidelity constraints.

In this paper, we develop efficient CKA strategies for general network topologies, while incorporating the various constraints in near-term quantum networks. We start with 3-party CKA. In this setting, we first derive a closed-form expression for estimating errors, and then use it to estimate key rate and develop an efficient CKA strategy. For general N-party CKA, we develop an approach that directly incorporates the estimated key rates of individual network paths into the design. For both cases, we use multipath routing, specifically multiple trees for connecting the N parties, to improve key rate. While multipath routing is also used in [8], our design differs significantly from [8] in that we explicitly consider fidelity and aim to maximize key rate for CKA.

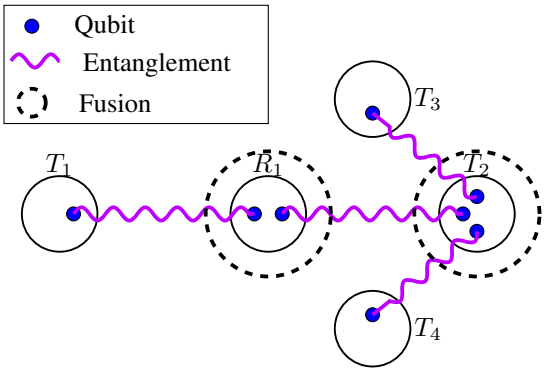
We evaluate our proposed approach in a wide range of settings. The evaluation results show that our approach achieves high key rate and degrades gracefully when increasing the number of parties. Furthermore, it significantly outperforms several baselines: it achieves up to 223% higher key rate than a fixed multi-tree algorithm, and achieves up to 85% higher key rate than a hop-count strategy in heterogeneous settings.

The rest of the paper is organized as follows. In Section II, we present background and network model. In Sections III and IV, we develop 3-party and N-party CKA strategies, respectively. In Section V, we present our evaluation results. In Section VI, we briefly review related work. Last, Section VII concludes the paper.

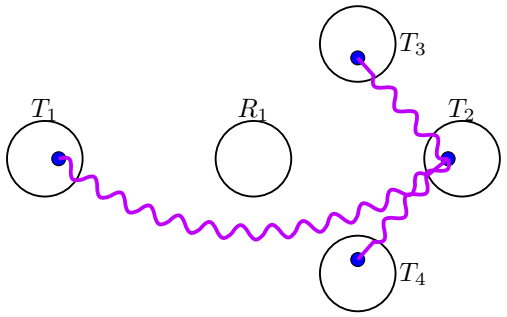
II. Background and Network Model

A. N-party CKA Protocol

We use the N-BB84 protocol in [10] for CKA among N parties. The N parties are denoted as Alice, A, and a set of Bobs, $B_1, \ldots, B_{N-1}$. The goal of the protocol is to establish a common secret key among all N parties. In the rest of the



(a) Fusion-and-discard at R_1; fusion-and-retain at T_2.



(b) Resulting 4-GHZ state shared by terminal nodes.

Fig. 1: Illustration of entanglement fusion.

paper, we refer to A as the *leader*, since it leads the error correction among the N parties (see below).

This protocol contains two stages: quantum and classical information processing stages. In the quantum information processing stage, the protocol starts with distributing multi-partite entangled N-GHZ states, $\frac{1}{\sqrt{2}}\left(|0\rangle^{\otimes N}+|1\rangle^{\otimes N}\right)$, over the quantum channel to the N parties. All parties perform local measurements, either in Z or X basis, based on a preshared key, on their respective quantum systems.

In the classical information processing stage, the parties reveal a random sample of the collected data over the authenticated classical channel for parameter estimation. Specifically, this process estimates Q_{A,B_i}, i.e., the Z basis quantum bit error rate (QBER) between A and B_i, $i=1,\ldots,N-1$, as well as Q_X, i.e., the X basis QBER across all N parties. At this point, the raw keys held by the N parties are partially correlated and partially secret. In order to correct the errors in the raw keys, A performs pairwise error correction with each B_i to create shared raw keys among the N parties. At last, through privacy amplification, the shared raw key is turned into shared secret key for the N parties.

Let r denote the *key rate*, i.e., the length of the shared secret key divided by the number of N-GHZ states distributed among the N parties. The study in [10] presents a finite-key expression for r, which converges to

$$r = 1 - H(Q_X) - \max_i H(Q_{A,B_i}) \ , \tag{1}$$

where $H(x) = -x\log x - (1-x)\log(1-x)$ is the binary entropy function. In this paper, we focus on asymptotic results, and use the above asymptotic key rate.

B. Network Model

The above CKA protocol requires the distribution of N-GHZ states to the N parties prior to initiating the protocol. We assume that the N-GHZ states are distributed using a quantum network that connects the N parties.

Consider a quantum network that contains a set of nodes and edges. Among them, the N nodes that will perform N-party CKA are referred to as *terminal nodes* or *terminals*, denoted as $T_1,\ldots,T_N$, and the rest of the nodes are quantum repeaters. As in [11]–[13], we assume that the nodes have synchronized clocks and the network operates in *rounds*. An edge that connects two nodes represents a fiber link, which can be noisy and lossy. We assume that each node has a quantum memory capable of storing one qubit for each of its connected edges. In addition, all qubits can only be stored for a single network round due to short quantum memory coherence time [14], after which it is discarded. The memory may also be noisy, causing the state to decohere with a certain probability.

