Processor-Aware Quantum Time-Memory-Data Trade-Offs

Aniket Ghara[1,2], Suman Pal[1,2], Prokash Kumar Kundu[1,2], and Arpita Maitra[1,2]

[1] Institute for Advancing Intelligence (IAI), TCG CREST, Kolkata, India
[2] Academy of Scientific and Innovative Research (AcSIR), Ghaziabad, India
aniket.ghara.179@tcgcrest.org, suman.pal.181@tcgcrest.org, prokash.kumar.kundu.180@tcgcrest.org, arpita.maitra@tcgcrest.org

Abstract. Time–memory–data trade-offs (TMDTOs) provide a fundamental framework for cryptanalytic attacks that balance preprocessing,memory, online computation, and available target data. In the quantum setting, Dunkelman et al. introduced quantum TMDTO attacks based on GroverMatch, obtaining the relation $N^2 = T^{4/3}M^2D^2$ for multiple-target inversion using a single quantum processor. In this work, we introduce a processor-aware formulation of quantum TMDTOs by incorporating parallel quantum computation into the GroverMatch framework. We partition the flavour space across K parallel quantum processors and use quantum counting to identify a promising local subspace, followed by Grover search within that subspace. Under an idealized parallel wall-clock model, we show that the online complexity becomes $T_K = \frac{t^{3/2}}{\sqrt{K}}$, and consequently derive the processor-aware trade-off $N^2 = T_K^{4/3}M^2D^2K^{2/3}$. The relation specializes to the previously known single-processor quantum TMDTO relation when $K = 1$. We instantiate the framework for both Quantum Hellman and Quantum Rainbow constructions and show that, for fixed N, M, and D, parallelization reduces the online wall-clock time by a factor $\sqrt{K}$, at the cost of increasing the aggregate quantum work by a factor of $\sqrt{K}$. We further analyze the resulting trade-off against generic Grover search under both single-processor and equal-processor-budget comparisons. Our results provide a processor-aware characterization of quantum TMDTO attacks and clarify the trade-off between quantum hardware parallelism and online cryptanalytic time.

1 Introduction

Time–memory trade-off (TMTO) attacks are fundamental cryptanalytic techniques for inverting one-way functions by balancing preprocessing memory against online computation. Originally proposed by Hellman [1], the approach uses an offline preprocessing phase to construct auxiliary tables that enable faster online inversion than exhaustive search. Subsequent works [2–5] extended this framework to the time–memory–data trade-off (TMDTO), where the adversary is

given multiple target outputs and succeeds by inverting at least one of them. For a random function with domain size $N = 2^n$, the classical TMTO relation is $TM^2 = N^2$ under preprocessing cost $O(N)$.

Quantum computing substantially changes the complexity landscape of cryptanalysis. Grover's algorithm [6] reduces exhaustive search from $O(N)$ to $O(\sqrt{N})$, while quantum counting [7] provides an efficient means of estimating the number of marked elements in a search space. These primitives form the basis of several quantum cryptanalytic techniques. In particular, Nayebi et al. [8] established the lower bound $MT^2 = \tilde{\Omega}(N)$ for inverting a random permutation with classical advice. Related bounds for random functions are obtained by Hham et al. [9] and Chung et al. [10].

More recently, Dunkelman *et al.* [11] introduced quantum TMDTO attacks based on the *GroverMatch* algorithm and developed quantum variants of the classical Hellman and Rainbow constructions. For multiple-target inversion, their single-processor construction achieves $N^2 = T^{4/3}M^2D^2$, where M, T, and D denote memory, online time, and the number of available target outputs, respectively. These constructions retain the preprocessing structure of the corresponding classical TMDTO attacks but perform the online search using a single quantum processor.

The single-processor setting naturally raises the question of whether the online search phase can be distributed across multiple quantum processors. The structured search space arising in GroverMatch can be partitioned among processors, making distributed quantum search a natural approach for reducing online wall-clock time. Here, wall-clock time refers to the elapsed time from the start to the completion of the parallel computation; it is distinct from the aggregate computational work performed by all processors. We focus on this idealized parallel wall-clock-time model and do not account for communication, synchronization, initialization, or other hardware-level costs.

A relevant primitive for this purpose is the distributed Grover search technique of Qiu *et al.* [12], which partitions the search space among multiple quantum processors and combines quantum counting with Grover search to identify and search promising subspaces. Although this approach does not reduce the total query complexity of unstructured search, it provides a $\sqrt{K}$ reduction in wall-clock time with K parallel processors. Thus, distributed Grover search provides a natural basis for parallelizing the online phase of quantum TMDTO attacks.

Distributed search is also relevant to current NISQ-era devices, where large Grover circuits are constrained by gate infidelity, decoherence, and accumulated errors [13]. By reducing the search depth executed by each processor, distributed Grover can make the computation more amenable to shallow-circuit implementations. Moreover, unlike several distributed quantum algorithms requiring interprocessor quantum communication [13–15], distributed Grover does not require quantum teleportation or high-fidelity Bell pairs. Its practical feasibility has also been demonstrated through an experimental distributed Grover attack on human-generated passwords [16].

Motivated by these observations, this work studies the distributed implementation of the GroverMatch-based quantum TMDTO framework. Our objective is not to reduce the total quantum work of GroverMatch, but to characterize its online wall-clock complexity when the flavour space is distributed across K quantum processors. This leads to a processor-aware quantum TMDTO relation that explicitly captures the trade-off between online wall-clock time, memory, data, and the number of parallel quantum processors.

Our Contributions. Our contributions are as follows.

- We introduce a processor-aware formulation of quantum TMDTOs in which the number of parallel quantum processors K is treated as an explicit attack resource and derive the processor-aware relation $N^2 = O(T_K^{4/3} M^2 D^2 K^{2/3})$, under an idealized parallel wall-clock model.
- We instantiate the framework for both Quantum Hellman and Quantum Rainbow attacks by distributing their GroverMatch search spaces across parallel quantum processors.
- We characterize the resulting trade-offs and advantage window against generic Grover search, distinguishing the standard single-processor comparison from the equal-processor budget setting.

Main Result. Our main technical result is the following processor-aware time–memory–data trade-off.

Theorem 1 (Processor-Aware Quantum TMDTO). *Consider a quantum TMDTO construction with search-space parameter t, memory M, target-data parameter D, and K parallel quantum processors. Suppose that the corresponding GroverMatch search space of size t can be partitioned into K disjoint subspaces of size t/K, while the evaluation of the underlying match predicate requires $O(t)$ function evaluations and the selected local subspace contains $O(1)$ marked elements. Under the parallel wall-clock-time model, the online complexity is $T_K = O\left(\frac{t^{3/2}}{\sqrt{K}}\right)$. Using the preprocessing relation $t = \frac{N}{MD}$, we obtain the processor-aware quantum TMDTO relation $N^2 = O\left(T_K^{4/3} M^2 D^2 K^{2/3}\right)$ that introduces quantum hardware parallelism as an explicit fourth resource alongside T, M, D. For $K = 1$, this reduces to the single-processor quantum TMDTO relation $N^2 = O\left(T_1^{4/3} M^2 D^2\right)$.*

The central contribution of this work is therefore not a new quantum-search primitive, but a processor-aware extension of the quantum TMDTO framework in which quantum parallelism becomes an explicit attack resource.

