Key Transport over Untrusted QKD Relay Networks

Sebastian Clermont[1], Antoine Gansel[2], and Patrick Struck[3]

[1] TU Darmstadt, sebastian.clermont@tu-darmstadt.de
[2] Universität Regensburg, antoine.gansel@ur.de
[3] Universität Konstanz, patrick.struck@uni.kn

Abstract. Quantum key distribution (QKD) enables the exchange of information-theoretically secure symmetric keys, but is fundamentally limited in range. Existing long-distance QKD networks rely on trusted relay nodes, any one of which can compromise the entire key.
We propose a key-transport protocol that removes this trust assumption by combining proactive secret sharing with one-time pad encryption over pairwise QKD links. At each layer of relay nodes, shares are reshared so that corruptions across different layers cannot be combined; security depends only on the maximum number of corruptions within any single layer, not on the total number of corrupted nodes. We formalize a game-based security model for layered secret transport, identify a cross-layer attack that affects a prior construction, and prove our protocol information-theoretically secure against a semi-honest adversary corrupting up to $t-1$ nodes per layer.

Keywords: Long-Distance QKD, Networks, Secret Sharing

1 Introduction

Key-exchange is a core cryptographic concept that allows two parties to establish a shared key. This key can then be used to securely communicate. The idea of key-exchange dates back to the seminal work by Diffie and Hellman [DH76], which provides a key-exchange protocol based on the hardness of discrete logarithms.

The advent of quantum computing had significant impact on cryptographic research. *Post-quantum cryptography* (PQC) emerged as a field of research to deal with the threat that quantum computers will have on public-key cryptography: by efficiently solving the underlying mathematical problems, quantum computers will render much of today's public-key cryptography insecure—in particular, the aforementioned key-exchange protocol due to Diffie and Hellman.

* This work was funded by the German Research Foundation (DFG) – SFB 1119 – 236615297, the German Federal Ministry of Research, Technology and Space (BMFTR) under the project SEQUIN (16KIS2123), the Hector Foundation II, and the German Federal Ministry of Research, Technology and Space and the Hessian Ministry of Higher Education, Research, Science and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE.

Besides its destructive effect, quantum computing, in the form of *quantum key distribution* (QKD), also brings constructive changes to cryptography. These arise from quantum-mechanical phenomena such as superposition and entanglement. Compared with traditional key-exchange methods, QKD offers several advantages. First, it does not rely on mathematical hardness assumptions. [4] Second, exchanged keys are perfect in the sense that an adversary has not learned anything about the key. Finally, it enables the exchange of truly random keys, whereas traditional methods only provide pseudorandom keys. [5]

The main disadvantage of QKD over traditional key-exchange methods is its limited range, particularly for fiber-based QKD [PLOB17]. There exist multiple different QKD protocols [Eke91,BB14,BBM92], but they all require the physical exchange of either entangled particles or particles prepared by one party and measured by the other. Due to attenuation and the limitations of current detectors, distances of 100km are generally considered feasible for establishing a secure key at a reasonable speed.

Extending the range is highly non-trivial, and none of the approaches for increasing the signal strength of classical communication translate to the QKD setting. A simple repeater is infeasible due to the no-cloning property of quantum states. Special quantum repeaters [AEE+23] do exist in theory, but a successful demonstration has yet to be made. Another approach, already used in practice, is to use satellite-to-ground QKD [LCL+17]. Although it can exchange keys over up to a thousand kilometres (depending on atmospheric conditions), this method often requires specific hardware that is not necessarily miniaturisable, making it unsuitable for most potential use cases. While progress is being made, the general consensus among researchers is that it can easily take a decade or more before any technology to overcome the trusted node problem becomes mature enough for proof-of-concept demonstrations.

To overcome this limitation, long-distance QKD networks have been designed to use so-called trusted nodes [CZC+21,HHB+25]. These trusted nodes have individual QKD connections to their predecessor and their successor node. They receive the key securely from the predecessor and use the QKD link with their successor node to forward it to its destination. The big downside is that end-to-end security is impossible in this setup. The trusted nodes must always be trusted to faithfully handle key forwarding and not leak any information, requiring physical security precautions at their deployment sites.

1.1 Contribution

In this work, we develop a protocol to securely exchange a shared key between two parties that have no QKD connection between one another but instead are

[4] An often neglected aspect is the problem of authentication, though recent works [GKS+26,WZW+21] show positive results regarding this.

[5] One could argue that traditional methods could leverage quantum random number generators to also generate truly random keys, however, the final application of a key-derivation function would still render these keys to be only pseudorandom.

connected via a sequence of nodes such that neighboring nodes do have a QKD connection. Our protocol relies on (proactive) secret sharing to ensure that the intermediate nodes do not learn the exchanged key and one-time pad encryption to secure the communication between neighboring nodes. We formalize a security model for the distributed long-distance exchange of information-theoretic symmetric keys by adapting the usual security models for key exchange and proactive secret sharing. We then provide a communication protocol based on proactive Shamir's secret sharing and prove its information-theoretic security against a semi-honest adversary that may corrupt up to $t-1$ nodes per layer. This is the first construction in the long-range QKD relay setting to come with a formal security model and proof; all prior work operates without any rigorous adversarial guarantees (cf. Section 1.2). Importantly, extending proactive secret sharing to the relay setting is not straightforward: we identify a cross-layer attack (Appendix A.3)that renders prior constructions [BS08] insecure, while our protocol is designed to resist it. The semi-honest model is well-suited to the envisioned deployment, where relay nodes reside in physically secured facilities and the key-transport phase executes in near real-time, leaving little room for active interference(see Appendix A.2 for a detailed discussion). Finally, we discuss practical considerations including QKD key consumption and outline directions for extending the model in the Appendix.

1.2 Related Work

The idea of leveraging secret sharing to avoid the necessity of trusted nodes has already been explored in the literature. However, existing solutions lack formal security models and provide weaker guarantees than our construction.

To the best of our knowledge, the first work on this topic is [BS08]. Although it presents an interesting first step, it presents some imprecision. Most importantly, they do not provide a formal security model. We further discuss their work in Appendix A.3. The concept of secret sharing to increase QKD's distance has then been further studied, but in a much more rudimentary way. [BP11,ZLHM22,WLX+23], for instance, all restrict themselves to simple secret sharing and fail to provide any adversarial model. Aside from secret sharing, other cryptographic primitives have also been studied in pursuit of range-extending results. [GMK22], although lacking a proper security model, proposes a protocol that forces the adversary to corrupt nodes, such as isolating both ends of the network's graph, to find the key. The current topic can also be related to Distributed Key Distribution Centers [DS02]. Despite the similarities, we will see that the key exchange use-case cannot use the same security model. Lastly, while [FKLR25] might seem related to [GMK22], it only considers external attackers and not possible relay corruption.

2 Preliminaries

Throughout this paper, we work in the information-theoretic setting: all adversaries are computationally unbounded, and all security guarantees hold uncon-

ditionally (i.e., without relying on computational hardness assumptions). After fixing notation, we recall the entropic quantities used to formalize secrecy and then introduce threshold secret sharing together with its proactive extension, which form the cryptographic core of our protocol. We conclude the section with the trust model for intermediate network nodes.

2.1 Notation

We define $[n] \coloneqq \{1, 2, \ldots, n\}$, $\forall n \in \mathbb{N}, n > 0$. All sets are written in *italic*.

We use the following fonts to denote **protocols** and variables. In our algorithms, a deterministic assignment to x is written $x \leftarrow \bullet$. Its uniformly random sampling from a group $\mathbb{S}$ is denoted $x \xleftarrow{\$} \mathbb{S}$, and $x \Rightarrow P$ (resp. $x \Leftarrow P$) means sending (resp. receiving) a variable x to (from) a party P.

2.2 Conditional Entropy

Let X, Y be discrete random variables. The conditional entropy of X given Y is defined as

$$H(X \mid Y) \coloneqq \sum_{y} \Pr[Y = y]\, H(X \mid Y = y), \tag{1}$$

where

$$H(X \mid Y = y) \coloneqq -\sum_{x} \Pr[X = x \mid Y = y] \log \Pr[X = x \mid Y = y]. \tag{2}$$

Intuitively, $H(X \mid Y)$ measures the remaining uncertainty about X after observing Y.

2.3 Secret Sharing

With a secret sharing scheme, a party holding some (cryptographic) secret can split this secret into multiple shares, which can individually be distributed. Usually, a thresholdized variant is used, where some data k is split into n distinct shares, out of which at least t are required to reconstruct k. We refer to such a scheme as a (t, n)-threshold secret sharing scheme. Secret Sharing (and its threshold variant) was first formalized by Shamir [Sha79]. Let $\mathbb{F}$ be a finite field with $|\mathbb{F}| > n$ (so that n distinct non-zero evaluation points exist). To share a secret $k \in \mathbb{F}$, the dealer samples a uniformly random polynomial Q of degree at most $t - 1$ subject to $Q(0) = k$ and distributes $[k]_i \coloneqq Q(\alpha_i)$ to party i, where $\alpha_1, \ldots, \alpha_n \in \mathbb{F} \setminus \{0\}$ are distinct, publicly known evaluation points. Any t shares determine Q uniquely via Lagrange interpolation, whereas any $t - 1$ or fewer shares are statistically independent of k.

Such schemes can be extended to support re-randomisation (or resharing) of shares without first reconstructing the secret [HJKY95]; the resulting primitive is called a *threshold proactive secret sharing scheme*. We will first give the definition here and an instantiation in Section 4.