Each network round is divided into three *phases* (see below). After a sufficient number of rounds, classical post-processing is conducted among all the terminal nodes to obtain secret key.

Phase 1. This phase distributes link-level bipartite entanglement over each link in the network. Specifically, each pair of connected nodes, u and v, attempts to share half of an entangled pair with each other over the fiber link. This process succeeds with probability $p_{u,v}$, which decays exponentially with distance [15]–[17]. In addition, due to noise in the link, memory system, or both, even when entanglement is established successfully between u and v, we assume a depolarizing channel with parameter $\gamma_{u,v}$. That is, the entanglement may depolarize and become a completely mixed state. Thus, at the end of this phase, with probability $p_{u,v}$, each pair of neighbors u,v share the state

$$\rho_{u,v} = \gamma_{u,v}\left|\Phi^+\right\rangle\left\langle\Phi^+\right| + (1-\gamma_{u,v})\frac{I}{4}\,, \tag{2}$$

where $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle+|11\rangle)$ and I is the identity matrix. Otherwise, with probability $1-p_{u,v}$, the neighbors share the vacuum state (i.e., they share no state). We assume nodes are able to determine whether they have a vacuum or not, while they cannot determine if the state is the desired Bell state $|\Phi^+\rangle$ or a completely mixed state.

Phase 2. In this phase, a routing algorithm is executed (see §III and §IV) to create N-GHZ state among the terminals. We focus on routing algorithms that use global information, assuming that link-level entanglement status is known through classical communication. Specifically, the routing algorithm will determine a routing tree, where the leaves of the tree are terminal nodes, and internal nodes can be quantum repeaters or terminal nodes.

Each internal node of the tree performs *entanglement fusion* to create an entanglement state among its neighbors [18]. When entanglement fusion at each internal node succeeds, an

N-GHZ state is created successfully among the N terminal nodes. As in [9], we use two types of fusions, *fusion-and-retain* where the node performing the fusion retains a single qubit that is part of the resulting N-GHZ state, and *fusion-and-discard* where the node is not part of the resulting state. In our context, if an internal node is a terminal node, it performs fusion-and-retain; otherwise, it performs fusion-and-discard. One example is in Fig. 1, where terminal T_2 shares entanglement with three neighbors, and performs fusion-and-retain, while quantum repeater R_1 shares entanglement with two neighbors and performs fusion-and-discard.

Henceforth, we refer to the above two types of fusion operations simply as fusion; the meaning is clear depending on whether a node performing fusion is a terminal or not. Fusion is a probabilistic operation. Let q_u denote the probability that the fusion operation at node u succeeds.

Phase 3. With N-GHZ state shared among the terminals, each terminal conducts the quantum portion of CKA protocol (see §II-A). At the end of this phase, all remaining qubits in the system are discarded, as we assume quantum memories can only store qubits for a single network round. The network then repeats the above three phases for the next round.

Post-processing. When the network has been running for a sufficient number of rounds, post-processing is executed. Here, error correction and privacy amplification are run among all terminal nodes, generating secret key material. As we are interested only in the asymptotic performance in this paper, we assume perfect error correction and use asymptotic key rates for the N-party CKA protocol. Our main performance metric is the final key rate, namely Eq. (1).

III. 3-Party CKA

In this section, we consider CKA with $N = 3$; CKA for general N is deferred to §IV. We focus on Phase 2 of each network round, i.e., designing routing algorithms to maximize the resultant key rate. These algorithms take a snapshot of the network as input, where each link in the snapshot represents successful link-level entanglement after Phase 1.

A. Estimating Error Rates and Expected Key Rate

In general, a routing tree for three terminal nodes can be represented as a star, with a center node, u, connecting the three terminal nodes, as shown in Fig. 2a. A special case of the above is that the center node u coincides with a terminal node, e.g., as in Fig. 2b.

Let $P_{T,u} = (T, v_1, \ldots, v_k, u)$ denote the path from terminal node T to center node u. Let $q_{T,u}$ denote the probability that all the fusion operations along path $P_{T,u}$ succeed. Then $q_{T,u} = \Pi_{i=1}^{k} q_{v_i}$, where q_{v_i} is the probability that the fusion at node v_i succeeds. Then with probability $q_{T,u}$, nodes T and u share the following state

$$\rho_{T,u} = \gamma_P \left|\Phi^+\right\rangle \left\langle\Phi^+\right| + (1 - \gamma_P) \frac{I}{4}, \tag{3}$$

where $\gamma_P = \gamma_{T,v_1}\gamma_{v_k,u}\Pi_{i=1}^{k-1}\gamma_{v_i,v_{i+1}}$, and $\gamma_{v,v'}$ is the depolarizing parameter for link (v, v').

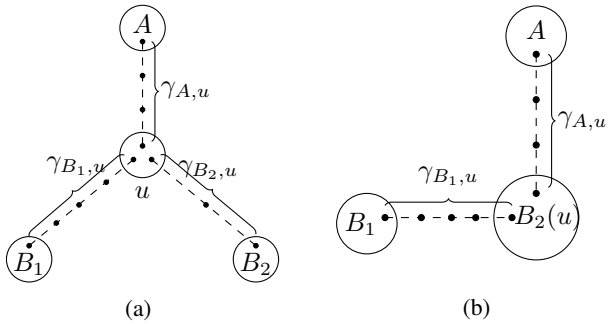


Fig. 2: 3-GHZ state distribution for 3-party CKA: (a) shows the general case where a center node (repeater) connects three terminal nodes, forming a star/tree; (b) shows a special case where the center node is a terminal, and hence the topology simplifies into a line.