The theorem is established for the Quantum Hellman and Quantum Rainbow constructions in section 5 and 6, respectively. In both cases, the distributed construction preserves the offline preprocessing phase and reduces the online wall-clock time by a factor of $\sqrt{K}$. The corresponding reduction is in parallel wall-clock time and does not imply a reduction in total quantum work. Table 1

Algorithm	Time–Memory–Data Relation
Classical Hellman	$N^2 = TM^2D^2$ [1] [4]
Classical Rainbow	$N^2 = TM^2D^2$ [17] [18]
Quantum Hellman	$N^2 = T^{4/3}M^2D^2$ [11]
Quantum Rainbow	$N^2 = T^{4/3}M^2D^2$ [11]
Distributed Quantum Hellman with K parallel processors	$N^2 = T^{4/3}M^2D^2K^{2/3}$
Distributed Quantum Rainbow with K parallel processors	$N^2 = T^{4/3}M^2D^2K^{2/3}$

Table 1: Summary of the time–memory–data trade-off relations. For the distributed constructions, T denotes parallel wall-clock time and K denotes the number of parallel quantum processors. Constant factors are omitted throughout the table.

summarizes the principal classical, quantum, and distributed quantum TMDTO relations considered in this work.

The remainder of this paper is organized as follows. Section 2 introduces the notation and quantum search primitives required in the subsequent analysis, including distributed Grover search. Section 3 reviews the Quantum Hellman attack of Dunkelman *et al.* [11]. Section 4 describes the corresponding Quantum Rainbow construction. Sections 5 and 6 develop the proposed distributed variants and analyze their online complexity as a function of the number of parallel quantum processors. Section 7 compares the proposed constructions with standard Grover search. Finally, Section 8 concludes the paper and discusses directions for future work.

2 Preliminaries

Throughout this paper, we denote the search space by $N = 2^n$. This section briefly reviews the classical Time–Memory–Data Trade-off (TMDTO) framework and the distributed Grover search algorithm that forms the basis of the proposed distributed quantum TMDTO attack.

2.1 Classical Time–Memory–Data Trade-off

Time–Memory Trade-off (TMTO) attacks reduce the online computational effort by performing an offline preprocessing phase and storing auxiliary information. Hellman [1] introduced the first practical TMTO construction, establishing the trade-off $TM^2 = N^2$, where T and M denote the online time and memory complexities, respectively.

Oechslin [17] later proposed the Rainbow table technique, which employs distinct reduction functions to reduce chain collisions and improve the practical efficiency of Hellman's construction. The corresponding trade-off is $N^2 = 2TM^2$.

In the multiple-data setting, the adversary is given D target outputs and succeeds by inverting at least one of them. We use D to denote the number of available target outputs throughout the paper. Following the parameterization adopted by Dunkelman *et al.* [11], the preprocessing parameters satisfy $\frac{N}{D} = mt^2$, where m denotes the number of starting points per table and t denotes the TMDTO parameter. Under this parameterization, the construction uses approximately t/D tables. Each chain has length related to $O(t)$ in the construction. Hence, the corresponding memory requirement is $M = \frac{mt}{D}$. These relations provide the parameterization used in the subsequent quantum TMDTO complexity analysis.

2.2 Grover Search

Grover's algorithm [6] provides a quadratic quantum speedup for searching an unstructured space. Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function, and suppose that $a = |\{x \in \{0,1\}^n : f(x) = 1\}| \geq 1$. The objective is to find an input x satisfying $f(x) = 1$. For a search space of size $N = 2^n$, Grover search requires $O\left(\sqrt{\frac{N}{a}}\right)$ oracle queries, whereas classical exhaustive search requires $O(N)$ queries in the worst case. When the number a of marked elements is known, the number of Grover iterations can be chosen as $\left\lfloor \frac{\pi}{4}\sqrt{\frac{N}{a}} \right\rfloor$, which yields a success probability close to one.

In the constructions considered in this paper, Grover search is applied to structured subspaces identified by a preceding quantum-counting procedure. Consequently, if a subspace of size S contains a marked elements, the subsequent Grover search requires $O\left(\sqrt{\frac{S}{a}}\right)$ oracle queries. This complexity is the basic quantum-search component used in the distributed GroverMatch constructions developed in Sections 5 and 6.

In the subsequent constructions, the oracle is not assumed to have unit implementation cost; the cost of evaluating the underlying Hellman or Rainbow transformation is accounted for separately.

2.3 Quantum Counting

Quantum counting [7] provides a quantum procedure for estimating the number of marked elements in a search space. Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function and let $a = |\{x \in \{0,1\}^n : f(x) = 1\}|$. Quantum counting estimates a by applying phase estimation to the Grover iterate associated with f. The procedure uses a quantum Fourier transform and controlled applications of the Grover iterate.

For a search space of size $S = 2^n$, the constant-precision counting procedure used in the distributed-search framework considered here requires $O(\sqrt{S})$ queries to the underlying Boolean function to obtain an estimate of the number of marked elements with constant success probability and an additive estimation error of order $O(\sqrt{a})$.

In the distributed GroverMatch construction considered in this paper, quantum counting is applied independently to the subspaces assigned to the parallel quantum processors. Its purpose is not to determine the exact number of solutions, but rather to identify subspaces that contain one or more candidate matches. Once such a subspace has been identified, Grover search is applied within that subspace. Here, we assume that the distributed quantum-counting and selection procedure identifies, with constant overall success probability, a processor whose local search space contains a marked element.

Thus, for a subspace of size S containing a marked elements, the relevant quantum-counting cost is $O(\sqrt{S})$, while the subsequent Grover search requires $O\left(\sqrt{\frac{S}{a}}\right)$ queries. These bounds are used in the complexity analysis of the distributed GroverMatch constructions in Sections 5 and 6.

2.4 Distributed Grover Search

Distributed Grover search [12] extends Grover's search algorithm to a collection of parallel quantum processors. Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function, and let $a = |\{x \in \{0,1\}^n : f(x) = 1\}| \geq 1$.

For an integer k, with $0 \leq k \leq n$, the search space can be partitioned into $K = 2^k$ disjoint subspaces according to the last k bits of the input. More precisely, for $i \in \{0, \ldots, K-1\}$, let $y_i \in \{0,1\}^k$ denote the binary representation of i and define $f_i(x) = f(x\|y_i), \qquad x \in \{0,1\}^{n-k}$, where $\|$ denotes concatenation. Let $a_i = \left|\left\{x \in \{0,1\}^{n-k} : f_i(x) = 1\right\}\right|$ denote the number of marked elements in the i-th subspace. Then $\sum_{i=0}^{K-1} a_i = a$.

The K subspaces can be assigned independently to K quantum processors. Each processor first applies quantum counting [7] to its local search space to determine whether the corresponding subspace contains marked elements. Processors corresponding to empty subspaces can then be discarded, while Grover search is applied to a promising nonempty subspace. If the i-th subspace contains $a_i \geq 1$ marked elements, its Grover search requires $O\left(\sqrt{\frac{2^{n-k}}{a_i}}\right)$ queries. Since $2^{n-k} = \frac{N}{K}$, the corresponding parallel search complexity is $O\left(\sqrt{\frac{N}{Ka_i}}\right)$.

The quantum-counting stage requires $O(\sqrt{N/K})$ queries per processor. Consequently, under the parallel wall-clock-time model, the search space partitioning provides a $\sqrt{K}$ reduction in the search complexity relative to a single-processor search of the same space, subject to the cost of the subsequent Grover search and the availability of K parallel quantum processors. In particular, if the relevant subspace contains a constant number of marked elements, the online search complexity scales as $T_K = O\left(\sqrt{\frac{N}{K}}\right) = \frac{1}{\sqrt{K}}O(\sqrt{N})$. The $\sqrt{K}$ improvement refers to parallel wall-clock time. The total number of oracle queries across all processors is not reduced by the same factor.