Definition 1 (Threshold Proactive Secret Sharing). *Any (t, n)-threshold proactive secret sharing scheme* **PSS** *($t \leq n$) consists of four algorithms* $\mathbf{PSS} = (\mathbf{Share}, \mathbf{Reshare}, \mathbf{Recombine}, \mathbf{Reconstruct})$*:*

Share $([k]_1, \ldots, [k]_n) \xleftarrow{\$} \mathbf{Share}(k, n, t)$*: Given the (secret) data to be shared k, the total number of shares n, and the minimum number of shares t required for reconstruction, the algorithm* **Share** *outputs n shares $([k]_1, \ldots, [k]_n)$. For $A \subseteq [n]$, we write $[k]_A \coloneqq \{[k]_i\}_{i \in A}$.*

Reconstruct $k \leftarrow \mathbf{Reconstruct}([k]_A)$*: Given a set $A \subseteq [n]$ with $|A| \geq t$, the algorithm* **Reconstruct** *reconstructs the data k.*

Reshare $([[k]_i]_1, \ldots, [[k]_i]_n) \xleftarrow{\$} \mathbf{Reshare}([k]_i)$*: Given a single share $[k]_i$, the algorithm* **Reshare** *generates n subshares. When applied to a share $[k]_i^{(r)}$ from round r, we write the resulting subshares as $[[k]_i^{(r)}]_1, \ldots, [[k]_i^{(r)}]_n$. The shares output by the algorithm* **Share** *are of round $r = 1$. In the layered network setting (Section 3), the round index r coincides with the layer index a, so round superscripts are left implicit in favour of the layer-indexed notation $[k]_{a,i}$.*

Recombine $[k]_j^{(r+1)} \leftarrow \mathbf{Recombine}(\{[[k]_i^{(r)}]_j\}_{i \in A})$ *with $A \subseteq [n]$, $|A| \geq t$: Given at least t subshares, each being the j-th subshare produced by a different party's resharing of its respective share, it produces a fresh share $[k]_j^{(r+1)}$ of the original secret. One may notice that correctness will later require to execute* $\mathbf{Reconstruct}([k]_{A^*}^{(r+1)})$ *with $A^* \subseteq [n]$ and $|A^*| \geq t$ for it to return the original data k.*

A scheme using only $(\mathbf{Share}, \mathbf{Reconstruct})$ *is a standard (t, n)-threshold secret sharing scheme* **SSS**.

A (t, n)-threshold proactive secret sharing scheme satisfies *correctness* if, for every data k:

1. $(\mathbf{Share}, \mathbf{Reconstruct})$ are said to generate a correct (t, n)-threshold secret sharing scheme **SSS**. That is,
$$\Pr[\mathbf{Reconstruct}([k]_A) = k] = 1$$
holds for all n and all $[k]_A$ where $A \subseteq [n]$, $|A| \geq t$ and $[k]_A \subseteq \mathbf{Share}(k, n, t)$.
2. For any $A, B, C \subseteq [n], |A| \geq t, |B| \geq t, |C| \geq t$ then the following implication holds with a probability of 1.
$$\mathbf{Reconstruct}([k]_A^{(r)}) = k \implies \mathbf{Reconstruct}([k]_B^{(r+1)}) = k$$
$$\text{where } [k]_j^{(r+1)} \leftarrow \mathbf{Recombine}((\{[[k]_i^{(r)}]_j\}_{i \in C}))$$
$$\text{and } ([[k]_i^{(r)}]_j) \leftarrow \mathbf{Reshare}([k]_i^{(r)}) \text{ for any } i \in C$$

Definition 2 (Security of Proactive Secret Sharing). *A (t, n)-threshold proactive secret sharing scheme* **PSS** *satisfies* privacy *if, for every secret k, any $R \in \mathbb{N}$, and any corruption sets $A_1, \ldots, A_R \subseteq [n]$ with $|A_r| \leq t - 1$ for all r:*

$$H(k \mid V_1, \ldots, V_R) = H(k)$$

where

$$V_r \coloneqq \left\{[k]_i^{(r)}\right\}_{i\in A_r} \cup \left\{[[k]_i^{(r)}]_j : i \in A_r \vee j \in A_r\right\}_{i,j\in[n]} \qquad \text{for } r < R$$
$$V_R \coloneqq \left\{[k]_i^{(R)}\right\}_{i\in A_R}$$

and shares evolve as in the correctness definition:

$$([[k]_i^{(r)}]_1, \ldots, [[k]_i^{(r)}]_n) \xleftarrow{\$} \mathbf{Reshare}([k]_i^{(r)}) \qquad \text{for all } i \in [n]$$
$$[k]_j^{(r+1)} \leftarrow \mathbf{Recombine}\left(\left\{[[k]_i^{(r)}]_j\right\}_{i\in[n]}\right) \qquad \text{for all } j \in [n]$$

For $R = 1$, the subshare terms vanish and the definition reduces to privacy of a standard (t, n)-threshold secret sharing scheme. The quantification over all $R \in \mathbb{N}$ is motivated by the information-theoretic security argument: since privacy is stated as an entropic equality rather than a negligible bound, it either holds for every number of resharings or it does not. In our layered network setting, it holds that $R = l$ (one resharing per layer transition). The physical limitations of building the required QKD nodes pose a natural limit for the number of resharings a realistic deployment will require.

2.4 Network Components

Depending on the security model, intermediate nodes in a network can be classified by the trust assumptions placed on them. A *trusted node* follows the protocol specification and preserves the confidentiality and integrity of all secret material it handles. An *honest-but-curious* (or *semi-honest*) node also follows the protocol faithfully, but may attempt to infer additional information from its internal state, inputs, outputs, and all messages it legitimately observes; in particular, it does not commit to keeping its internal state secret. In this work, all intermediate parties are assumed to be honest-but-curious; the formal corruption model is captured in the security game of Fig. 5.

3 Key Transport over a QKD Network

A central challenge in deploying QKD at scale is extending the reach of point-to-point links beyond their physical range limit. Any solution necessarily relies on intermediate classical relay nodes, which introduces a trust problem: how can the endpoint secret key remain confidential when it must traverse potentially untrusted relays? In this section, we present three progressively stronger relay architectures, formalize the network model, and define the protocol syntax and security game.

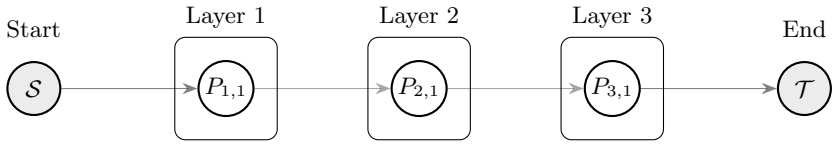

Fig. 1. The topology of a long-range QKD network relying on trusted nodes.

3.1 The Necessity of Relay Nodes

Fiber-based quantum key distribution is fundamentally limited in range: the transmittance of an optical fiber decreases exponentially with distance, imposing an upper bound on the achievable secret-key rate between two parties [PLOB17]. Quantum repeaters [BDCZ98] were the first solution theorised to overcome this limitation. They have the attractive property of preserving information-theoretic security even if individual repeater nodes are adversarially controlled. However, quantum repeaters require long-lived quantum memories, a technology that is not yet available in practice [AEE+23].

Without quantum repeaters, extending QKD beyond a single link requires forwarding *classical* information through intermediate relay nodes. This observation gives rise to a fundamental design question: how can we relay a secret key across a chain of potentially malicious classical nodes without compromising information-theoretic security? In the following, we introduce three progressively stronger network architectures that address this question.

Trusted Relay Nodes. The simplest approach is a linear chain of relay nodes, where each consecutive pair of nodes shares a QKD key. The sender $\mathcal{S}$ encrypts the secret key under the QKD key it shares with the first relay $P_{1,1}$, who decrypts it, re-encrypts it under the QKD key shared with $P_{2,1}$, and forwards it. This process repeats until the receiver $\mathcal{T}$ obtains the key from the last relay. This approach is illustrated in Fig. 1.

The critical weakness is that *every* relay node learns the key in the clear. A single corrupted node therefore breaks the security of the entire scheme. This is the standard trusted-node model deployed in current QKD networks, and it is precisely the trust assumption we aim to remove.

Naive Secret Sharing. A natural improvement is to split the key into n shares using a secret sharing scheme and transmit each share along an independent path. The sender $\mathcal{S}$ computes the shares and distributes them across n parallel relay chains; the receiver $\mathcal{T}$ collects the shares and reconstructs the key. This is illustrated in Fig. 2.

A single corrupted node can no longer recover the key—it only learns one share, which reveals no information about the secret. Moreover, corrupting multiple nodes on the same path yields no advantage, since every node on a given path handles the same share. However, an adversary who corrupts t nodes on

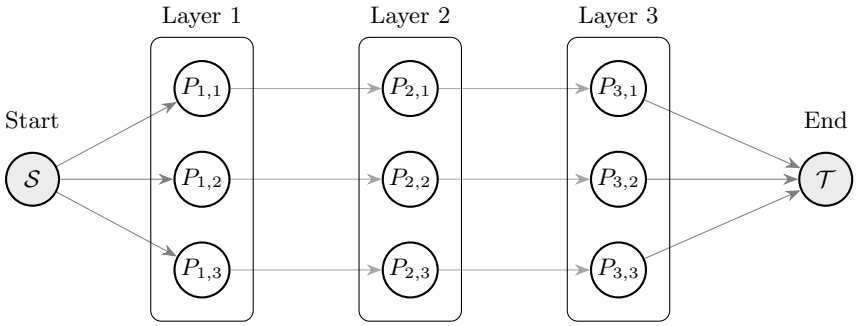

Fig. 2. The simple approach to secret sharing in a layered-hop QKD network: The source node performs the secret sharing and all subsequent nodes only forward the shares they have received from the previous node.

different paths obtains t distinct shares, which suffice to reconstruct the key. Crucially, because the shares are never re-randomised, the adversary needs only one corruption per path, regardless of the path length. The longer the network, the more likely such a cross-path attack becomes.

Proactive Secret Sharing. The weakness of the naive approach stems from the fact that shares remain static as they traverse the network: a share observed at any point along a path is identical to the share at every other point. To break this correlation, we let the nodes in each layer jointly *reshare* their shares before forwarding them to the next layer. When done properly, this ensures that the shares in any given layer are independently and uniformly distributed, conditioned only on reconstructing to the same secret.

This is illustrated in Fig. 3: each node $P_{a,i}$ splits its share into subshares and distributes them to all nodes in the next layer, who recombine the received subshares into fresh shares of the original secret.

Because each layer holds an independent sharing of the key, an adversary can no longer combine corruptions across layers. Recovering the key now requires corrupting t nodes within *the same* layer. Consequently, the protocol can tolerate up to $l \cdot (t-1)$ corrupted nodes in total, as long as no single layer contains t or more corruptions. In other words, security depends not on the total number of corrupted nodes in the network, but on the maximum number of corruptions in any individual layer.

3.2 Network Model

Having motivated the need for proactive resharing at each relay layer, we now formalize the network topology and notation used throughout the rest of this paper.