After determining the leader in a network round (see below), the leader is denoted as A, and the other two terminals are denoted as B_1 and B_2. Let $\gamma_{A,u}$ denote the depolarizing parameter for the path between leader A and u; similarly, define $\gamma_{B_i,u}$, $i = 1, 2$. Both $\gamma_{A,u}$ and $\gamma_{B_i,u}$ can be obtained based on per-link depolarizing parameter as described earlier. In the special case in Fig. 2b where $u = B_2$, we have $\gamma_{B_2,u} = 1$.

To estimate the key rate in Eq. (1), we need to estimate the Z and X basis error rates. For a given center u, let $Q_{A,B_i}^{(u)}$ denote the Z basis error rate, and $Q_X^{(u)}$ denote the X basis error rate. We derive them for the 3-party case as follows:

$$Q_{A,B_i}^{(u)} = \frac{1 - \gamma_{A,u}\gamma_{B_i,u}}{2}, \tag{4}$$

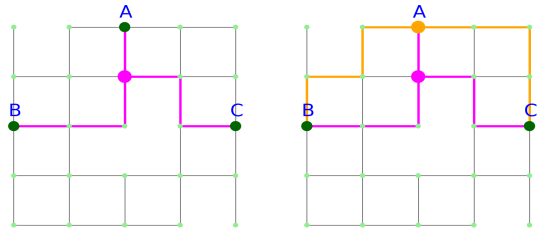
$$Q_X^{(u)} = \frac{1 - \gamma_{A,u}\prod_i \gamma_{B_i,u}}{2}. \tag{5}$$

The derivation is based on the stabilizer formalism [19] and is found in a longer version [20]. Note from Eq. (4) that $Q_{A,B_i}^{(u)}$ depends on the choice of the leader A, while from Eq. (5), $Q_X^{(u)}$ does not depend on leader selection (it is simply related to the depolarizing parameters on the paths from center node u to each of the terminals).

Given the above estimated error rates, the expected key rate when using u as the center node is

$$r^{(u)} = q_u \left[\prod_T q_{T,u}\right] r_{\max}^{(u)}, \tag{6}$$

where q_u is the success probability of the entanglement fusion at center node u, and the product of the probabilities in the bracket represents the success probability of fusion on each path between a terminal T and u, and $r_{\max}^{(u)}$ is the maximum key rate that can be obtained for the given 3-GHZ state with u as the center node, which depends on the leader selection in the CKA protocol. Specifically, to determine the leader, we consider each terminal as the leader A, and compute $H(Q_{A,B_i}^{(u)})$ accordingly. To find the leader that yields the



(a) (b)

Fig. 3: 3-party CKA: (a) Finding one star using the proposed algorithm. (b) Repeatedly running the algorithm until no more star is found yields two stars.

highest key rate, we choose A such that $\max_i H(Q^{(u)}_{A,B_i})$ is minimized, resulting in the maximum key rate for center u as

$$r^{(u)}_{\max} = 1 - H(Q^{(u)}_X) - \min_A \max_i H(Q^{(u)}_{A,B_i}) \,. \tag{7}$$

B. Center Selection and Multipath Routing

In each network round, we select the center node using a method similar to the shortest-star algorithm in [7]. Intuitively the paths from the terminals to the best center node are the paths that yield the highest key rate. We refer to such paths as the *shortest paths*. Additionally, the paths to the center node must be disjoint due to entanglement being a resource that can only be used once. This means that our paths to the best center will be the shortest possible disjoint paths. We consider each node as a center, and check if shortest disjoint paths exist. This results in the following algorithm.

1. Each terminal finds all shortest paths to each other node in the network.
2. Each node u in the network is considered as the center node. If disjoint paths from the terminals to that center node do not exist, the node is ignored. Otherwise, the expected key rate, $r^{(u)}$, of that center node is computed (see Eq. (6)).
3. The node that yields the highest key rate is chosen as the center node.

Running the above algorithm results in a star with the three terminals at the tip of the star. In the special case where a terminal is selected as the center node, our graph is a line with the remaining two terminals located at each end. One example is shown in Fig. 3a for a grid topology, where the edges represent successful link-level entanglements after Phase 1. The resulting star from the routing algorithm is marked in purple in Fig. 3a. If we remove all edges used in the star in Fig. 3a, and re-run the above algorithm, we are able to find another star/line (marked in orange) as seen in Fig. 3b. We repeat this process of greedily finding stars until no more star can be found. Henceforth, we refer to the above algorithm as *multi-star* algorithm. We leave the complexity analysis of this algorithm to future research.