In the proposed quantum TMDTO constructions, the same principle is applied to the flavour space rather than to the complete input space. The flavours are partitioned among K parallel quantum processors, quantum counting is used

to identify processors whose local flavour subspaces contain candidate matches, and Grover search is subsequently performed within the selected subspaces. The resulting K-dependent wall-clock complexity is analyzed explicitly in Sections 5 and 6.

2.5 Distributed GroverMatch Algorithm

We combine the *GroverMatch* framework of Dunkelman *et al.* [11] with the distributed Grover search paradigm of Qiu *et al.* [12] to construct a distributed variant of GroverMatch. The main objective of this work is not to reduce the total quantum query complexity of GroverMatch, but to characterize how its online wall-clock complexity changes when the flavour space is distributed across multiple quantum processors. This leads to a processor-aware quantum TMDTO relation that explicitly captures the trade-off between online time and the number of parallel quantum processors.

The GroverMatch procedure searches for a match between values generated by a function $\mathcal{G}$ and entries stored in a quantum-accessible classical memory (QACM). Let H denote the corresponding match predicate, defined by

$$H(x) = \begin{cases} 1, & \text{if the value generated from } x \text{ matches an entry in the QACM table,} \\ 0, & \text{otherwise.} \end{cases}$$

If the search space contains S candidate inputs and the evaluation of $\mathcal{G}$ requires time $\mathcal{T}_{\mathcal{G}}$, GroverMatch requires $O\left(\mathcal{T}_{\mathcal{G}}\sqrt{S}\right)$ time in the single-processor setting, ignoring constant factors and the cost of preprocessing.

In the multiple-data-point setting considered in this paper, there are D target outputs and approximately t/D flavours. Let $n' = \frac{t}{D}$. We distribute these flavours among $K = 2^k$ parallel quantum processors. The flavour indices are partitioned according to their first k bits, so that each processor receives approximately $\frac{n'}{K} = \frac{t}{KD}$ flavours. Unlike the partitioning convention adopted in Qiu *et al.* [12], where the processor assignment is defined using the last k bits, we use the first k bits of the flavour index to determine the processor. This difference is only a choice of indexing convention and does not affect the partition size, search-space decomposition, or the resulting asymptotic complexity. Since each processor considers all D target outputs, the effective search space assigned to each processor has size $D \cdot \frac{t}{KD} = \frac{t}{K}$.

For each processor, the corresponding flavours and data points are evaluated coherently using the Quantum Hellman or Quantum Rainbow procedure. The resulting superposition is mapped to the associated distinguished points, which are subsequently compared with the entries stored in the QACM table(s) associated with the assigned flavour subspaces. Quantum counting is first used to estimate whether the local subspace contains one or more candidate matches. A processor associated with a promising local subspace is then selected, and Grover search is performed within that subspace to identify a matching flavour and distinguished point.

The distributed procedure is summarized in Algorithm 1. The algorithm is applied to both the Quantum Hellman and Quantum Rainbow constructions; the only difference lies in the procedure $\mathcal{G}$ used to map a flavour and data point to the corresponding distinguished point.

Algorithm 1: Distributed GroverMatch

Input: An algorithm $\mathcal{G}$ corresponding to the Quantum Hellman or Quantum Rainbow construction; $K = 2^k$ parallel quantum processors; D target outputs; QACM table(s) associated with the assigned flavour subspaces.

Output: A matching pair consisting of a data point and its corresponding flavour, together with the associated distinguished point.

1 Set $n' = t/D$ and partition the flavour indices into $K = 2^k$ disjoint subsets.
2 Assign one flavour subset to each quantum processor.
3 For each processor $j \in \{0, \ldots, K-1\}$, prepare the superposition over its assigned flavours and the D target outputs, and apply $\mathcal{G}$ to obtain the corresponding distinguished points.
4 Apply quantum counting to estimate the number of marked pairs in the local search space.
5 Select a processor j^* with the highest estimated marked-element count, which indicates a nonempty/promising local search space.
6 Apply Grover search on processor j^*.
7 Measure the resulting state and verify the candidate against the corresponding QACM table.
8 **return** *the matching data point, flavour, and distinguished point if a valid match is found.*

The parallelization does not alter the offline preprocessing phase. Each processor uses the QACM information required for evaluating the match predicate on its assigned flavour subspace. Thus, the distributed construction changes only the organization of the online search. In particular, the search space of size t in the corresponding single-processor GroverMatch procedure is partitioned into K subspaces of size approximately t/K.

Consequently, the Grover search performed by an individual processor acts on a space of size t/K. The corresponding parallel wall-clock complexity is $T_K = O\left(t\sqrt{\frac{t}{K}}\right) = O\left(\frac{t^{3/2}}{\sqrt{K}}\right)$, where the factor t accounts for the cost of evaluating the corresponding Quantum Hellman or Quantum Rainbow transformation. For $K = 1$, this reduces to the single-processor GroverMatch complexity $T_1 = O(t^{3/2})$. Hence, under the parallel wall-clock-time model, the distributed construction provides a speedup of $\frac{T_1}{T_K} = \sqrt{K}$, when K quantum processors are employed.

2.6 Computational Model and Complexity Measures

We analyze the proposed distributed constructions under an idealized parallel quantum-computing model consisting of K quantum processors operating concurrently. The objective is to characterize the processor-dependent online *wall-clock complexity* of the proposed Distributed Quantum Hellman and Distributed Quantum Rainbow attacks.

The K processors are assumed to operate independently and in parallel, with the flavour space partitioned into K disjoint subsets. Each processor operates on its assigned subset of flavours together with the complete set of D target

data points. Consequently, the local GroverMatch search space has size $S_K = D \cdot \frac{t}{KD} = \frac{t}{K}$.

The cost of evaluating a local match predicate is determined by the corresponding Hellman or Rainbow traversal. As established in the preceding sections, evaluation of either traversal requires $O(t)$ function evaluations. Therefore, Grover search over the local search space requires $O\left(\sqrt{\frac{t}{K}}\right)$ applications of the corresponding match predicate, resulting in the time complexity $O\left(t\sqrt{\frac{t}{K}}\right)$.

The same cost order applies to the quantum-counting stage when the cost of evaluating the underlying match predicate is taken into account. Specifically, quantum counting over a local search space of size S_K requires $O\left(\sqrt{S_K}\right) = O\left(\sqrt{\frac{t}{K}}\right)$ evaluations of the underlying match predicate. Since each such evaluation requires $O(t)$ function evaluations, the corresponding time complexity is $O\left(t\sqrt{\frac{t}{K}}\right)$.

Because quantum counting is performed independently and concurrently on all K processors, its contribution to the parallel wall-clock time is the cost incurred by a single processor rather than the sum of the costs over all processors. Thus, up to constant factors, the online wall-clock complexity of the distributed construction is $T_K = O\left(t\sqrt{\frac{t}{K}}\right) = O\left(\frac{t^{3/2}}{\sqrt{K}}\right)$.

Throughout this work, T_K denotes the parallel wall-clock complexity, rather than the total amount of quantum computation performed across all processors. The corresponding aggregate computational work is therefore proportional to $W_K = KT_K = O\left(t^{3/2}\sqrt{K}\right)$.