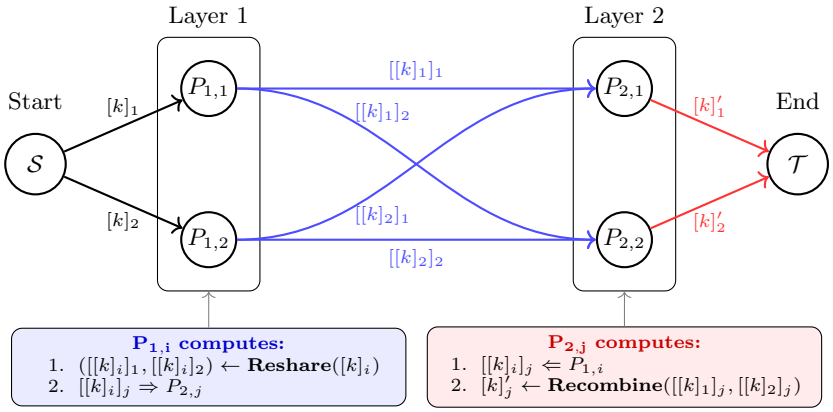

Fig. 3. Two-layer resharing protocol with two parties per layer. The sender distributes shares s_1, s_2 to Layer 1. Each node $P_{1,i}$ *reshares* its share into subshares $([[k]_i]_1, [[k]_i]_2)$ and sends $[[k]_i]_j$ to $P_{2,j}$ (blue edges). Each Layer 2 node $P_{2,j}$ *recombines* the subshares $[[k]_1]_j, [[k]_2]_j$ into a fresh share $[k]'_j$ of the original secret (red edges). Note that for a secure instantiation, all communication between nodes must be encrypted.

We consider QKD network topologies with a single sender and a single receiver, but potentially multiple *layers* of intermediate hops, in a feed-forward manner (i.e., edges only between consecutive layers).

We informally define a *layer* as follows: nodes are grouped into ordered sets ("layers") between the start node $\mathcal{S}$ and the end node $\mathcal{T}$. Layer 1 contains the nodes that can be reached in one hop from S, layer 2 contains nodes that can be reached in one additional hop from layer 1, and so on, until the last intermediate layer whose nodes connect directly to $\mathcal{T}$. We only consider forward connections, meaning that edges go from layer i to layer $i+1$.

Formally, we consider a directed acyclic graph with a source $\mathcal{S}$, a receiver $\mathcal{T}$ and l intermediate layers $L_1, \ldots, L_l$. Each layer L_a consists of n parties $P_{a,1}, \ldots, P_{a,n}$. $\mathcal{S}$ is connected to each node of L_1 and each node of L_l is connected to $\mathcal{T}$. Additionally, each pair of consecutive layers forms a complete bipartite graph: every node in L_a is connected to every node in L_{a+1}. This layout is depicted in Fig. 4.

Should a network have additional nodes in a single layer, the described structure can be obtained by randomly selecting a subset of n nodes from the layer and disregarding the remaining nodes. We will refer to graphs $\mathcal{G}$ that are of this form as *valid*.

Parties. We refer to the adversary as $\mathcal{A}$ and to a legitimate party as $P_{a,i}$, where a identifies the layer L_a to which the party belongs and i is the party's index within that layer. When the layer is clear from context, we write P_i. As a convention, a (resp. b) indexes a sending (resp. receiving) layer, and i (resp. j) indexes a party within that layer. Consecutive layers are numbered such that $b = a + 1$.

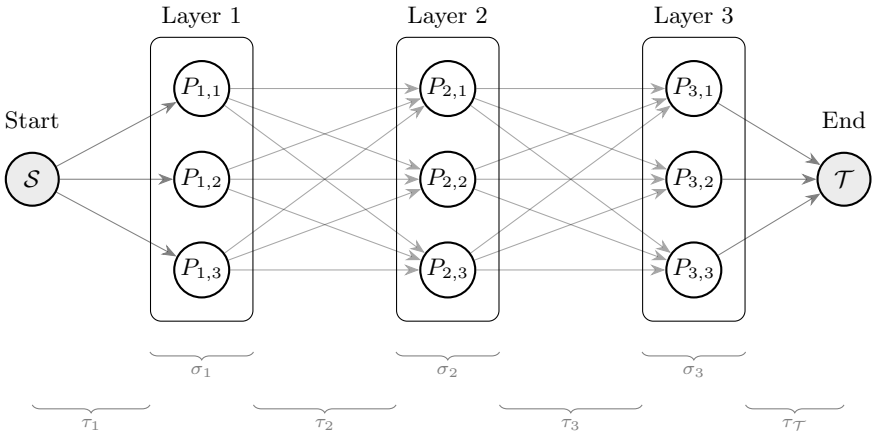

Fig. 4. Layered-hop network model with a single start node $\mathcal{S}$, a single end node $\mathcal{T}$, and l layers of intermediate relay nodes. Edges represent pairwise channels between consecutive layers. The transcript τ_a captures the messages received by layer a (with τ_1 for the initial distribution from $\mathcal{S}$ to L_1 and $\tau_{\mathcal{T}}$ for the delivery from L_l to $\mathcal{T}$). The layer state σ_a captures the internal state of all parties in layer a after receiving their incoming messages.

Secret Sharing. We employ the (t, n)-threshold proactive secret sharing scheme from Section 2.3, which tolerates up to $t-1$ corruptions per layer. In the layered setting, we write $[k]_{a,i}$ for the share held by party $P_{a,i}$ at layer L_a, and $[[k]_{a,i}]_j$ for the subshare that party $P_{a,i}$ sends to party $P_{a+1,j}$ during resharing (cf. Section 4).

Multi-Round Protocol. The key transport protocol executed on the network is a synchronous multi-round protocol with the following structure:

- **Initialization.** The sender $\mathcal{S}$ samples a uniformly random key k and splits it into n shares $([k]_{1,1}, \ldots, [k]_{1,n})$ using a secret sharing scheme.
- **First round.** $\mathcal{S}$ distributes one share to each party in L_1.
- **Intermediate rounds.** Each party $P_{a,i}$ in layer L_a uses a resharing scheme to split its share $[k]_{a,i}$ into n fragments and sends one fragment to each party in L_{a+1}. Each party in L_{a+1} recombines the received fragments into a fresh share of k.
- **Final round.** The receiver $\mathcal{T}$ collects n shares from the final layer and reconstructs k via **Reconstruct**.

The formal protocol syntax, which abstracts these steps into four composable algorithms and defines the associated state and transcript notation, is given in Section 3.3.

3.3 Protocol Syntax

We now formalise the multi-round structure described above as a parametric protocol syntax. A *layered hop-forwarding protocol* is a key-establishment protocol that transports a secret k from a sender $\mathcal{S}$ to a receiver $\mathcal{T}$ across a layered-hop network $\mathcal{G} = (\mathcal{S}, \mathcal{T}, L_1, \ldots, L_l)$, where each layer L_a consists of n parties $\{P_{a,1}, \ldots, P_{a,n}\}$ and each pair of adjacent layers forms a complete bipartite graph as visualized in Fig. 4. The protocol is parameterized by the network topology $\mathcal{G}$ and three protocol suites:

- A *short-range key-distribution* protocol **SKD** that establishes pairwise symmetric keys between every pair of adjacent parties (including $\mathcal{S}$ with L_1 and L_l with $\mathcal{T}$). The protocol syntax is agnostic to the specific key-distribution mechanism; any protocol that produces shared symmetric keys with a bounded adversarial advantage suffices. In the motivating application, **SKD** is instantiated with QKD, but the security analysis applies to any key-distribution protocol satisfying the same interface.
- A *channel protocol* **Ch** that provides secure point-to-point communication over each **SKD** key. All share and subshare transmissions between adjacent parties are protected by **Ch**.
- A *proactive secret sharing scheme* **PSS** that potentially splits, reshares, and reconstructs the secret across layers.

We write $\Pi[\mathcal{G}, \mathbf{SKD}, \mathbf{Ch}, \mathbf{PSS}]$ for the fully parameterized protocol.

The protocol $\Pi[\mathcal{G}, \mathbf{SKD}, \mathbf{Ch}, \mathbf{PSS}]$ consists of four algorithms. The *first-hop* algorithm

$$\Pi.\mathbf{FirstHop}(k) \longrightarrow (\tau_1,\ \sigma_1)$$

models the sender's role: it prepares initial data from k and distributes it to the parties of L_1 over their pairwise channels, producing the transcript τ_1 and the initial layer state σ_1. The *next-hop* algorithm

$$\Pi.\mathbf{NextHop}(\sigma_a) \longrightarrow (\tau_{a+1},\ \sigma_{a+1})$$

is executed jointly by the parties of L_a and L_{a+1}. It produces the transcript τ_{a+1} of the transition and the updated layer state σ_{a+1}. The *last-hop* algorithm

$$\Pi.\mathbf{LastHop}(\sigma_l) \longrightarrow (\tau_{\mathcal{T}},\ \sigma_{\mathcal{T}})$$

models the delivery to the receiver: the parties of L_l send their data to $\mathcal{T}$ over their pairwise channels, producing the transcript $\tau_{\mathcal{T}}$ and the receiver state $\sigma_{\mathcal{T}}$. The *finalisation* algorithm

$$\Pi.\mathbf{finalize}(\sigma_{\mathcal{T}}) \longrightarrow k'$$

is executed locally by the receiver $\mathcal{T}$: it recovers the secret k' from its internal state. Throughout, the transcript τ_a captures the messages *received* by layer a: τ_1 corresponds to the data distributed by $\mathcal{S}$ to L_1, τ_a for $a \in \{2, \ldots, l\}$ to the

messages sent from L_{a-1} to L_a, and $\tau_{\mathcal{T}}$ to the delivery from L_l to $\mathcal{T}$. We write $\tau_{a,i}$ for the messages received by party $P_{a,i}$, so that $\tau_a = \{\tau_{a,i}\}_{i\in[n]}$.