IV. N-Party CKA

For N-party CKA, unlike 3-party CKA, multi-star routing is no longer the best strategy. For example, when there are 5 terminals in a grid topology, it is not even possible to connect the nodes via a central node since each node has at maximum a degree of 4. Instead, for N-party CKA, we design a multi-tree routing algorithm to maximize its key rate. In the special case that a routing tree is a star, we can estimate the Z and X basis error rates as in Eq. (4) and Eq. (5), respectively. For general tree topology, we were not able to derive closed-form error rate expressions, and estimate the error rates numerically using density matrices for a given network setting [19]. Once a routing tree is determined and the N-GHZ state is distributed to the terminals, we use a similar method to determine the leader for error correction as in the 3-party case.

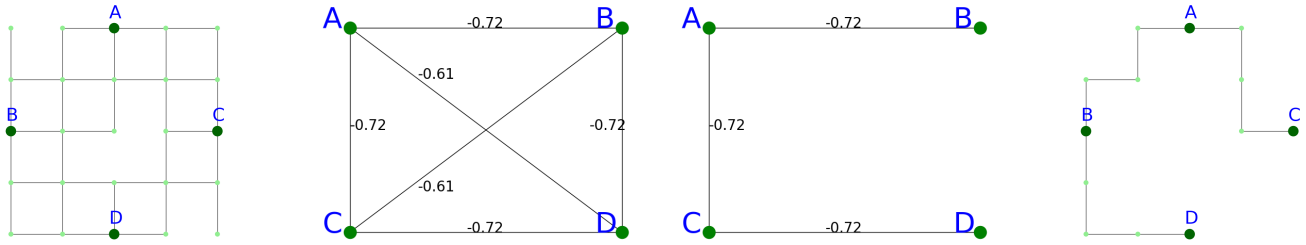
A. Multi-tree Routing Algorithm

Classically finding the minimum cost tree that connects N terminals is an NP-hard problem known as the Steiner tree problem [21]. In the classical setting, the cost of a tree is determined by the sum of the weights of the paths that make up the tree. The study in [8] presents multipath routing algorithms based on Steiner trees for distributing N-GHZ states. Their algorithm, however, assumes the fidelity of all the links to be perfect, and focuses on routing for optimizing entanglement generation rate. As such, their approximation algorithm weights edges equally, and weights paths only by the number of links that they contain. Our design below weights edges and paths proportional to their key rate, allowing us to find routing trees that are better suited for CKA. As we shall see (§V), our algorithm significantly outperforms the algorithm in [8] for key rate in CKA.

Our design is based on the Steiner tree approximation algorithm in [21] and retains the efficient runtime of $\mathcal{O}(|V||E|^2)$, with V, E denoting the vertices and edges respectively. It consists of the following five main steps (see illustration in Fig. 4):

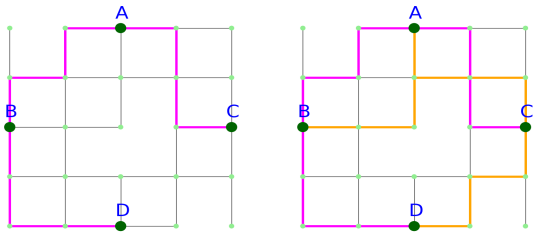
1. For a given snapshot of the network (i.e., after Phase 1 in a network round), find the shortest path (shortest in terms of negative key rate) between each pair of terminals. Construct graph G_1. Each vertex of G_1 is a terminal, and each edge connects a pair of terminals, where the weight of the edge is the negative key rate of the associated shortest path.
2. Find the minimum spanning tree of G_1. Since the weight of each edge represents the negative key rate, the minimum spanning tree will connect terminals by paths with maximal key rates.
3. Construct subgraph G_S where each edge of the minimum spanning tree is replaced by the shortest path (select one arbitrarily if multiple paths exist). If G_S is a Steiner tree, then we are done (i.e., we can skip steps 4 and 5). This is the case for the example in Fig. 4.
4. Find the minimum spanning tree of G_S.
5. Delete edges in the minimum spanning tree until the only leaf nodes are terminal nodes.

After using our algorithm to find a tree to distribute an N-GHZ state, there may still be residual entanglement. One example is shown in Fig. 5a. As in the 3-party case, we



(a) Initial snapshot of G. (b) Subgraph G_1. (c) Min. spanning tree of G_1. (d) Outputted Steiner Tree

Fig. 4: Illustration of the proposed routing algorithm for N-party CKA. (a) Given a snapshot of our graph G, first we find the shortest path between each pair of terminals. (b) Next construct graph G_1, where each edge is the shortest path between terminals. (c) Find the minimum spanning tree of G_1. (d) Replace each edge in G_1 by its corresponding shortest path.



(a) (b)

Fig. 5: Running the proposed routing algorithm results in finding one tree in (a). Repeatedly running the algorithm until no tree is found yields two trees in (b).

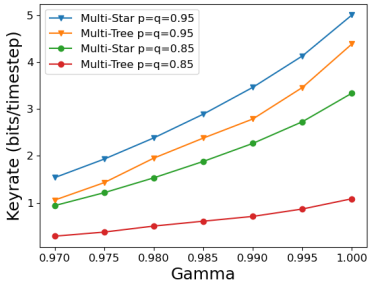


Fig. 6: Results for 3-party CKA for random graphs: multi-star vs. multi-tree algorithm.

remove all the edges that have been used in the tree from the graph, and repeatedly run our routing algorithm until no more tree is found. This process leads to another tree as shown in Fig. 5b. Henceforth, we refer to the above algorithm as *multi-tree* algorithm, which has runtime of $\mathcal{O}(|V||E|^3)$.