Hence, increasing the number of quantum processors reduces the wall-clock time by a factor of $\sqrt{K}$, but does not reduce the total quantum work by the same factor. The proposed construction should therefore be interpreted as a trade-off between online wall-clock time and the number of parallel quantum processors.

For the asymptotic analysis, we assume that the processors have access to the QACM information required for evaluating their respective local match predicates. The physical implementation of QACM distribution, whether through partitioning, replication, or another architecture, is outside the scope of this work. Similarly, the costs associated with inter-processor communication, synchronization, processor initialization, and classical coordination are not included in the asymptotic wall-clock complexity. The analysis therefore characterizes the idealized parallel-search complexity of the proposed constructions rather than the end-to-end implementation cost of a specific distributed quantum architecture.

3 Hellman Attack

Hellman's time–memory trade-off attack [1] provides the basic table-based construction underlying the quantum TMDTO attack considered in this work. In this section, we first briefly describe the relevant classical construction and then review its quantum version based on *GroverMatch* [11]. The detailed classical

TMDTO framework has already been introduced in Section 2; here we focus on the parameters and procedures that are required for the subsequent distributed construction.

3.1 Hellman Construction

The basic idea of Hellman's construction is to perform an offline preprocessing phase in which chains of function evaluations are generated and only selected endpoints are stored. During the online phase, the target value is iteratively transformed and the resulting values are compared with the stored endpoints. If a match is found, the corresponding chain is recomputed to recover the preimage.

A standard refinement [19] uses *distinguished points*. A distinguished point is an element satisfying a publicly specified property, for example, a prescribed pattern in a fixed number of output bits. Instead of terminating every chain after a fixed number of iterations, the chain is continued until a distinguished point is encountered, subject to a maximum allowed number of iterations. The use of distinguished points facilitates the identification of candidate chains while keeping the amount of stored information small.

3.2 Offline Phase

In the multiple-data setting, the adversary is given D target outputs and aims to invert at least one of them. Following the parameterization used in the quantum TMDTO construction of Dunkelman *et al.* [11], let m denote the number of chains stored per table and let t denote the TMDTO parameter. The preprocessing construction uses approximately t/D tables. Each table contains m chains, and each chain requires $O(t)$ function evaluations.

Consequently, the corresponding preprocessing work is approximately $\frac{N}{D} = \frac{mt^2}{D}$, while the memory requirement is $M = \frac{mt}{D}$. Equivalently, $t = \frac{N}{MD}$. These relations will be used throughout the subsequent quantum and distributed complexity analyses.

For each flavour ℓ, let $f_\ell(x) = f(L_\ell(x))$, where L_ℓ denotes the corresponding invertible transformation. Starting from a randomly selected point, the construction iteratively evaluates the flavour-specific function until a distinguished point is reached or a prescribed maximum number of iterations is exceeded. Only the relevant starting and distinguished points are retained in the table; intermediate chain values are discarded.

3.3 Online Phase

Given a target value y, the classical Hellman attack applies the appropriate flavour-specific transformations and checks whether a distinguished point is reached. If a distinguished point matches an entry in the corresponding table, the associated starting point is recovered and the chain is recomputed to determine whether it contains a preimage of y.

For given D data point, the online complexity of the classical construction is $T = O(t^2)$. Together with $M = \frac{mt}{D}$ and $t = \frac{N}{MD}$, this gives the classical TMDTO relation $N^2 = TM^2D^2$.

3.4 Quantum Hellman Attack

The quantum Hellman attack of Dunkelman *et al.* [11] replaces the classical online search component by a quantum *GroverMatch* procedure over the target data points and flavours. The preprocessing phase remains classical, and the resulting tables are represented as quantum-accessible classical memory (QACM). Thus, the quantum attack does not require any modification of the offline preprocessing procedure.

Let $\mathcal{D}$ denote the set of target data points and let $\mathcal{F}$ denote the set of flavours. We have $|\mathcal{D}| = D$ and $|\mathcal{F}| = \frac{t}{D}$. An individual target data point is denoted by $d \in \mathcal{D}$, while an individual flavour is denoted by $\ell \in \mathcal{F}$.

For a pair $(d, \ell) \in \mathcal{D} \times \mathcal{F}$, the function F_{Hellman} iteratively applies the flavour-specific function f_ℓ until a distinguished point is encountered or a maximum number of iterations is reached. More precisely, F_{Hellman} is defined as follows.

Algorithm 2: The function F_{Hellman} used in the Quantum Hellman attack

Input: A data point $d \in \mathcal{D}$ and a flavour $\ell \in \mathcal{F}$.
Output: The distinguished point reached from d under flavour ℓ, or the final value after $8t$ iterations if no distinguished point is reached.

Set $d' \leftarrow d$.;
for $i = 1$ **to** $8t$ **do**
  $d_{\text{tmp}} \leftarrow f_\ell(d')$.;
  if d' *is a distinguished point* **then**
    $d' \leftarrow d'$.;
  end
  else
    $d' \leftarrow d_{\text{tmp}}$.;
  end
end
return d'.;

The function F_{Hellman} can be evaluated coherently on a superposition of target data points and flavours. For coherent evaluation, the loop is executed for the full $8t$ iterations; once a distinguished point is reached, the state is frozen and subsequent iterations do not alter the output.

In particular, the input state can be written as $\frac{1}{\sqrt{D}} \sum_{d \in \mathcal{D}} |d\rangle \otimes \frac{1}{\sqrt{t/D}} \sum_{\ell \in \mathcal{F}} |\ell\rangle$. The application of F_{Hellman} produces the corresponding distinguished points while preserving the data-point and flavour registers.

The *GroverMatch* procedure searches for a pair (d, ℓ) whose corresponding output under F_{Hellman} occurs in the QACM table. We therefore define the matching predicate

$$H_{\text{Hellman}}(d, \ell) = \begin{cases} 1, & \text{if } F_{\text{Hellman}}(d, \ell) \text{ matches an entry} \\ & \text{in the corresponding QACM table,} \\ 0, & \text{otherwise.} \end{cases}$$

Grover search is then applied to amplify the pairs satisfying $H_{\text{Hellman}}(d, \ell) = 1$.

Once a matching pair (d, ℓ) is obtained, the corresponding table entry is retrieved and the associated chain is recomputed classically to determine whether it contains a preimage of d. The online procedure is summarized in Algorithm 3.

Algorithm 3: Quantum Hellman Attack: Online Phase

Input: A superposition over data points $d \in \mathcal{D}$, a superposition over flavours $\ell \in \mathcal{F}$, and the corresponding QACM table $\mathbf{T}$.
Output: A pair (x, ℓ) such that x is a preimage of a target data point $d \in \mathcal{D}$.

Use *GroverMatch* to find a pair (d, ℓ) satisfying

$$H_{\text{Hellman}}(d, \ell) = 1.$$

Compute

$$z = F_{\text{Hellman}}(d, \ell).$$

Retrieve the corresponding table entry (start, z) from the QACM table $\mathbf{T}_\ell$.;
Set $x \leftarrow \text{start}$.;
for $i = 1$ **to** $8t$ **do**
  if $f_\ell(x) = d$ **then**
    return (x, ℓ).;
  end
  $x \leftarrow f_\ell(x)$.;
end

3.5 Complexity of the Quantum Hellman Attack

We now analyze the online complexity of the single-processor quantum Hellman attack. The *GroverMatch* procedure searches over the Cartesian product $\mathcal{D} \times \mathcal{F}$. Since $|\mathcal{D}| = D$ and $|\mathcal{F}| = \frac{t}{D}$, the total search space has size $|\mathcal{D} \times \mathcal{F}| = D \cdot \frac{t}{D} = t$.