In the above formulae, the *layer state* $\sigma_a = \big(\{\mathsf{M}_{a,i},\ \Omega_{a,i},\ \tau_{a,i}\}_{i\in[n]}\big)$ bundles the internal memory, random tape, and incoming messages of each party on layer a. Since each hop algorithm is executed jointly by the parties of the corresponding layers, every party $P_{a,i}$ samples its random tape $\Omega_{a,i}$ locally as part of the joint execution; the layer state records these tapes alongside the memory and incoming messages. We write $\sigma_{a,i} \coloneqq (\mathsf{M}_{a,i},\ \Omega_{a,i},\ \tau_{a,i})$ for the state of party $P_{a,i}$, i.e., the projection of σ_a onto the components of party i. The *view* of party $P_{a,i}$ is defined as $\sigma_{a,i} \coloneqq \sigma_{a,i}$, comprising all information accessible to the party: its internal memory, random tape, and received messages. Outgoing messages are deterministically computable from the view. Each hop algorithm (**FirstHop**, **NextHop**, and **LastHop**) proceeds in two sequential phases:

1. **Communication phase.** Each party $P_{a,i}$ on the sending layer computes its outgoing messages from its memory $\mathsf{M}_{a,i}$ and random tape $\Omega_{a,i}$, and transmits them to the parties of L_{a+1} over their pairwise channels.
2. **Update phase.** Each party $P_{a+1,j}$ on the receiving layer updates its memory $\mathsf{M}_{a+1,j}$ based on the incoming messages $\tau_{a+1,j}$.

The two-phase structure is visualized in Fig. 3. The update phase must be completed before the next invocation of **NextHop**, i.e., the state σ_{a+1} is well-defined only after both phases have executed. The transcript τ_{a+1} comprises all communication observable by an external party during the transition from layer a to layer $a+1$.

Definition 3 (Correctness). *We consider a layered hop-forwarding protocol* $\Pi[\mathcal{G}, \mathbf{SKD}, \mathbf{Ch}, \mathbf{PSS}]$ *with* l *layers. It is* correct *if for every valid graph* $\mathcal{G}$ *and every* $k \in \{0,1\}^\lambda$*, it holds that*

$$\Pr\left[\, k' = k \;\middle|\; \begin{array}{l} (\tau_1,\, \sigma_1) \leftarrow \Pi.\mathbf{FirstHop}(k) \\ \textbf{for } a = 1, \ldots, l-1\colon (\tau_{a+1},\, \sigma_{a+1}) \leftarrow \Pi.\mathbf{NextHop}(\sigma_a) \\ (\tau_{\mathcal{T}},\, \sigma_{\mathcal{T}}) \leftarrow \Pi.\mathbf{LastHop}(\sigma_l) \\ k' \leftarrow \Pi.\mathbf{finalize}(\sigma_{\mathcal{T}}) \end{array} \right] = 1.$$

When the parameterization is clear from context, we simply write Π instead of $\mathit{protocol}[\mathcal{G}, \mathbf{SKD}, \mathbf{Ch}, \mathbf{PSS}]$.

3.4 Security Game

We now define the security notion for layered hop-forwarding protocols. The central question is: *does the adversary learn anything about the transported secret, beyond what is implied by the network topology?* To formalise this, we use a real-or-random indistinguishability game in which a two-stage adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ interacts with the protocol as follows. In the first stage, the adversary inspects the network topology $\mathcal{G}$ and commits to the set of parties it wishes to corrupt on each layer, reflecting the non-adaptive corruption model.

The game then executes the protocol on a uniformly random secret k_0 via a corruption-agnostic Run procedure and assembles the accumulated transcript τ, capturing all inter-layer communication, and the corrupted views $\sigma_{\mathcal{A}}$, comprising the complete internal state of every corrupted party. In the second stage, the adversary receives either the real secret k_0 or an independent uniformly random key k_1, along with its saved state, the transcript, and the corrupted views. It must then determine which key it was given. The protocol is key-secret if no adversary can distinguish the two cases better than by guessing.

Figure 5 presents the game formally. The game is parameterized by a protocol $\Pi[\mathcal{G}, \mathbf{SKD}, \mathbf{Ch}, \mathbf{PSS}]$ as defined above and applies to any protocol conforming to the syntax of Section 3.3.

We assume all instances of **SKD** were executed *before* the start of our protocol. As such, when $\mathcal{A}$ managed to obtain some of the exchanged keys, it starts the game with those keys in its view. It can then choose its corruptions with respect to this knowledge.

On the Semi-Honest Model. As detailed in Appendix A.2, theusual security model for proactive secret sharing cannot fit those types of protocol. The current work models the adversary as semi-honest (honest-but-curious): corrupted nodes follow the protocol faithfully but attempt to extract information from their collective views. This choice is motivated by both the application setting and the state of the literature. The semi-honest model directly captures the threat that the trusted-node assumption is meant to address: an adversary with persistent read access to a relay node's internal state. Our protocol weakens this trust assumption from full confidentiality of each relay to honest execution of the protocol, which is a strictly weaker requirement. The non-adaptive corruption model is similarly justified by the protocol's execution characteristics. Once QKD keys have been pre-established, the key-transport phase runs at the speed of classical communication—on the order of milliseconds per hop. Adaptively compromising a node during this window is significantly harder than maintaining passive access to a node compromised before deployment. The non-adaptive model, in which the adversary commits all corruptions before execution, reflects this operational constraint. We discuss the path toward active adversaries in Appendix C.

Definition 4 (LH-Key-secrecy). *Let $\Pi[\mathcal{G}, \mathbf{SKD}, \mathbf{Ch}, \mathbf{PSS}]$ be a layered hop-forwarding protocol over a valid graph $\mathcal{G}$, and* **SKD**, **Ch**, **PSS** *respectively be correct short-range key distribution, channel, and proactive secret sharing protocols. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary playing the security game defined in Fig. 5 where $\mathcal{A}_1$ outputs a corruption set $\{\mathcal{I}_a\}_{a\in[l]}$, where each $\mathcal{I}_a \subseteq \{i : P_{a,i} \in L_a\}$ contains at most $t-1$ elements. We define the* lh-key-secrecy advantage *of $\mathcal{A}$ against Π to be*

$$\mathbf{Adv}^{\textit{lh-key-secrecy}}_{\Pi,\mathcal{A}}(\lambda) \coloneqq \left| \Pr\left[\mathsf{Exp}^{\mathsf{lh-key-secrecy}}_{\Pi,\mathcal{A}}(\lambda) = 1 \right] - \frac{1}{2} \right|.$$

We say that Π is key-secret *if, for every (computationally unbounded) adversary $\mathcal{A}$, the advantage $\mathbf{Adv}^{\textit{lh-key-secrecy}}_{\Pi,\mathcal{A}}(\lambda)$ is zero.*

$\mathsf{Exp}^{\mathsf{lh-key-secrecy}}_{\Pi[\mathcal{G},\mathbf{SKD},\mathbf{Ch},\mathbf{PSS}],\mathcal{A}}(\lambda)$:

▷ commit corruption sets
$(st,\ \{\mathcal{I}_a\}_{a\in[l]}) \leftarrow \mathcal{A}_1(\mathcal{G})$
▷ sample challenge secrets
$k_0 \xleftarrow{\$} \{0,1\}^\lambda; k_1 \xleftarrow{\$} \{0,1\}^\lambda$
▷ choose challenge bit
$b \xleftarrow{\$} \{0,1\}$
▷ execute protocol
$\left(\{\tau_a\}_{a\in[l]},\ \tau_\mathcal{T},\ \{\sigma_a\}_{a\in[l]},\ \sigma_\mathcal{T}\right) \leftarrow \mathsf{Run}(k_0)$
▷ assemble transcript
$\tau \leftarrow \tau_1 \parallel \cdots \parallel \tau_l \parallel \tau_\mathcal{T}$
▷ extract corrupted views
$\sigma_\mathcal{A} \leftarrow \bigcup_{a\in[l]} \{\sigma_{a,i}\}_{i\in\mathcal{I}_a}$
▷ adversary guesses
$b' \leftarrow \mathcal{A}_2(st,\ k_b,\ \tau,\ \sigma_\mathcal{A})$
return $[\![b = b']\!]$

$\mathsf{Run}(k)$:

▷ first hop
$(\tau_1,\ \sigma_1) \leftarrow \Pi.\mathbf{FirstHop}(k)$
▷ forward through layers
for $a = 1$ **to** $l-1$:
  $(\tau_{a+1},\ \sigma_{a+1}) \leftarrow \Pi.\mathbf{NextHop}(\sigma_a)$
▷ last hop
$(\tau_\mathcal{T},\ \sigma_\mathcal{T}) \leftarrow \Pi.\mathbf{LastHop}(\sigma_l)$
return $\left(\{\tau_a\}_{a\in[l]},\ \tau_\mathcal{T},\ \{\sigma_a\}_{a\in[l]},\ \sigma_\mathcal{T}\right)$

Fig. 5. Real-or-random security game for layered secret transport. The two-stage adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ first commits corruption sets $\{\mathcal{I}_a\}$ (each of size at most $t-1$); the game runs the protocol on k_0, assembles the transcript τ and corrupted views $\sigma_\mathcal{A}$, then challenges $\mathcal{A}_2$ to distinguish the real key (b=0) from an ideal key (b=1).

4 Instantiating the Proactive Secret Sharing

The concrete sub-protocols for **Reshare** and **Recombine** are detailed in Figs. 6 and 7, respectively. These protocols are closely related to [Wan11]; we nonetheless include full descriptions and proofs of correctness, privacy, and liveness.

4.1 Correctness

Intuitively, both our sub-protocols are correct if, after the execution of **Reshare** by any subset of at least t parties in L_a, the result of **Recombine** satisfies the input requirement for **Reshare**, for every $P_{a+1,j} \in L_{a+1}$. Notice that, because the hiding polynomial is required to be the same for all party in a same level, it implies the rerandomised share would reconstruct to the valid secret.

Theorem 1 (Correctness of *reshare and recombine*). *Consider a correct (t,n)-threshold secret sharing protocol, of which $\{[k]_{a,i}\}_{\forall i\in[n]}$ are valid shares of a secret k. Let every $\{\omega_{a,i,q}\}_{\forall 1\le q\le t-1}$ be chosen uniformly at random over $\mathbb{F}$. Let us split, for any i in a subset $\mathcal{I}$ of size at least t, each $[k]_{a,i}$ in sub shares $\{[[k]_{a,i}]_j\}_{\forall j\in[\overline{n}]}$ (with $\overline{n} > t-1$) through*

$$[[k]_{a,i}]_j \coloneqq \left(\prod_{z=1}^{t-1} \frac{j-z}{-z}\right) \cdot [k]_{a,i} + \sum_{q=1}^{t-1} \left(\prod_{z=0, z\neq q}^{t-1} \frac{j-z}{q-z} \right) \cdot \omega_{a,i,q}$$

INPUT: A session id ID, two adjacent layers L_a and L_b, (L_b depends on L_a and ID) of size n, m resp.; list of parties for both layer, $\{P_{a,i}\}_{i\in[n]}$ and $\{P_{b,j}\}_{j\in[m]}$ resp.; private shares for all $P_{a,i}$, $[k]_{a,i} = Q_a(i) = \sum_{q=1}^{t-1} \alpha_{a,q} \cdot i^q$, where the secret k is hidden at $Q_a(0) = k$.