B. Multi-tree vs. Multi-star Algorithm

The multi-tree algorithm for the general N-party setting can be used for 3-party case. However, for the 3-party case, the multi-star algorithm can significantly outperform the multi-tree algorithm. One example is in Fig. 6, which shows the key rates obtained by the two algorithms for random graphs for 3-party CKA (see more details on evaluation settings in §V-A). It shows the results for two settings, both homogeneous (and hence we omit subscript for each parameter): $p = q = 0.85$ and $p = q = 0.95$, while varying γ parameter for a link from 0.97 to 1. We see a large gap between the multi-star algorithm and the multi-tree algorithm. This is because the multi-star algorithm finds more stars and stars with higher key rates. For example, when $\gamma = 1$, the multi-star algorithm finds 0.3 more stars/trees per round on average than the multi-tree algorithm.

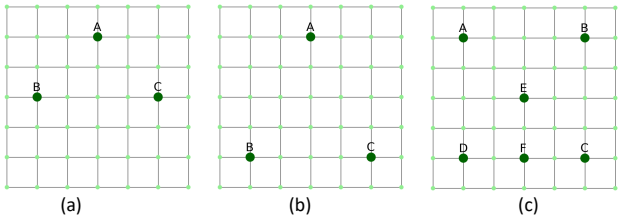


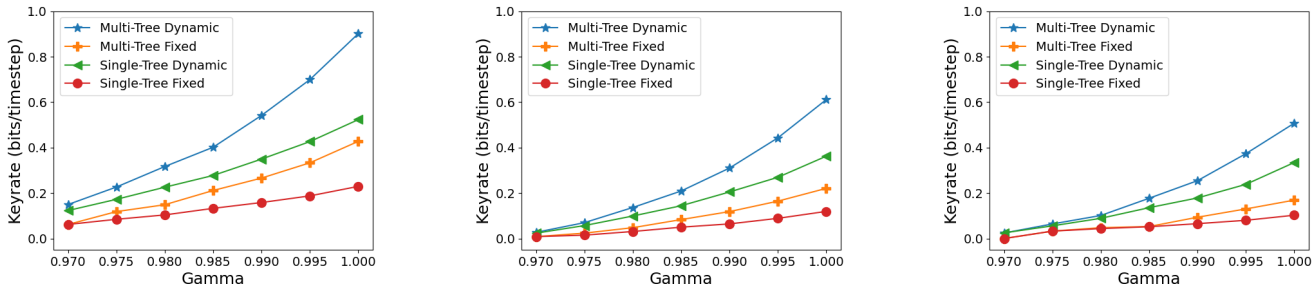
Fig. 7: Placement of terminals in a 7×7 grid. (a) ᗺ (Bet) layout. (b) Δ (Dalet) layout. (c) ℸ (Giml) layout with A, B and C; parties D, E, and F added for $N = 4, 5, 6$.

V. Performance Evaluation

In this section, we evaluate our proposed approaches for CKA using extensive simulations. We first present the evaluation setup, and then the results.

A. Evaluation Setup

We consider two different choices of routing strategies. The first choice is between *fixed routing* and *dynamic routing*. In fixed routing, the routing algorithm is run once prior to link-level entanglement establishment (Phase 1). This results in a fixed star/tree being chosen; and if a single link in the pre-chosen path fails to generate entanglement, then entanglement distribution will fail in that round. In dynamic routing, in each round, the routing algorithm is run on a snapshot of the graph after link-level entanglement establishment (Phase 1), where link-level entanglement success is assumed to be global knowledge (obtained via classical communication). The second choice is between *single-tree* vs. *multi-tree* routing. In single-tree routing, the routing algorithm finds a single tree for distributing entanglement, while in multi-tree routing, multiple trees/stars are found as described in §III and §IV. The fixed strategies are presented as a baseline to show how dynamic strategies that leverage global information of link-state success



(a) ᚹ (Bet) layout. (b) Δ (Dalet) layout. (c) ᛏ (Giml) layout.

Fig. 8: Results for various 3-party routing schemes for 3-party CKA in three layouts, 7×7 grid, $p = q = 0.85$.

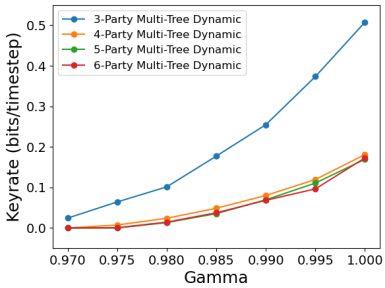


Fig. 9: Key rate when varying N from 3 to 6 for dynamic multi-tree scheme, $p = q = 0.85$, incremental ᛏ (Giml) layout in 7×7 grid.

(after Phase 1) can greatly improve the key rate. Additionally, single-tree vs. multi-tree routing is intended to highlight how utilizing as many resources as possible can improve the key rate. In the rest of this section, for ease of exposition, we refer to multi-star and multi-tree algorithms both as multi-tree; if it is for 3-party CKA, then the results are obtained using the multi-star algorithm due to its better performance.

We consider both grid and random graph topologies. The grid topologies are used to gain insights into how varying different parameters impacts the key rate. Random graph topologies are used to evaluate the routing algorithms under more realistic settings.