For each pair (d, ℓ), evaluating F_{Hellman} requires at most $8t$ evaluations of f_ℓ. Hence, the cost of evaluating the GroverMatch predicate is $O(t)$. Grover search over a search space of size t requires $O(\sqrt{t})$ iterations. Therefore, the online wall-clock complexity with a single quantum processor is $T_1 = O\left(t\sqrt{t}\right) = O\left(t^{3/2}\right)$. Ignoring constant factors, we can write $T_1 = t^{3/2}$.

After a matching pair has been identified, the classical chain reconstruction requires at most $O(t)$ additional evaluations. Since $O(t) \subset O(t^{3/2})$ for the relevant asymptotic regime, this post-processing step does not change the overall online complexity.

From the preprocessing phase, we have $M = \frac{N}{tD}$,and hence the trade off relation becomes $N^2 = T_1^{4/3} M^2 D^2$,with constraint$T_1 \geq D^{3/2}$.

This single-processor construction serves as the baseline for the distributed quantum Hellman attack presented in Section 5.

4 Rainbow Attack

The Rainbow table construction, introduced by Oechslin [17], is a refinement of Hellman's time–memory trade-off technique. Unlike the Hellman construction, which uses multiple tables with different flavours, a Rainbow table uses a single table in which different reduction functions are employed at different positions

along a chain. This position-dependent structure reduces chain collisions and improves the practical coverage of the search space.

4.1 Offline Phase

In the offline phase, m random starting points $x_1, x_2, \ldots, x_m$ are selected. A chain of length t is generated from each starting point using a sequence of position-dependent functions $f_1, f_2, \ldots, f_t$. Thus, a chain starting from x_i has the form $x_i \to f_1(x_i) \to f_2(f_1(x_i)) \to \cdots \to y_i$, where $y_i = (f_t \circ f_{t-1} \circ \cdots \circ f_1)(x_i)$. Each f_j represents the reduction/function transformation used at the j-th position of the chain. Only the starting point x_i and the final point y_i are stored, while the intermediate chain values are discarded.

The resulting table therefore requires $M = m$ memory entries. Since each of the m chains contains approximately t function evaluations, the preprocessing complexity is $N \approx mt$. During the online phase, the target value is processed using the position-dependent functions in reverse order. The total number of function evaluations is approximately $T = \frac{t^2}{2}$.

Consequently, the classical Rainbow construction satisfies $N^2 = 2TM^2$. As usual, constant factors such as the factor 2 are ignored when considering asymptotic trade-offs, giving $N^2 = O(TM^2)$.

In the multiple-data quantum Rainbow construction, we denote by $S = \frac{t}{D}$ the number of position-dependent functions (or Rainbow positions) used in the quantum traversal. The parameter t remains the overall TMDTO parameter. For $D = 1$, $S = t$, and the construction reduces to the conventional single-target setting.

4.2 Online Phase

Given a target value d, the online Rainbow procedure searches for a chain whose endpoint corresponds to a transformation of d. Starting from the last position of the chain, the adversary first applies the corresponding function and checks whether the resulting value occurs as an endpoint in the table. If no match is found, the procedure considers the preceding position and applies the appropriate sequence of functions. More generally, the candidate endpoint values are obtained from compositions of the form $f_t \circ f_{t-1} \circ \cdots \circ f_j(d)$, where $j = t, t-1, \ldots, 1$. Whenever a candidate endpoint matches an entry in the table, the corresponding chain is recomputed from its starting point to determine whether it contains a preimage of d.

The number of function evaluations in this search is approximately $T = \frac{t(t+1)}{2} = O(t^2)$. Together with $M = m$ and $N \approx mt$, this gives the classical Rainbow time–memory relation $N^2 = O(TM^2)$.

For D number of data points the trade-off relation reduces to $N^2 = TM^2D^2$

4.3 Quantum Rainbow Table

The quantum Rainbow attack of Dunkelman *et al.* [11] uses the same *Grover-Match* framework as the quantum Hellman attack, but replaces the flavour-specific chain computation by the position-dependent function sequence of the Rainbow construction. The offline preprocessing phase remains classical, and the resulting start–end pairs are stored in quantum-accessible classical memory (QACM).

Let $\mathcal{D}$ denote the set of D target data points and let $|\mathcal{D}| = D$. Let $S = \frac{t}{D}$ denote the number of possible starting positions in the Rainbow traversal, and let $\mathcal{F} = \{1, 2, \ldots, S\}$ denote the corresponding set of starting positions. For a target data point $d \in \mathcal{D}$ and a starting position $\ell \in \mathcal{F}$, the function F_{Rainbow} performs the corresponding reverse Rainbow traversal and returns the distinguished point reached by that traversal, subject to the maximum number of iterations used in the preprocessing construction.

In a Rainbow table, the functions are position-dependent. For $\ell \in \mathcal{F}$, the reverse traversal begins with the function corresponding to position ℓ and subsequently applies the functions associated with the following positions. Thus, the initial part of the traversal is $d \longrightarrow f_\ell(d) \longrightarrow f_{\ell+1}(f_\ell(d)) \longrightarrow \cdots \longrightarrow f_S \circ \cdots \circ f_\ell(d)$. We regard the sequence of $S = t/D$ consecutive functions $f_1, \ldots, f_S$ as one complete Rainbow basic unit. After the initial position-dependent traversal, the construction repeatedly applies complete basic units. A distinguished-point test is performed only at the boundary of a complete basic unit, rather than after individual function evaluations. Once a distinguished point is reached, the state is frozen for all remaining basic units; otherwise, the next complete basic unit is applied. The traversal therefore uses at most $8t/S$ complete basic units. The corresponding procedure is represented by F_{Rainbow}.

Algorithm 4: The function F_{Rainbow} used in the Quantum Rainbow attack

Input: A target data point $d \in \mathcal{D}$ and a starting position $\ell \in \mathcal{F}$.
Output: The distinguished point reached from d, or the final value after the maximum number of iterations.

Set $d' \leftarrow d$.;
for $j = \ell$ **to** S **do**
  $d' \leftarrow f_j(d')$;
end
for $i = 1$ **to** $8t/S$ **do**
  if d' *is a distinguished point* **then**
    $d' \leftarrow d'$.;
  end
  else
    $d' \leftarrow f_S(f_{S-1}(\cdots f_1(d')\cdots))$.;
  end
end
return d'.;

The function F_{Rainbow} can be evaluated coherently on a superposition of target data points and starting positions. The corresponding input state is $\frac{1}{\sqrt{D}} \sum_{d \in \mathcal{D}} |d\rangle \otimes \frac{1}{\sqrt{S}} \sum_{\ell=1}^{S} |\ell\rangle$. Its evaluation consists of at most $S = t/D$ function evaluations in the initial position-dependent traversal, followed by at most $8t/S$ complete Rainbow basic units, each containing S function evaluations. Once a

distinguished point is reached, the corresponding state is frozen and no further function evaluations are performed on that branch. Hence, the total number of function evaluations is $O(t)$.