For each $\mathbf{P_{a,i}}$:

1. $\omega_{a,i,q} \xleftarrow{\$} \mathbb{F}$; $(\forall q \in [1, t-1])$
2. $R_{a,i}(x) \leftarrow (\prod_{z=1}^{t-1} \frac{x-z}{-z}) \cdot [k]_{a,i} + \sum_{q=1}^{t-1} (\prod_{\substack{z=0 \\ z\neq q}}^{t-1} \frac{x-z}{q-z}) \cdot \omega_{a,i,q}$
3. $[[k]_{a,i}]_j \leftarrow R_{a,i}(j)$; $(\forall j \in [m])$
4. $([[k]_{a,i}]_j, \mathrm{ID}, i, j) \Rightarrow P_{b,j}$

Fig. 6. Reshare sub-protocol.

Input: a session id ID, two adjacent layers L_a and L_b, $b = a+1$, of size n, m resp.; list of parties for both layer, $\{P_{a,i}\}_{i\in[n]}$ and $\{P_{b,j}\}_{j\in[m]}$ resp.
For each $\mathbf{P_{b,j}}$:

1. **Do**
2. $\quad [[k]_{a,i}]_j, \mathrm{ID}, i, j \Leftarrow P_{a,i}$
3. $\quad \mathcal{I} \leftarrow \{i : \forall j,\ [[k]_{a,i}]_j \Leftarrow P_{a,i}\}$
4. **WaitUntil**$(|\mathcal{I}| \geq t)$
5. $[k]_{b,j} \leftarrow \sum_{i\in\mathcal{I}} (\prod_{\substack{q\in\mathcal{I} \\ q\neq i}} \frac{-q}{i-q}) [[k]_{a,i}]_j$

Fig. 7. Recombine sub-protocol.

and recombine as per

$$[k]_{b,j} = \sum_{i\in\mathcal{I}} \Big(\prod_{q\in\mathcal{I}, q\neq i} \frac{-q}{i-q} \Big) [[k]_{a,i}]_j$$

Then, we can equivalently write $[k]_{b,j} = Q_b(j)$, where $Q_b(x)$ is independent of $P_{b,j}$, and yields $Q_b(0) = k$.

Proof. Refer to Appendix B.1. □

4.2 $(t-1)$-Secrecy of Individual (Sub-)Shares

The secrecy of our protocol requires, first of all, the well-known secrecy result of basic Shamir secret shares [Sha79]. That is, consider any correct (t, n)-threshold Shamir secret sharing protocol, and assume the knowledge of $t-1$ of its shares. Given this limited knowledge, there exists an infinite number of polynomials $Q_{\text{false}}(x)$ that could have resulted in those t shares. Moreover, considering the secret is stored at $x = x_k$, the values of every possible $Q_{\text{false}}(x_k)$ are uniformly distributed over $\mathbb{F}$.

Theorem 2 (Secrecy of the sub-shares). *Consider correct (t, n)-threshold secret sharing protocol, of which $\{[k]_{a,i}\}_{\forall i\in[n]}$ are valid shares of a secret k.*

Let every $\{\omega_{a,i,q}\}_{\forall 1 \leq q \leq t-1}$ *be chosen uniformly at random over* $\mathbb{F}$*, and define* $R_{a,i}(x) \coloneqq \gamma^{(x)}_{0,[0;t-1]}[k]_{a,i} + \sum_{q=1}^{t-1} \gamma^{(j)}_{q,[0;t-1]}\omega_{a,i,q}$.

Let us split, for all $P_{a,i} \in L_a$*, each* $[k]_{a,i}$ *in sub shares* $\{[[k]_{a,i}]_j\}_{\forall j \in [|L_b|]}$ *as*

$$[[k]_{a,i}]_j \coloneqq R_{a,i}(j).$$

Given the view of at most $t-1$ *parties on each layer, no information can be computed on the secret* k*.*

Refer to Appendix B.2.

Proof. □

5 Protocol Instantiation

We instantiate the layered hop-forwarding that has been motivated and formalized in Section 3.3 with the following concrete choices:

- Recall that **SKD** is the short-range key-distribution protocol used to establish pairwise symmetric keys between adjacent parties (cf. Section 3.3). We set **SKD** $\coloneqq$ **QKD**, quantum key distribution. Every pair of adjacent parties runs a QKD protocol to establish a pairwise symmetric key whose security rests on the laws of quantum mechanics rather than computational assumptions. Parties will use this key for their subsequent communication.
- The channel protocol **Ch** secures all point-to-point communication over the keys produced by **SKD**. We set **Ch** $\coloneqq$ **OTP**, the one-time pad. Each QKD-established key is consumed as a one-time pad to encrypt exactly one share or subshare in transit, yielding a channel that is perfectly secret against any eavesdropper regardless of computational power. This preserves the information-theoretic security guarantee of the underlying QKD keys at every hop. The combination of QKD key generation with OTP encryption ensures that the overall protocol never relies on computational hardness assumptions[6].
- The proactive secret sharing scheme **PSS** handles splitting, resharing, and reconstruction of the secret across layers. We set **PSS** $\coloneqq$ **SPSS**, our proactive secret sharing scheme based on Shamir's (t, n)-threshold secret sharing. At each layer transition, every party reshares its current share into fresh subshares for the next layer, and the receiving parties recombine these subshares into new shares of the same secret. This proactive resharing ensures that an adversary must compromise at least t parties *within the same layer* to learn anything about the secret; corruptions across different layers cannot be combined. Shamir's scheme is chosen over additive sharing because it natively supports threshold reconstruction, allowing the receiver to recover the secret

[6] In adversarial models where the attacker can modify messages in transit, one would replace the plain one-time pad with a Carter–Wegman authenticated one-time pad to additionally guarantee integrity.

from only t of the shares. Its ($\mathbf{Share}, \mathbf{Reshare}, \mathbf{Recombine}, \mathbf{Reconstruct}$) are detailed above.

The resulting protocol is $\Pi[\mathcal{G}, \mathbf{QKD}, \mathbf{OTP}, \mathbf{SPSS}]$. While [BS08] considered additive secret sharing, we employ Shamir's scheme because it natively supports threshold reconstruction.

5.1 Formal Description

We now describe our instantiated protocol, $\Pi[\mathcal{G}, \mathbf{QKD}, \mathbf{OTP}, \mathbf{SPSS}]$.

Pre-computation. Every pair of adjacent parties (including the sender $\mathcal{S}$ and receiver $\mathcal{T}$ with their respective neighbouring layers) runs **QKD** to establish a pairwise symmetric key. All subsequent communication is protected by **OTP** over the resulting keys.

First hop. The sender $\mathcal{S}$ generates a (t, n)-threshold Shamir sharing of the secret and distributes one share to each party of the first layer. Concretely, $\Pi.\mathbf{FirstHop}(k)$ proceeds as follows:

1. $\mathcal{S}$ computes $([k]_{1,1}, \ldots, [k]_{1,n}) \xleftarrow{\$} \mathbf{SPSS}.\mathbf{Share}(k, n, t)$.
2. For each $i \in [n]$, $\mathcal{S}$ sends $[k]_{1,i}$ to $P_{1,i}$ via **OTP**.
3. Each party $P_{1,i}$ stores $\mathsf{M}_{1,i} \coloneqq [k]_{1,i}$.

The resulting shares lie on a polynomial of degree $t-1$, so any t of them suffice to recover k. The transcript τ_1 captures the communication from $\mathcal{S}$ to L_1.

Next hop. The transition from layer a to layer $a+1$ (for $a \in [l-1]$) is realised by **SPSS**.**Reshare** followed by **SPSS**.**Recombine**. $\Pi.\mathbf{NextHop}(\sigma_a)$ proceeds as follows:

1. **Communication phase.** Each party $P_{a,i}$ computes

$$([[k]_{a,i}]_1, \ldots, [[k]_{a,i}]_n) \xleftarrow{\$} \mathbf{SPSS}.\mathbf{Reshare}([k]_{a,i})$$

   and, for each $j \in [n]$, sends $[[k]_{a,i}]_j$ to $P_{a+1,j}$ via **OTP**.
2. **Update phase.** Each party $P_{a+1,j}$ receives subshares $\{[[k]_{a,i}]_j\}_{i\in[n]}$ and computes its new share:

$$[k]_{a+1,j} \leftarrow \mathbf{SPSS}.\mathbf{Recombine}\big(\{[[k]_{a,i}]_j\}_{i\in[n]}\big)\,.$$

   It stores $\mathsf{M}_{a+1,j} \coloneqq [k]_{a+1,j}$.

The transcript τ_{a+1} captures the communication from L_a to L_{a+1}.

Last hop. The final layer delivers shares directly to the receiver without resharing. $\Pi.\mathbf{LastHop}(\sigma_l)$ proceeds as follows:

1. Each party $P_{l,i}$ sends its share $[k]_{l,i}$ to $\mathcal{T}$ via **OTP**.
2. $\mathcal{T}$ stores $\mathsf{M}_{\mathcal{T}} \coloneqq \{[k]_{l,i}\}_{i\in[n]}$.

The transcript $\tau_{\mathcal{T}}$ captures the communication from L_l to $\mathcal{T}$.

Finalisation. The receiver holds $\mathsf{M}_{\mathcal{T}} = \{[k]_{l,i}\}_{i\in[n]}$, from which any t shares suffice to recover the secret. $\Pi.\mathbf{finalize}(\sigma_{\mathcal{T}})$ computes:

$$k' \leftarrow \mathbf{SPSS}.\mathbf{Reconstruct}(\mathsf{M}_{\mathcal{T}})\,.$$

Correctness. The instantiation satisfies the correctness requirement of Definition 3. By definition of **SPSS**.**Share**, the first hop produces valid (t, n)-threshold Shamir shares of k. Each next hop applies **SPSS**.**Reshare** and **SPSS**.**Recombine**; by Theorem 1, the resulting shares $\{[k]_{a+1,j}\}_{j\in[n]}$ lie on a degree-$(t-1)$ polynomial with the same constant term k. This invariant is preserved across all l layers, and the finalisation step thus correctly recovers k.