We choose to vary several parameters and examine how they impact the key rate. Specifically, link-level entanglement success probability p is set to 0.85 or 0.95, entanglement swapping success probability q is also set to 0.85 or 0.95, and channel noise parameter γ is varied from 0.97 to 1.0, with step size of 0.005. The number of parties, N, is varied from 3 to 6. We consider both homogeneous settings where each link/node in the network has the same parameter (§V-B and §V-C), and heterogeneous settings (§V-D).

B. Results for Grid Topologies (Homogeneous Setting)

We explore two grid topologies: 7×7 and 11×11. For both of them, we examine three 3-party layouts. Based on the locations of the terminals, we refer to them as ᚹ (Bet), Δ (Dalet), and ᛏ (Giml) layouts due to their similarities to letters in the Phoenician alphabet. Fig. 7 shows these three layouts in the 7×7 grid; the layouts in the 11×11 grid are in a similar manner. For both In 7×7 and 11×11 grids, we further incrementally add terminals to the ᛏ (Giml) layout to form the setup for $N = 4$, 5, and 6.

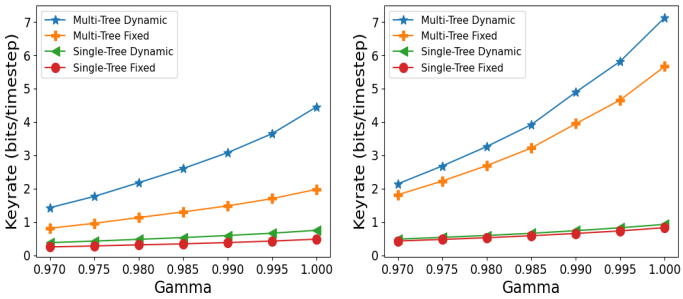
We next present the results for the 7×7 grid. In Fig. 8, we compare the performance of four routing algorithms when $p = q = 0.85$; the results for other settings show similar trend. As expected, single-tree and multi-tree dynamic algorithms outperform single-tree and multi-tree fixed algorithms, respectively. We also see that the ᚹ (Bet) layout yields higher key rates than the Δ (Dalet) and ᛏ (Giml) layouts due to the reduced distance between terminals in the ᚹ (Bet) layout. We see that due to limited node degree (no more than 4) of the grid network, the single-tree dynamic algorithm outperforms the multi-tree fixed algorithm, which is not true in networks with richer connectivity (see Fig. 10).

As the number of parties increases, naturally we expect the key rate to decrease. Interestingly, for the incremental ᛏ (Giml) layout, we see in Fig. 9 that 4, 5, and 6 party CKA have very similar key rates, with a large gap between the 3-party and 4-party case. The closeness of the 4, 5, and 6 party key rates is due to the similarity of the trees found. The large gap between the 3 and 4 party cases can be partially explained by the non-optimality of our Steiner tree approximation algorithm that is used for the 4-party case, while the multi-star algorithm is used for the 3-party case. Additionally, this gap is exacerbated in the grid setting due to the limited connectivity between nodes. We shall later see that with richer connectivity, as is the case for random graphs, the degradation for increasing N is much less significant.

The results for 11×11 grid show similar trends albeit with significantly lower key rates due to the increased distance between nodes. In fact, for values of γ below 0.98, all 3-party schemes in the 11×11 grid result in a key rate of 0. This performance only worsens as N increases.

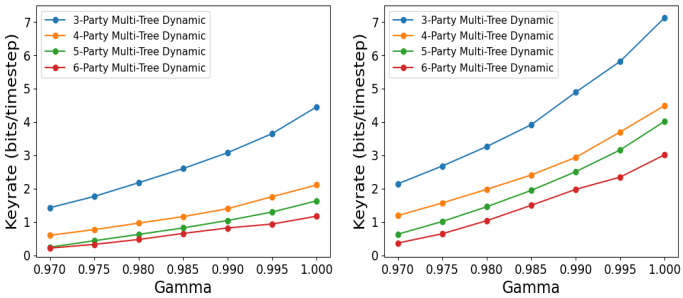
C. Results for Random Graphs (Homogeneous Setting)

We examine five random graphs and report the average results. Each graph is generated by placing 50 nodes randomly in a unit square. Two nodes are connected by a link if their distance is less than 0.3. These random graphs have a significantly higher average node degree than our grid graphs



(a) $p = q = 0.85$. (b) $p = q = 0.95$.

Fig. 10: Results for 3-party CKA (averaged over five random graphs). The results for the other two settings ($p = 0.85$, $q = 0.95$; $p = 0.95$, $q = 0.85$) show similar trend and are omitted.



(a) $p = q = 0.85$. (b) $p = q = 0.95$.

Fig. 11: Results for N-party CKA under dynamic multi-tree scheme (averaged over 5 random graphs) when varying γ, p and q, for $N = 3$, 4, 5, and 6. The results for the other two settings ($p = 0.85$, $q = 0.95$; $p = 0.95$, $q = 0.85$) show similar trend and are omitted.

(9.75 vs. roughly 4). We expect that this will improve the performance of the multi-tree routing algorithm, as higher node degrees lead to more routing trees.