The *GroverMatch* procedure searches for a pair (d, ℓ) whose corresponding output under F_{Rainbow} occurs in the QACM table. We therefore define the matching predicate

$$H_{\text{Rainbow}}(d, \ell) = \begin{cases} 1, & \text{if } F_{\text{Rainbow}}(d, \ell) \text{ matches an entry in the QACM table,} \\ 0, & \text{otherwise.} \end{cases}$$

Grover search is then applied to amplify the pairs satisfying $H_{\text{Rainbow}}(d, \ell) = 1$. Once a matching pair (d, ℓ) is identified, the corresponding table entry is retrieved and the associated chain is recomputed classically to determine whether it contains a preimage of d. The online procedure is summarized in Algorithm 5.

Algorithm 5: Quantum Rainbow Attack: Online Phase

Input: A superposition over target data points $d \in \mathcal{D}$, a superposition over positions $\ell \in \mathcal{F}$, and the corresponding QACM table $\mathbf{T}$.
Output: A pair (x, ℓ) such that x is a preimage of a target data point $d \in \mathcal{D}$.

Use *GroverMatch* to find a pair (d, ℓ) satisfying

$$H_{\text{Rainbow}}(d, \ell) = 1.$$

Compute

$$z = F_{\text{Rainbow}}(d, \ell).$$

Retrieve the corresponding table entry (start, z) from the QACM table $\mathbf{T}$.;
Set $x \leftarrow \text{start}$.;
Recompute the corresponding Rainbow chain from x.;
if *the chain contains* d **then**
  return (x, ℓ).;
end

Complexity Analysis: The input to the algorithm is a superposition over D data points and t/D starting positions. Therefore, the effective search space for the *GroverMatch* procedure is $D \cdot S = D \cdot \frac{t}{D} = t$. The evaluation of F_{Rainbow} requires $O(t)$ function evaluations. Consequently, applying Grover search over a search space of size t gives the single-processor online wall-clock complexity $T_1 = O\left(t\sqrt{t}\right) = O\left(t^{3/2}\right)$. Ignoring constant factors, we write $T_1 = t^{3/2}$.

Using the preprocessing relation $M = \frac{N}{tD}$, we obtain $N^2 = T_1^{4/3} M^2 D^2$. Since $t \geq D$, the corresponding constraint is $T_1 \geq D^{3/2}$.

5 Distributed Grover on Hellman Attack

In this section, we incorporate the distributed Grover search paradigm of Qiu *et al.* [12] into the single-processor Quantum Hellman attack described in Section 3.4. Our objective is to reduce the online *wall-clock* time by distributing the search over multiple quantum processors.

The single-processor Quantum Hellman attack searches over the Cartesian product of the set of target data points and the set of relevant flavours. For D target data points and t/D flavours, the resulting search space has size $D \cdot \frac{t}{D} = t$.

As established in Section 3.4, evaluating the Hellman function F_{Hellman} requires $O(t)$ function evaluations. Consequently, Grover search over the search space of size t gives the single-processor online complexity $T_1 = O(t\sqrt{t}) = O(t^{3/2})$.

We now distribute the flavour space over K quantum processors. Let $K = 2^k$, where $0 \leq k \leq \log_2(t/D)$. The t/D flavour indices are represented using $\log_2(t/D)$ bits. In this work, the first k bits determine the processor, with the integer value represented by these bits identifying the corresponding processor, while the remaining $\log_2(t/D) - k$ bits specify the local flavour index within that processor. Thus, the flavour space is partitioned into K disjoint subsets, each containing $\frac{t}{KD}$ flavours.

This differs only in indexing convention from the partitioning convention adopted in Qiu *et al.* [12], where the processor assignment is defined using the last k bits. The choice of the first or last k bits does not affect the size of the partition or the resulting asymptotic complexity.

For simplicity, we assume that t/D and K are powers of two. Otherwise, the flavour space can be partitioned approximately evenly among the processors, without affecting the asymptotic complexity.

Each processor operates on the complete set of D target data points together with its assigned subset of flavours. Hence, the search space handled by each processor has size $D \cdot \frac{t}{KD} = \frac{t}{K}$.

The K processors perform the corresponding searches in parallel. Consequently, the wall-clock cost of the subsequent Grover search is $T_K = O\left(t\sqrt{\frac{t}{K}}\right) = O\left(\frac{t^{3/2}}{\sqrt{K}}\right)$. Ignoring constant factors, we write $T_K = \frac{t^{3/2}}{\sqrt{K}}$.

Thus, relative to the single-processor construction, the distributed search provides a wall-clock speedup of $\frac{T_1}{T_K} = \sqrt{K}$. This square-root dependence is the usual parallel-search speedup: the search space assigned to each processor is reduced by a factor K, while the Grover complexity depends on the square root of the search-space size.

It is important to distinguish the parallel wall-clock time from the total computational work. The latter is proportional to $KT_K = t^{3/2}\sqrt{K}$, and therefore increases with K. Accordingly, the distributed construction should be interpreted as a trade-off between online wall-clock time and the number of quantum processors, rather than as a reduction in the total amount of quantum computation.

5.1 Offline Preprocessing Phase

The offline preprocessing phase is unchanged from the single-processor Quantum Hellman construction. In particular, the same Hellman tables are constructed using the same set of flavours and the same chain parameters. The resulting tables are stored in quantum-accessible classical memory (QACM).

Let $\mathcal{F} = \{0, 1, \ldots, t/D - 1\}$ denote the set of flavours. We partition $\mathcal{F}$ into $K = 2^k$ disjoint subsets $\mathcal{F} = \mathcal{F}_0 \cup \mathcal{F}_1 \cup \cdots \cup \mathcal{F}_{K-1}$, where $|\mathcal{F}_j| = \frac{t}{KD}$. The first k bits of the binary representation of a flavour index determine the processor j to which the flavour is assigned and hence the corresponding portion of the QACM

table, while the remaining bits identify the local index of the flavour within that processor. Thus, each processor is assigned exactly $\frac{t}{KD}$ flavours.

Accordingly, the global QACM structure can be viewed as $\mathbf{T} = \mathbf{T}_0 \cup \mathbf{T}_1 \cup \cdots \cup \mathbf{T}_{K-1}$, where $\mathbf{T}_j$ contains the entries corresponding to the flavours in $\mathcal{F}_j$.

No additional preprocessing is required as a consequence of the parallelization. In particular, the preprocessing complexity and the overall memory requirement remain the same as in the single-processor Quantum Hellman construction.

5.2 Distributed Distinguished-Point Computation

For each processor j, we define a restricted Hellman function $F^{(j)}_{\text{Hellman}}$ whose flavour register ranges only over $\mathcal{F}_j$. The function is otherwise identical to F_{Hellman} defined in Algorithm 2.

The input consists of a target data point $d \in \mathcal{D}$ and a flavour $\ell \in \mathcal{F}_j$. Starting from d, the function iteratively applies the flavour-specific function f_ℓ until a distinguished point is encountered or the maximum number of iterations is reached.

Algorithm 6: $F^{(j)}_{\text{Hellman}}$ for the distributed Hellman attack

Input: A target data point $d \in \mathcal{D}$ and a flavour $\ell \in \mathcal{F}_j$.
Output: The distinguished point reached from d under flavour ℓ, or the final value after $8t$ iterations.