5.2 Security of the Protocol

We now derive the adversarial advantage in winning the experiment defined in Fig. 5, $\mathsf{Exp}^{\mathsf{lh-key-secrecy}}_{\Pi[\mathcal{G},\mathbf{QKD},\mathbf{OTP},\mathbf{SPSS}],\mathcal{A}}(\lambda)$.

Theorem 3. *Let* **QKD** *be a quantum key distribution protocol parameterised by a security parameter λ, and let* **OTP** *be the one-time pad using a* **QKD** *key. Write* $\mathbf{SPSS} \coloneqq (\mathbf{Share}, \mathbf{Reshare}, \mathbf{Recombine}, \mathbf{Reconstruct})$ *for our (t, n)-threshold proactive secret sharing protocol. Consider a network $\mathcal{G}$ with l layers and n parties per layer. Then, for any unbounded adversary $\mathcal{A}$ corrupting at most $t-1$ parties per layer,*

$$\mathbf{Adv}^{\textit{lh-key-secrecy}}_{\Pi[\mathcal{G},\mathbf{QKD},\mathbf{OTP},\mathbf{SPSS}],\mathcal{A}}(\lambda) \leq 1 - (1-\epsilon)^{2n+(l-1)n^2},$$

where ϵ is the adversarial advantage against **QKD**.

Proof Sketch. The proof proceeds by a sequence of game hops (Appendix B.3). The first hop replaces each QKD key by an ideal key, incurring at most ϵ loss per key instance; the product bound "$1-(1-\epsilon)^{2n+(l-1)n^2}$" accounts for each instance independently and is therefore tight.

With ideal keys in place, the one-time pad renders all ciphertexts perfectly random, and the remaining security reduces to the information-theoretic privacy of the proactive secret sharing scheme (Theorem 2), which guarantees zero advantage when fewer than t parties per layer are corrupted. □

References

ABN+25. Shahla Atapoor, Karim Baghery, Georgio Nicolas, Robi Pedersen, and Jannik Spiessens. Batched and packed (publicly) verifiable secret sharing: A unified framework and applications. Cryptology ePrint Archive, Paper 2025/2018, 2025.

AEE+23. Koji Azuma, Sophia E. Economou, David Elkouss, Paul Hilaire, Liang Jiang, Hoi-Kwong Lo, and Ilan Tzitrin. Quantum repeaters: From quantum networks to the quantum internet. *Rev. Mod. Phys.*, 95:045006, Dec 2023.

AVY24. Nicolas Alhaddad, Mayank Varia, and Ziling Yang. Haven++: Batched and packed dual-threshold asynchronous complete secret sharing with applications. *CiC*, 1(4):3, 2024.

BB14. Charles H. Bennett and Gilles Brassard. Quantum cryptography: Public key distribution and coin tossing. *Theoretical Computer Science*, 560:7–11, December 2014.

BBM92. Charles H. Bennett, Gilles Brassard, and N. David Mermin. Quantum cryptography without bell's theorem. *Phys. Rev. Lett.*, 68:557–559, Feb 1992.

BDCZ98. H.-J. Briegel, W. Dür, J. I. Cirac, and P. Zoller. Quantum repeaters: The role of imperfect local operations in quantum communication. *Phys. Rev. Lett.*, 81:5932–5935, Dec 1998.

BELO15. Joshua Baron, Karim El Defrawy, Joshua Lampkins, and Rafail Ostrovsky. Communication-optimal proactive secret sharing for dynamic groups. In Tal Malkin, Vladimir Kolesnikov, Allison Bishop Lewko, and Michalis Polychronakis, editors, *ACNS 2015*, volume 9092 of *LNCS*, pages 23–41, New York, NY, USA, June 2–5, 2015. Springer, Cham, Switzerland.

BP11. S. M. Barnett and S. J. D. Phoenix. Securing a quantum key distribution relay network using secret sharing. In *2011 IEEE GCC Conference and Exhibition (GCC)*, page 143–145. IEEE, February 2011.

BS08. Travis R. Beals and Barry C. Sanders. *Distributed Relay Protocol for Probabilistic Information-Theoretic Security in a Randomly-Compromised Network*, page 29–39. Springer Berlin Heidelberg, 2008.

CZC+21. Yu-Ao Chen, Qiang Zhang, Teng-Yun Chen, Wen-Qi Cai, Sheng-Kai Liao, Jun Zhang, Kai Chen, Juan Yin, Ji-Gang Ren, Zhu Chen, et al. An integrated space-to-ground quantum communication network over 4,600 kilometres. *Nature*, 589(7841):214–219, 2021.

DH76. W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

DS02. Paolo D'Arco and Douglas R. Stinson. On unconditionally secure robust distributed key distribution centers. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 346–363, Queenstown, New Zealand, December 1–5, 2002. Springer Berlin Heidelberg, Germany.

Eke91. Artur K. Ekert. Quantum cryptography based on bell's theorem. *Phys. Rev. Lett.*, 67:661–663, Aug 1991.

FKLR25. Nicole Franzoi, Stephan Krenn, Thomas Lorunser, and Sebastian Ramacher. Secure key forwarding for large-scale quantum key distribution networks. In *2025 International Conference on Quantum Communications, Networking, and Computing (QCNC)*, pages 486–493, 2025.

FNR23. Brett Hemenway Falk, Daniel Noble, and Tal Rabin. Proactive secret sharing with constant communication. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023, Part II*, volume 14370 of *LNCS*, pages 337–373, Taipei, Taiwan, November 29 – December 2, 2023. Springer, Cham, Switzerland.

GKS+26. Antoine Gansel, Juliane Krämer, Tim Schumacher, Patrick Struck, Maximilian Tippmann, and Thomas Walther. Field-tested authentication for quantum key distribution and dos attacks. In Mark Manulis, editor, *Applied Cryptography and Network Security Workshops*, pages 245–265, Cham, 2026. Springer Nature Switzerland.

GMK22. Andrei Gaidash, George Miroshnichenko, and Anton Kozubov. Quantum network security dependent on the connection density between trusted

nodes. *Journal of Optical Communications and Networking*, 14(11):934–943, 2022.

HHB+25. P. Horoschenkoff, J. Henrich, R. Böhn, I. Khan, J. Rödiger, M. Gunkel, M. Bauch, J. Benda, P. Bläcker, E. Eichhammer, U. Eismann, G. Frenck, H. Griesser, W. Jontofsohn, N. Kopshoff, S. Röhrich, F. Seidl, N. Schark, E. Sollner, D. von Blanckenburg, A. Heinemann, M. Stiemerling, and M. Gärtner. Demoquandt: a carrier-grade qkd network. *Journal of Optical Communications and Networking*, 17(9):743, August 2025.

HJKY95. Amir Herzberg, Stanisław Jarecki, Hugo Krawczyk, and Moti Yung. Proactive secret sharing or: How to cope with perpetual leakage. In Don Coppersmith, editor, *Advances in Cryptology — CRYPT0' 95*, pages 339–352, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

KRBM07. Robert König, Renato Renner, Andor Bariska, and Ueli Maurer. Small Accessible Quantum Information Does Not Imply Security. *Physical Review Letters*, 98(14):140502, 2007. Publisher: American Physical Society.

LCL+17. Sheng-Kai Liao, Wen-Qi Cai, Wei-Yue Liu, Liang Zhang, Yang Li, Ji-Gang Ren, Juan Yin, Qi Shen, Yuan Cao, Zheng-Ping Li, Feng-Zhi Li, Xia-Wei Chen, Li-Hua Sun, Jian-Jun Jia, Jin-Cai Wu, Xiao-Jun Jiang, Jian-Feng Wang, Yong-Mei Huang, Qiang Wang, Yi-Lin Zhou, Lei Deng, Tao Xi, Lu Ma, Tai Hu, Qiang Zhang, Yu-Ao Chen, Nai-Le Liu, Xiang-Bin Wang, Zhen-Cai Zhu, Chao-Yang Lu, Rong Shu, Cheng-Zhi Peng, Jian-Yu Wang, and Jian-Wei Pan. Satellite-to-ground quantum key distribution. *Nature*, 549(7670):43–47, Sep 2017.

PLOB17. Stefano Pirandola, Riccardo Laurenza, Carlo Ottaviani, and Leonardo Banchi. Fundamental limits of repeaterless quantum communications. *Nature Communications*, 8(1):15043, April 2017.

Ren06. Renato Renner. *Security of Quantum Key Distribution*. PhD thesis, arXiv, 2006. arXiv:quant-ph/0512258.

Sha79. Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

Wan11. Xiuqun Wang. A novel adaptive proactive secret sharing without a trusted party. Cryptology ePrint Archive, Paper 2011/241, 2011.

WLX+23. Mingjun Wang, Jian Li, Kaiping Xue, Ruidong Li, Nenghai Yu, Yangyang Li, Yifeng Liu, Qibin Sun, and Jun Lu. A segment-based multipath distribution method in partially-trusted relay quantum networks. *IEEE Communications Magazine*, 61(12):184–190, 2023.

WZW+21. Liu-Jun Wang, Kai-Yi Zhang, Jia-Yong Wang, Jie Cheng, Yong-Hua Yang, Shi-Biao Tang, Di Yan, Yan-Lin Tang, Zhen Liu, Yu Yu, et al. Experimental authentication of quantum key distribution with post-quantum cryptography. *npj quantum information*, 7(1):67, 2021.

ZLHM22. Hongyi Zhou, Kefan Lv, Longbo Huang, and Xiongfeng Ma. Quantum network: Security assessment and key management. *IEEE/ACM Trans. Netw.*, 30(3):1328–1339, January 2022.

A Additional Background

A.1 Quantum Key Distribution

Quantum Key Distribution (QKD) refers to every protocol that uses the laws of quantum mechanics to distribute shared randomness and generate symmetric cryptographic keys between two or more entities.

As such, QKD protocols involve two main steps. In the first step, Alice and Bob send, receive, and measure quantum states, giving them pairs of partially correlated (and potentially corrupted) bit strings. In the second step, namely the classical post-processing, they turn this raw data into a secure symmetric key. In this second phase, the parties collaborate to eliminate errors that occurred and correlations Eve might have gained during the communication on the quantum-physical layer. The classical communication, in particular, is akin to a key derivation function in classical key exchange. Its role is to estimate the modification the adversary applied on the quantum state when looking at it, such as to guess (and eliminate from the final key) the amount of information it may have learned.