Fig. 10 plots the average key rate for 3-party CKA in the random graphs for various values of p and q. Compared to 3-party CKA for the three grid topologies in Fig. 8, we indeed see much higher key rate for the random graphs. For instance, in Fig. 10a, the multi-tree dynamic algorithm achieves a maximum key rate of over 4, compared to the maximum in the grid setting of just over 0.9 in Fig. 8a.

In Fig. 10, increasing p has the most impact on fixed routing algorithms due to their reliance on specific links being established to distribute entanglement. Similarly, we see that increasing q has the largest impact on the dynamic routing algorithms. Once again this makes sense due to q and γ being the only parameters in our key rate equation for dynamic routing. Increasing p and q leads to much higher impact on the multi-tree routing algorithms than the single-tree algorithm, since each found tree is independently impacted by the increase of p or q. As expected, when increasing both p and q we see the largest increase in the key rate for both the fixed and dynamic multi-tree routing algorithms.

We next compare the results of the different algorithms.

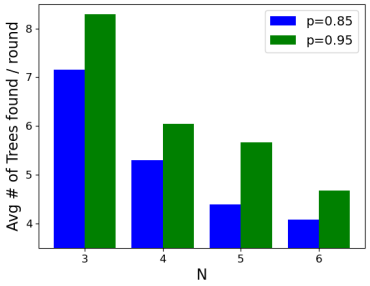


Fig. 12: Histogram showing the average number of trees found in each round versus N when $p = 0.85$ or 0.95, and $q = 0.85$.

In Fig. 10 for 3-party CKA, compared to the baseline fixed single-tree algorithm, the dynamic single-tree algorithm results in a key rate increase of 11% to 54% for the various values of p and q. This increase is also seen in the dynamic multi-tree compared to the fixed multi-tree algorithm, resulting in a key rate increase of 25% to 148%. As we increase N, we see even larger percentage increase. For instance, for $N = 4$, 5, and 6 parties, dynamic multi-tree algorithm leads to 137%, 191%, 223% higher key rate than fixed multi-tree algorithm when $p = q = 0.85$ (figure omitted). This is because as the number of parties increases, the trees that are found consist of more links, which degrade the performance of fixed routing schemes exponentially in both p and q, whereas our dynamic schemes degrade only in q.

Fig. 11 plots the key rate for dynamic multi-tree algorithm when varying N from 3 to 6 in the random graphs. We see a clear gap between each plotted key rate in Fig. 11, in contrast to the grid topology where 4, 5, and 6 parties achieved similar key rates. Adding terminals to random graphs has significantly less impact than that in the grid topology due to richer connectivity in random graphs. Again we see a large gap between the key rates as N increases from 3 to 4. When varying p and q for increasing N, we see similar trends exhibited in the 3-party case. We see that increasing p has little effect on the dynamic routing schemes. As seen in the 3-party case, when increasing q, the key rate increases significantly. In addition, increasing q reduces the gap between the 3 and 4 party key rates.

Fig. 12 plots the histogram of the average number of trees found for $N = 3$ to 6, where $q = 0.85$ and p is 0.85 or 0.95. We see that the average number of trees found tends to decrease when increasing N. As expected, when increasing p we see that more trees are found since higher p leads to more successful link-level entanglements, and hence more trees can be found. Finding trees is independent of q, which is only taken into account when attempting to distribute entanglement across nodes in the routing trees.

D. Results for Random Graphs (Heterogeneous Settings)

We now present the results in heterogeneous settings for random graphs. For each of the 5 random graphs, we created 10 settings, where the parameters for each node and link

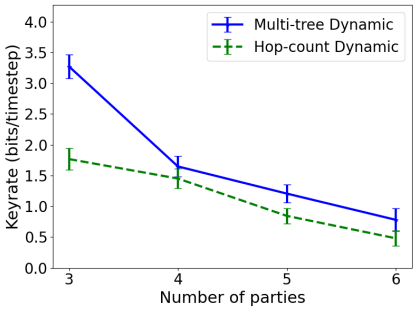


Fig. 13: Results for random graphs in heterogeneous settings (averaged over 50 settings): comparison of our scheme and a hop-count based scheme.

is uniformly randomly chosen in their ranges (see §V-A). We compare our dynamic multi-tree algorithm with a simple strategy that is based on hop count, i.e., similar to that in [8]. For each algorithm, we obtain the results from 50 settings and plot the average value with 95% confidence intervals as shown in Fig. 13. When $N = 3$, 4, 5, and 6, on average, our scheme leads to 85%, 14%, 43%, and 62% higher key rate than the hop-count based scheme.

VI. Related Work

The works that are closest to ours are [5]–[9]. As described in §I, they focus on designing efficient multipartite entanglement distribution strategies over quantum networks, not for CKA. The design of our multi-tree algorithms for CKA are inspired by them. On the other hand, we explicitly consider key rate in our design to optimize key rate. Our design is unique in that it focuses on dynamic entanglement distribution and accounts for channel noise while maximizing key rate.

The multipath routing strategies that we develop for CKA are broadly related to the extensive literature on routing in quantum networks (e.g., [11], [12], [22]–[25]). These studies however focus on pairwise Bell pair distribution, instead of multipartite entanglement distribution to multiple parties. Routing multipartite entanglement to multiple parties (as done in this work) is significantly more challenging than routing bipartite entanglement to two parties.