Set $d' \leftarrow d$.;
for $i = 1$ **to** $8t$ **do**
  $d_{\text{tmp}} \leftarrow f_\ell(d')$.;
  if d' *is not a distinguished point* **then**
    $d' \leftarrow d_{\text{tmp}}$.;
  end
end
return d'.;

5.3 Distributed GroverMatch: Online Phase

The K processors evaluate their respective restricted functions $F^{(j)}_{\text{Hellman}}$ in parallel. For processor j, the input search space is $\mathcal{D} \times \mathcal{F}_j$, whose size is $|\mathcal{D} \times \mathcal{F}_j| = D \cdot \frac{t}{KD} = \frac{t}{K}$.

A matching predicate for processor j is defined by

$$H_j(d, \ell) = \begin{cases} 1, & \text{if } F^{(j)}_{\text{Hellman}}(d, \ell) \text{ matches an entry in } \mathbf{T}_j, \\ 0, & \text{otherwise.} \end{cases}$$

Quantum counting is used to estimate which processor is associated with a local subspace containing one or more marked pairs. Once such a processor, denoted by j^*, is identified, Grover search is performed over its local search space. Since all processors perform the counting step in parallel, the relevant quantity for the distributed implementation is the corresponding parallel wall-clock complexity.

Once a matching pair (d, ℓ) is identified, the corresponding entry $(\text{start}, \text{end})$ is retrieved from the QACM table $\mathbf{T}_{j^*}$. The associated chain is then recomputed

classically, starting from start, until the target data point d is reached. The corresponding input is thereby recovered as a preimage of d. This classical post-processing is identical to that of the single-processor Quantum Hellman attack and does not alter the asymptotic online complexity derived below.

The algorithm is presented in Algorithm 7, along with the schematic diagrams shown in Figures 1 and 2.

Algorithm 7: Distributed Quantum Hellman Online Phase

Input: A set of D target data points $\mathcal{D}$; local QACM tables $\{\mathbf{T}_j\}_{j=0}^{K-1}$; the flavour space $\mathcal{F}$ of size t/D; and $K = 2^k$ parallel quantum processors.

Output: A pair (d, ℓ) corresponding to a valid match, where $d \in \mathcal{D}$ and $\ell \in \mathcal{F}$, together with the recovered preimage.

Partition $\mathcal{F}$ into $K = 2^k$ disjoint subsets $\mathcal{F}_0, \ldots, \mathcal{F}_{K-1}$, where $|\mathcal{F}_j| = t/(KD)$.;

foreach *processor* $j \in \{0, 1, \ldots, K-1\}$ *in parallel* **do**

  Prepare a superposition over $\mathcal{D} \times \mathcal{F}_j$.;

  Apply $F_{\text{Hellman}}^{(j)}$ to map each $(d, \ell) \in \mathcal{D} \times \mathcal{F}_j$ to its corresponding distinguished point.;

  Apply quantum counting to estimate the number of marked pairs in the local search space $\mathcal{D} \times \mathcal{F}_j$.;

end

Select a processor j^* with the highest quantum-counting estimate, which indicates a nonempty local search space.;

On processor j^*, apply Grover search using the local match predicate

$$H_{j^*}(d, \ell) = \begin{cases} 1, & \text{if } F_{\text{Hellman}}^{(j^*)}(d, \ell) \text{ matches an entry in } \mathbf{T}_{j^*}, \\ 0, & \text{otherwise,} \end{cases}$$

to obtain a marked pair (d, ℓ).;

Retrieve the corresponding table entry from $\mathbf{T}_{j^*}$ and perform the classical chain reconstruction to recover the preimage of d.;

return (d, ℓ) *and the recovered preimage.*;

5.4 Processor-Aware TMDTO Relation for Distributed Quantum Hellman

Proposition 1. *The Distributed Quantum Hellman construction achieves the processor-aware relation presented in Theorem 1.*

Proof. From the preprocessing phase, we have $M = \frac{N}{tD}$. With K parallel quantum processors, the flavour space is partitioned into K subsets, and each processor operates on a local search space of size t/K. The evaluation of the Quantum Hellman transformation requires $O(t)$ function evaluations. Therefore, the parallel wall-clock complexity of the Grover search is $T_K = O\left(t\sqrt{\frac{t}{K}}\right) = O\left(\frac{t^{3/2}}{\sqrt{K}}\right)$. Ignoring constant factors, we can write $T_K = \frac{t^{3/2}}{\sqrt{K}} = \frac{1}{\sqrt{K}}\left(\frac{N}{MD}\right)^{3/2}$. Rearranging we get,$N^2 = T_K^{4/3} M^2 D^2 K^{2/3}$.

We next consider the operating point used to obtain the specialized trade-off relation. The fundamental structural constraint arising from partitioning the flavour space is that the number of processors cannot exceed the number of available flavours. Since the total flavour space contains t/D elements, $K \leq \frac{t}{D}$. In addition, to obtain a representative balanced operating point, we choose the number of processors to be of the same asymptotic order as the number of

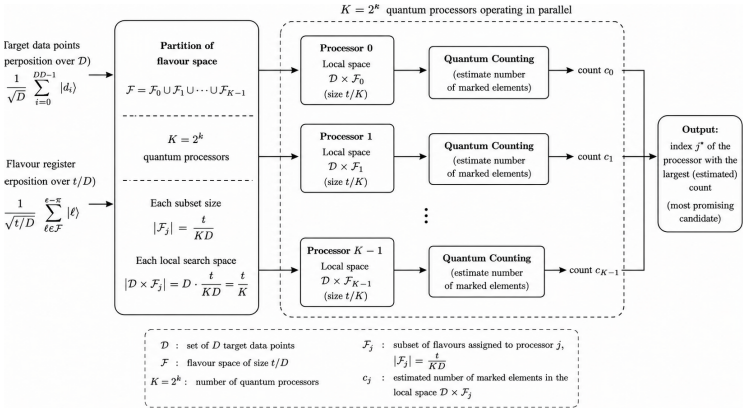

Fig. 1: Distributed quantum counting for the Quantum Hellman attack. The flavour space $\mathcal{F}$ is partitioned into $K = 2^k$ disjoint subsets $\mathcal{F}_0, \ldots, \mathcal{F}_{K-1}$, each containing $t/(KD)$ flavours. Each processor operates on the corresponding local space $\mathcal{D} \times \mathcal{F}_j$, of size t/K, and performs quantum counting in parallel. The counting stage identifies a processor j^* whose local space is likely to contain a marked pair.

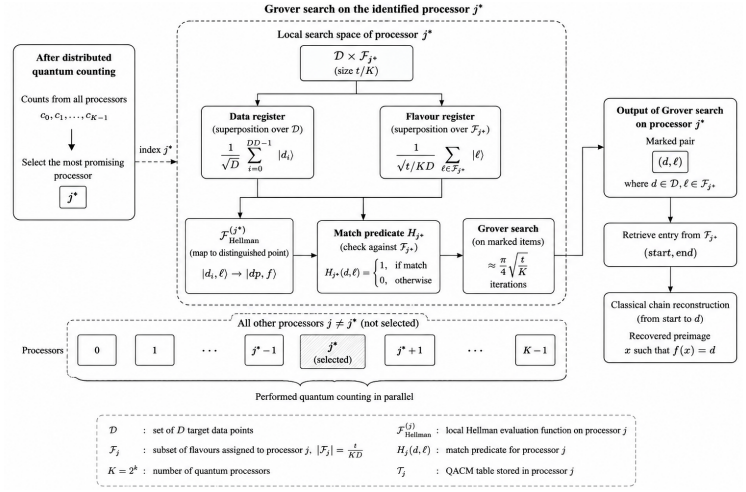

Fig. 2: Distributed Grover search on the identified processor j^* in the proposed Distributed Quantum Hellman attack. Following quantum counting across the $K = 2^k$ processors, the processor j^* with a promising local search space is selected. Grover search is then performed over $\mathcal{D} \times \mathcal{F}_{j^*}$, followed by QACM lookup and classical chain reconstruction to recover the corresponding preimage.