Definition 5 (Simplified Security of QKD [Ren06,KRBM07]). *Let us denote by ρ_{ABE} (resp. ρ_{ideal}) the quantum probability distribution (i.e., density matrix) on all quantum and classical messages sent, received or observed by Alice (A), Bob (B) and the adversary (E) during a real (resp. ideal) QKD protocol.*

A QKD protocol is ϵ-secure if the statistical distance between ρ_{ABE} and ρ_{ideal} is upper bounded by epsilon,

$$\|\rho_{ABE} - \rho_{ideal}\|_1 \leq \epsilon,$$

where $\|\cdot\|_1$ is the trace norm (analogous to the classical Kolmogorov distance).

This can equivalently be defined by a requirement on the key length λ extractable from n quantum signals,

$$\lambda \leq \mathcal{O}(nH(\rho_{AB}|E))^7.$$

Although it is not visible in the simplified definition, the key length λ and the security parameters ϵ are directly related. ϵ can thus be chosen freely, resulting in $\lambda = 0$ in the worst case.

A.2 Parallel with the Usual Proactive Secret Sharing Model

The proximity between our use case and proactive secret sharing is evident from the discussions above. However, the two use cases differ in ways that influence which security model is possible. Mobile adversaries, in particular, hardly make sense in this context.

[7] A concrete formula (that is, valid for a specific type of QKD protocols) can be found in [Ren06, Lem. 6.5.1]

Remember that the security model for proactive secret sharing always assumes an active adversary. In PSS, the protocol proceeds in *epochs*: discrete time frames during which a designated group of parties, called a *committee*, collectively holds the shares of the secret. Within each epoch, $\mathcal{A}$ is allowed to corrupt up to $t-1$ out of n committee members. Between epochs, corrupted parties can be rebooted, eliminating any adversarial presence from their systems before the next committee takes over.

From there, we need to distinguish some major differences. Most notable is the question of execution time. While proactive secret sharing's epochs are not required to last any specific amount of time, their approximate duration and the number of parties composing it remain the only required information at the start of the protocol. Which node takes part in a specific committee may remain completely unknown until the start of the related reshare phase.

In contrast, our layers are mainly defined by their spatial positions, which instructs on the possible members composing each layer. The start and end points of our protocol are indeed publicly known. Moreover, the necessary restriction on the communication distance, inherited from the QKD protocol, allows one to thin down the possible members of each committee to a quite minimal set of nodes.

An adversary allowed to corrupt $t-1$ new parties each time we move to a new layer could focus its efforts on the space around $\mathcal{T}$, which is likely to quickly result in more than $t-1$ corruption in the effective final layer. Contrary to proactive secret sharing, it is not possible here to argue that the adversary needs to "free" its corrupted nodes to attack another committee. This assumption would require the ability to reboot the nodes before we communicate with them, which would significantly delay communication. While this would be a theoretically viable solution, such a protocol would be of no practical interest for a key exchange.

As such, the traditional proactive secret-sharing security model needs to be slightly modified; we align it closely with standard key-exchange security models.

A.3 Cross-Layer Corruption

A natural question in the layered setting is whether an adversary gains additional power by distributing corruptions across adjacent layers, combining partial information from senders and receivers to increase the number of known shares beyond the per-layer corruption budget. We show, through the example of [BS08], what components lead to such an attack, and derive the requirement to avoid it. Note that, although they consider additive secret sharing instead of Shamir's secret sharing, this has no impact on the theoretical discussion.

The Issue of Distributed Randomness. Although the construction of Beals and Senders [BS08] is inspired by proactive secret sharing, it is closer to the naive protocol presented in this paper (cf. Fig. 2). Each party $P_{a,i} \in L_a$ communicates only with its positional counterpart on the adjacent layers, $P_{a-1,i}$ and $P_{a+1,i}$, rather than with all parties in those layers. Because of this restricted communication pattern, the nodes within a layer must coordinate their randomness to

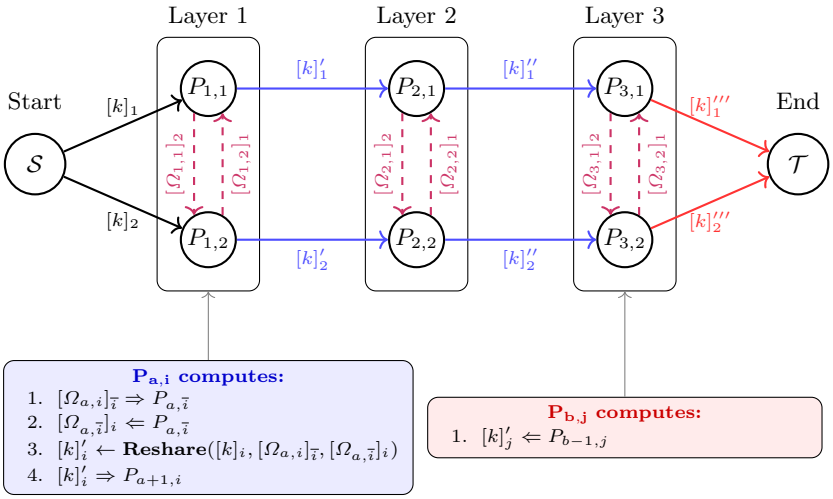

Fig. 8. Three-layer resharing protocol with two parties per layer, vulnerable to cross-layer collaboration. The sender distributes shares s_1, s_2 to Layer 1. All node $P_{1,i}$ *agrees* on some shared randomness Γ (purple, dotted edges), and use it to send *rerandomised* $[k]'_i$ to $P_{2,i}$ (blue edges).

produce a valid re-randomised sharing of k (we abstract this resharing protocol in Fig. 8).

The vulnerability is apparent from Fig. 8: before re-randomisation, each party in a given layer holds a copy of the share it received from its positional counterpart on the previous layer. An adversary corrupting party $P_{a,i}$ and party $P_{a+1,i}$ therefore sees the share both before and after re-randomisation, effectively learning the applied randomness and gaining information beyond the per-layer corruption budget.

Solving Cross-Layer Eavesdropping. Intuitively, the above issue can be mitigated by synchronising the resharing's randomness in the receiving layer, instead of in the sending layer. We distill the necessary and sufficient conditions into the following claim, which is an informal restatement of the properties formally captured and proven in Theorem 2.

Claim ($(t-1)$-secrecy of resharing protocols)**.** *A resharing protocol is secure against $t-1$ corruptions in each of two adjacent layers if and only if*

1. *one needs t share fragments to reconstruct both the original and the rerandomised share,*
2. *the share fragments possessed by a party give no information on the share owned by parties on the same layer, and each fragment is only related to one party's share in the sending layer,*
3. *none of the received share fragment may relate to a same share.*

Looking at [BS08] through this lens, one could improve their secrecy threshold to $t-1$ corruptions per layer. Their protocol indeed requires additively sharing a random value to all other parties, to then synchronise the applied rerandomisation by adding their random value and the other parties' random value shares:

$$[k]'_{a,i} = [k]_{a,i} \oplus \left(\bigoplus_{\forall \bar{i}} [\Omega_{a,i}]_{\bar{i}} \oplus [\Omega_{a,\bar{i}}]_i \right).$$

This operation, however, does not need to be performed on the sending layer. It could have been equivalently implemented (in terms of correctness) by having all $P_{a,i}$ sending $[k]_{a,i} \oplus (\bigoplus_{\forall \bar{i}} [\Omega_{a,i}]_{\bar{i}})$ to $P_{a+1,i}$, and $[\Omega_{a,i}]_{\bar{i}}$ to each $P_{a+1,\bar{i}}$.

B Postponed Proofs

B.1 Proof of Theorem 1

For all $i \in [n]$, consider any set of indices $\mathcal{I}$, $|\mathcal{I}| \geq t$, and for readability's sake, let's write the Lagrange interpolation parameter as $\gamma^{(x)}_{y,\mathcal{S}} = \prod_{q \in \mathcal{S}, q \neq y} \frac{x-q}{y-q}$. We write $Q_a(x)$ the polynomial such that $Q_a(i) = [k]_{a,i}$
Using Lagrange's interpolation, we get the following:

$$\sum_{i \in \mathcal{I}} \gamma^{(0)}_{i,\mathcal{I}} [[k]_{b,j}]_i = \sum_{i \in \mathcal{I}} \gamma^{(0)}_{i,\mathcal{I}} \sum_{q=0}^{t-1} \gamma^{(j)}_{q,[0;t-1]} \omega_{a,i,q} \tag{3a}$$

$$= \gamma^{(j)}_{0,[0;t-1]} \sum_{i \in \mathcal{I}} \gamma^{(0)}_{i,\mathcal{I}} [k]_{a,i} + \sum_{q=1}^{t-1} \gamma^{(j)}_{q,[0;t-1]} \sum_{i \in \mathcal{I}} \gamma^{(0)}_{i,\mathcal{I}} \omega_{a,i,q} \tag{3b}$$

$$= \gamma^{(j)}_{0,[0;t-1]} \cdot k + \sum_{q=1}^{t-1} \gamma^{(j)}_{q,[0;t-1]} \Gamma_{a,\mathcal{I},q} \tag{3c}$$

$$= Q_b(j) \tag{3d}$$

Where the equality between Eq. (3b) and Eq. (3c) holds by the assumed correctness of the initial shares $[k]_{a,i}$. Moreover, because the polynomial is of degree $t-1$, k can be interpolated from any t $[k]_{b,j}$. $Q_b(x)$ is thus a valid polynomial for (t,n)-threshold secret sharing of k.

B.2 Proof of Theorem 2

Let us write $\mathcal{I}$ and $\mathcal{J}$ for the sets of indices on layers L_a and L_b respectively, such that $\{P_{a,i}\}_{\forall i \in \mathcal{I}}$ and $\{P_{b,j}\}_{\forall j \in \mathcal{J}}$ are corrupted. That is, $\mathcal{A}$'s view consists of each corrupted party's random tape, and of all sub shares $[[k]_{a,i}]_j$, $\forall i \in \mathcal{I}$ and $\forall j$. Additionally, $\forall j \in \mathcal{J}$, its view contains $[[k]_{b,j}]_{\bar{i}}$, $\forall \bar{i} \notin \mathcal{I}$.