The design and security analysis of CKA have been studied extensively (see [1] and the references within). They assume entangled GHZ state of particular fidelity, but do not consider CKA in quantum networks, which is the focus of this study.

VII. Conclusion

In this paper, we developed efficient CKA strategies over quantum networks. We derived an analytical expression for determining 3-party CKA key rates, and designed strategies for 3-party and general N-party CKA by incorporating various constraints in near-term quantum network technologies into the design to optimize secret key generation rate. Using extensive evaluation in both grid and random graphs under a wide range of settings, we demonstrated that our schemes achieve high key rates and degrade gracefully when increasing the number of parties.

References

[1] G. Murta, F. Grasselli, H. Kampermann, and D. Bruß, "Quantum conference key agreement: A review," *Adv. Quantum Technol.*, vol. 3, 2020.

[2] M. Epping, H. Kampermann, C. Macchiavello, and D. Bruß, "Multi-partite entanglement can speed up quantum key distribution in networks," *New Journal of Physics*, vol. 19, p. 093012, sep 2017.

[3] M. Proietti, J. Ho, F. Grasselli, P. Barrow, M. Malik, and A. Fedrizzi, "Experimental quantum conference key agreement," *Science Advances*, vol. 7, no. 23, 2021.

[4] A. Pickston, J. Ho, A. Ulibarrena, F. Grasselli, M. Proietti, C. L. Morrison, P. Barrow, F. Graffitti, and A. Fedrizzi, "Conference key agreement in a quantum network," *npj Quantum Information*, vol. 9, Aug. 2023.

[5] C. Meignant, D. Markham, and F. Grosshans, "Distributing graph states over arbitrary quantum networks," *Physical Review A*, vol. 100, November 2019.

[6] A. Fischer and D. Towsley, "Distributing graph states across quantum networks," in *Proc. of QCE*, IEEE, 2021.

[7] L. Bugalho, B. C. Coutinho, F. A. Monteiro, and Y. Omar, "Distributing multipartite entanglement over noisy quantum networks," *Quantum*, vol. 7, p. 920, February 2023.

[8] E. Sutcliffe and A. Beghelli, "Multiuser entanglement distribution in quantum networks using multipath routing," *IEEE Transactions on Quantum Engineering*, vol. 4, p. 1–15, 2023.

[9] M. Ghaderibaneh, H. Gupta, and C. Ramakrishnan, "Generation and Distribution of GHZ States in Quantum Networks," in *IEEE QCE*, 2023.

[10] F. Grasselli, H. Kampermann, and D. Bruß, "Finite-key effects in multipartite quantum key distribution protocols," *New Journal of Physics*, vol. 20, p. 113014, nov 2018.

[11] M. Pant, H. Krovi, D. Towsley, L. Tassiulas, L. Jiang, P. Basu, D. Englund, and S. Guha, "Routing entanglement in the quantum internet," *npj Quantum Information*, vol. 5, no. 1, pp. 1–9, 2019.

[12] S. Shi and C. Qian, "Concurrent entanglement routing for quantum networks: Model and designs," in *ACM Sigcomm*, August 2020.

[13] O. Amer, W. O. Krawec, and B. Wang, "Efficient routing for quantum key distribution networks," in *Proc. of QCE*, 2020.

[14] A. Dahlberg, M. Skrzypczyk, T. Coopmans, L. Wubben, F. Rozpędek, M. Pompili, A. Stolk, P. Pawełczak, R. Knegjens, J. de Oliveira Filho, R. Hanson, and S. Wehner, "A link layer protocol for quantum networks," in *Proc. of ACM SIGCOMM*, 2019.

[15] O. Svelto, *Principles of Lasers*. Springer US, 2010.

[16] H. Kaushal, V. K. Jain, and S. Kar, *Free Space Optical Communication*. Springer, 2017.

[17] S. Pirandola, R. Laurenza, C. Ottaviani, and L. Banchi, "Fundamental limits of repeaterless quantum communications," *Nature communications*, vol. 8, no. 1, pp. 1–15, 2017.

[18] S. de Bone, R. Ouyang, K. Goodenough, and D. Elkouss, "Protocols for creating and distilling multipartite GHZ states with Bell pairs," *IEEE Transactions on Quantum Engineering*, vol. 1, p. 1–10, 2020.

[19] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[20] S. Oslovich, M. Z. Hossain, T. Thomas, B. Wang, W. Krawec, and K. Goodenough, "Efficient multiparty quantum key distribution over quantum networks," 2024. https://arxiv.org/abs/2404.19720.

[21] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for steiner trees," *Acta Informatica*, vol. 15, pp. 141–145, 06 1981.

[22] M. Caleffi, "Optimal routing for quantum networks," *IEEE Access*, vol. 5, pp. 22299–22312, 2017.

[23] K. Chakraborty, F. Rozpedek, A. Dahlberg, and S. Wehner, "Distributed routing in a quantum internet," *arXiv preprint arXiv:1907.11630*, 2019.

[24] Y. Zhao and C. Qiao, "Redundant entanglement provisioning and selection for throughput maximization in quantum networks," in *IEEE INFOCOM*, 2021.

[25] E. Kaur and S. Guha, "Entanglement distribution in two-dimensional square grid network," *arXiv preprint arXiv:2306.03319*, 2023.