First of all, we prove that **(i)** every evaluation of any valid source polynomial at any layer L_a, $Q_a(x)$, is random given eve's view. It then trivially follows that

Eve cannot interpolate k from polynomial on two different levels. Then, we show that **(ii)** eve's knowledge is not sufficient to distinguish any $[k]_{a,\bar{i}}$ ($\forall \bar{i} \notin \mathcal{I}$) neither **(iii)** $[k]_{b,\bar{j}}$ ($\bar{j} \notin \mathcal{J}$) from random. In particular, we show it is directly implied by our initial requirement (i.e., by randomness of the hiding polynomial $Q_a(x)$).

(i) Assume **(ii)** and **(iii)**. The randomness of $Q_a(x)$ holds by **(ii)** and by our initial assumption (i.e., $Q_a(x)$ is a correct generator of Shamir's secret shares). On L_b, $\mathcal{A}$ has the most information on polynomials computed from $\{[[k]_{a,i}]_j\}_{i\in\mathcal{I}}$. Write, $\overline{\mathcal{J}} = \mathcal{J} \cup \bar{j}$, where $\bar{j} \notin \mathcal{J}$. The Lagrange interpolation yields,

$$Q_b(x) = \sum_{j\in\overline{\mathcal{J}}}^{t-1} \gamma_{j,\overline{\mathcal{J}}}^{(x)} Q_b(j),$$

with $Q_b(j) = [k]_{b,j}$. $Q_b(x)$'s randomness with respect to the adversarial view thus follows from **(iii)**.

(ii) For any $\bar{i} \notin \mathcal{I}$, $\mathcal{A}$ knows its $t-1$ sub shares $[[k]_{a,i}]_{\bar{j}}$, $\forall j \in \mathcal{J}$. By our initial assumption, $Q_a(x)$'s parameters were generated randomly, and $[k]_{a,\bar{i}}$ thus appears random from the sole knowledge of $\{[k]_{a,i}\}_{i\in\mathcal{I}}$.

Let us write $\overline{\mathcal{J}} = \mathcal{J} \cup \bar{j}$. We have

$$[k]_{a,\bar{i}} = \sum_{j\in\overline{\mathcal{J}}}^{t-1} \gamma_{j,\overline{\mathcal{J}}}^{(0)} [[k]_{a,i}]_{\bar{j}},$$

with $[[k]_{a,\bar{i}}]_{\bar{j}} = R_{a,\bar{i}}(\bar{j})$. As the adversarial view does not contain $P_{a,\bar{i}}$'s random tape, this value appears fully random to $\mathcal{A}$, and so does $[k]_{a,\bar{i}}$.

(iii) It can be shown similarly as per **(ii)**.

A simple recursive arguments lets us find back Definition 2.

The randomness of $Q_1(x)$, implied by (i), can be restated as

$$H(k \mid \{Q_1(i)\}_{\forall i\in\mathcal{I}}) = H(k).$$

which correctly initializes the recursive argument with the assumption

$$H(k \mid \{Q_1(i)\}_{\forall i\in\mathcal{I}_1}, \ldots, \{Q_n(i)\}_{\forall i\in\mathcal{I}_n}) = H(k).$$

Recursively, the randomness of $Q_b(x)$ is once again ensured by (i). Together with (ii) and (iii), it ensures that

$$H(k \mid \{Q_{b-1}(i)\}_{\forall i\in\mathcal{I}}, \{Q_b(j)\}_{\forall j\in\mathcal{J}}) = H(k \mid \{Q_{b-1}(i)\}_{\forall i\in\mathcal{I}}).$$

It then follows the expected result from Definition 2,

$$H(k \mid \{Q_1(i)\}_{\forall i\in\mathcal{I}_1}, \ldots, \{Q_{b-1}(i)\}_{\forall i\in\mathcal{I}_{b-1}}, \{Q_b(j)\}_{\forall j\in\mathcal{J}}) = H(k).$$

B.3 Proof of Theorem 3

We proceed with game-hops, based on our $\mathsf{Exp}^{\mathsf{lh-key-secrecy}}_{\Pi[\mathcal{G},\mathbf{SKD},\mathbf{Ch},\mathbf{SSS}],\mathcal{A}}(\lambda)$ from Fig. 5. Except for G_0, we only state the change with respect to the previous game.

Game G_0: In this game, each **SKD** is instantiated with ϵ-secure **QKD** protocol, **PSS** with a standard (t, n)-threshold Shamir's secret sharing and the **Reshare** and **Recombine** protocols from Fig. 6 and Fig. 7.

Game G_1: In this game, **SKD** is idealized. The keys shared between the parties are perfectly private, uniformly random keys, rather than keys from **SKD**.

Game G_2: In this game, the encryption scheme is idealised. The ciphertexts sent to uncorrupted parties are replaced by uniformly random bit strings.

$G_0 \to G_1$: Remark, if $\mathcal{A}$ compromised a **SKD** key, it may have chosen its corruption accordingly. Compromising any **SKD** key would thus mark the adversary's win. It could indeed choose its corruptions such that the compromised key belongs to two uncorrupted parties, and learn the key this way. The protocol involve $2n + (l-1)n^2$ **SKD** key, each perfectly secure with probability at most $1-\epsilon$, and fully compromised otherwise (cf. Definition 5). It thus follows,

$$\mathbf{Adv}^{G_0}_{\Pi,\mathcal{A}}(\lambda) \leq 1 - (1 - \mathbf{Adv}^{\mathrm{R/I}}_{\mathbf{SKD},\mathcal{A}}(\lambda))^{2n+(l-1)n^2} + \mathbf{Adv}^{G_1}_{\Pi,\mathcal{A}}(\lambda).$$

$G_1 \to G_2$: From the well-known security of the one-time pad, no adversary has any advantage in the distinguishing game between a real and ideal information-theoretic cipher **ITE**. As such, we can write

$$\mathbf{Adv}^{G_1}_{\Pi,\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{G_2}_{\Pi,\mathcal{A}}(\lambda).$$

Security of G_2. It follows directly from Theorem 2 that an adversary would have no advantage on G_2 given $t-1$ corruption per layer. As such, and because **PSS** is the only non-ideal protocol remaining,

$$\mathbf{Adv}^{G_2}_{\Pi,\mathcal{A}}(\lambda) = 0.$$

Considering an ϵ-secure key distribution protocol **SKD**, the adversary thus has the following advantage in winning the full game,

$$\mathbf{Adv}^{\text{lh-key-secrecy}}_{\Pi[\mathcal{G},\mathbf{QKD},\mathbf{OTP},\mathbf{SPSS}],\mathcal{A}}(\lambda) \leq 1 - (1-\epsilon)^{2n+(l-1)n^2}$$

B.4 Proof of Theorem 4

Intuitively, we show that, each time we start the **Reshare** protocol, at least $2t-1$ parties holding valid shares are available. That is, $2t-1$ parties in L_b need to recombine a valid sharing of our secret.

Assume $m = 0$. We start with $2t-1$ available parties in L_a, of which $t-1$ fail to proceed with **Reshare**. Each $P_{b,j} \in L_b$ receive t subshares which, by Theorem 1, is sufficient to correctly reshare to all receiving parties. It follows that $|L_j| = 2t-1$ is sufficient

Assume $m > 0$. Of the $t-1$ failing parties, as much of them as possible are in $L_a \cup L_b$. Then, $\min(m, t-1)$ parties in L_b are unavailable as well. Following the previous argument, it directly follows we require $|L_j| = 2t - 1 + \min(m, t-1)$

C Discussion

C.1 Communication Complexity and Latency

Crucially, QKD keys are (and will likely remain) expensive to generate. As such, while using multiple QKD keys to exchange one ITS symmetric key cannot be avoided, this overhead needs to be kept to a minimum. This necessity is further reinforced by the security cost of each key exchange (cf. Theorem 3).

Currently, considering l layers of equal size n, the protocol requires $(l-1) \cdot n^2 + 2n$ QKD keys. It thus requires n symmetric keys for the main client at both extremities, and $2n$ keys for each party in a layer. The $\mathcal{O}(n^2)$ cost per layer transition is inherent to the resharing step, in which every party on the sending layer must communicate with every party on the receiving layer. In practice, n is constrained by the number of nodes within QKD range of a given geographic location, which limits layers to moderately sized groups.

Since the protocol requires all pairwise QKD keys to be established before the key-transport phase begins (cf. Section 3.3), the pre-computation latency is dominated by QKD key generation. Adjacent pairs can run their QKD sessions in parallel, so the wall-clock time scales with the slowest link rather than the total number of keys. Once the key material is available, the transport phase itself executes at the speed of classical communication.

C.2 Resilience

Although the honest-but-curious model does not account for node failures, hardware faults remain a practical concern. The protocol requires at least t working nodes per layer to maintain correctness (cf. Theorem 1).

Theorem 4 (Resilience). *Assume a Party can only fail at the start of both protocols,* **Reshare** *or* **Recombine**. *Consider two layers , L_a, L_b, such that $|L_a \cup L_b| = m$, $m \geq 0$. Then, our protocol is resilient to $t-1$ failures per layer, considering committees of at least $n = 2t - 1 + \min(m, t-1)$ parties.*

Proof. Refer to Appendix B.4. □

C.3 Verifiability

A natural extension is to handle active adversaries that may deviate from the prescribed computation, for instance by sending malformed shares or selectively dropping messages. In principle, replacing Shamir's secret sharing with a verifiable secret sharing (VSS) scheme such as [DS02] would allow honest parties to detect and discard invalid contributions. However, the additional verification

rounds would substantially increase QKD key consumption, which is already the protocol's main practical cost. The challenge is therefore to integrate verifiability while keeping the number of QKD keys within deployable bounds. verifiability becomes relevant only when moving to the active setting.

C.4 Optimising the Communications

QKD key being expensive to compute, lowering the amount of communication needed would also be a nice extension to the protocol. It is sometimes proposed to amortise the costs of Shamir's secret sharing by batching multiple secrets in a same polynomial. While this technique is a well studied and still actively improved upon [AVY24,ABN+25], it remains challenging in the proactive setting. [BELO15], for instance, requires fairly large committees that still need the assistance of trusted nodes.

Alternatively, finding an approach such as [FNR23] suitable for Shamir's secret sharing could be an interesting direction, granted that it works for small committees.

C.5 Optimizing the Network Infrastructure

While the communication complexity can give a good indication of the protocol's viability in practice, it is not sufficient here. Indeed, if QKD's costs tend to increase linearly at short distances, it becomes exponential when we approach its maximal range. The network infrastructure would thus need to be balanced. Although balancing layer spacing and key rate would be fairly trivial considering a single node per layer, this might prove more challenging in our use-case.