Grover iterations performed by each processor. Thus, we set $K = \Theta\left(\sqrt{\frac{t}{K}}\right)$, which gives $K = \Theta(t^{1/3})$. For simplicity, we consider the representative scaling $K = t^{1/3}$. This should be viewed as a choice of operating point rather than as a fundamental feasibility constraint.

This choice is feasible provided that $t^{1/3} \leq \frac{t}{D}$, or equivalently, $t \geq D^{3/2}$. At this operating point, the online complexity becomes $T_K = t\sqrt{\frac{t}{t^{1/3}}} = t^{4/3}$. Furthermore, $K = t^{1/3} = T_K^{1/4}$. Thus, the proposed construction uses $T_K^{1/4}$ parallel quantum processors at this operating point.Substituting $K = T_K^{1/4}$ into the processor-aware trade-off relation yields $N^2 = T_K^{4/3} M^2 D^2 \left(T_K^{1/4}\right)^{2/3} = T_K^{3/2} M^2 D^2$ in the parameter regime $T_K \geq D^2$, together with the underlying requirement that K be an admissible number of parallel quantum processors.

Remark 1. For asymptotic analysis, $K = t^{1/3}$ denotes the balanced scaling; in an implementation, K may be chosen as the nearest admissible power of two.

6 Distributed Grover on Rainbow Attack

In this section, we incorporate the distributed Grover search paradigm of Qiu *et al.* [12] into the Quantum Rainbow attack of Dunkelman *et al.* [11]. The construction follows the same distributed-search principle used for the Quantum Hellman attack, while taking into account the position-dependent functions that characterize Rainbow tables.

In the Quantum Rainbow attack, the online procedure operates on a superposition of D target data points and t/D possible starting positions (flavours). Thus, the effective search space has size $D \cdot \frac{t}{D} = t$. The function F_{Rainbow} used in the *GroverMatch* procedure requires $O(t)$ function evaluations. Consequently, the single-processor online complexity is $T_1 = O(t\sqrt{t}) = O(t^{3/2})$, which gives the same asymptotic quantum TMDTO relation as the Quantum Hellman construction.

The distributed construction partitions the flavour space among $K = 2^k$ parallel quantum processors. Each processor therefore operates on a smaller local search space, while the Rainbow table construction and its preprocessing assumptions remain unchanged.

6.1 Offline Preprocessing Phase

The offline preprocessing phase is identical to that of the Quantum Rainbow construction described in Section 4. In the multiple-data quantum traversal, the relevant set of starting positions has size $S = \frac{t}{D}$. These positions correspond to the admissible entry points in the existing Rainbow construction; the underlying Rainbow table and its position-dependent functions remain unchanged.

Unlike the Hellman construction, the Rainbow construction does not require multiple independent tables corresponding to different flavours. Instead, each

zero-based flavour label ℓ identifies the Rainbow starting position $\ell+1$ at which the target data point enters the reverse Rainbow traversal. In the distributed setting, these flavour indices are partitioned among the processors.

As in the Distributed Quantum Hellman attack, we partition the flavour space over K quantum processors. Let $K = 2^k$, where $0 \leq k \leq \log_2(t/D)$. The t/D flavour indices are represented using $\log_2(t/D)$ bits. In this work, the first k bits determine the processor, with the integer value represented by these bits identifying the corresponding processor, while the remaining $\log_2(t/D) - k$ bits identify the flavours within that processor. Thus, the flavour space is partitioned into K disjoint subsets, each containing $\frac{t}{KD}$ flavours.

In other words, let $\mathcal{F} = \{0, 1, \ldots, S-1\}$, $S = \frac{t}{D}$, denote the set of admissible entry points. We now partition $\mathcal{F}$ into $K = 2^k$ disjoint subsets $\mathcal{F} = \mathcal{F}_0 \cup \mathcal{F}_1 \cup \cdots \cup \mathcal{F}_{K-1}$, where $|\mathcal{F}_j| = \frac{S}{K} = \frac{t}{KD}$. Each $\ell \in \mathcal{F}$ specifies the flavour label which is zero-based and $\ell+1$ is the starting position of the reverse Rainbow traversal.

The partition is logical and applies only to the flavour/search space. Each processor is associated with one subset $\mathcal{F}_j$. It does not modify the underlying Rainbow table. Each processor is assumed to have access to the QACM information required to evaluate the match predicate for its assigned flavour subset. Thus, the total logical QACM size remains M, while the physical mechanism for providing this shared access, or partitioning where possible, or replication, is outside the scope of the present model.

6.2 Distributed GroverMatch: Online Phase

For processor j, let $\mathcal{F}_j$ denote its assigned subset of flavour indices. The corresponding function $F^{(j)}_{\text{Rainbow}}$ is the same Rainbow evaluation function used in the single-processor construction, with its flavour register restricted to $\mathcal{F}_j$.

For a target data point $d \in \mathcal{D}$ and $\ell \in \mathcal{F}_j$, the function first performs the appropriate position-dependent sequence of Rainbow functions beginning at position $\ell + 1$. It then repeatedly applies the complete Rainbow transformation over the prescribed number of iterations. Once a distinguished point is reached, the corresponding state is frozen; otherwise, the next complete Rainbow transformation is applied. The resulting distinguished point is used for the QACM matching operation. The procedure is summarized in Algorithm 8.

Algorithm 8: $F^{(j)}_{\text{Rainbow}}$ for the Distributed Rainbow attack

Input: A target data point $d \in \mathcal{D}$ and a flavour $\ell \in \mathcal{F}_j$.
Output: The distinguished point reached from d under flavour ℓ, or the final value after the prescribed number of iterations.

Set $d' \leftarrow d$;
for $i = 0$ **to** $S - 1$ **do**
  if $i \geq \ell$ **then**
    $d' \leftarrow f_{i+1}(d')$;
  end
end
for $i = 1$ **to** $8t/S$ **do**
  if d' *is a distinguished point* **then**
    $d' \leftarrow d'.$;
  end
  else
    $d' \leftarrow f_S(f_{S-1}(\cdots f_1(d')\cdots)).$;
  end
end
return $d'.$;

The number of function evaluations performed by $F^{(j)}_{\text{Rainbow}}$ is $O(t)$, as in the original Quantum Rainbow construction. In particular, the initial position-dependent traversal requires at most $S = t/D$ function evaluations, while the $8t/S$ complete Rainbow rounds require $8t$ function evaluations. Therefore, restricting the flavour register to $\mathcal{F}_j$ does not change the asymptotic cost of evaluating the function for a fixed pair (d, ℓ).

For each processor, we define the local match predicate as follows.

$$H_j(d,\ell) = \begin{cases} 1, & \text{if } F^{(j)}_{\text{Rainbow}}(d,\ell) \text{ matches an entry in the QACM table,} \\ 0, & \text{otherwise.} \end{cases}$$

The K processors evaluate their corresponding local search spaces in parallel. Quantum counting is first applied to the local spaces to identify a processor j^* whose local subspace is likely to contain a marked pair. Grover search is then performed on the selected processor using the predicate H_{j^*}.

The resulting procedure is summarized in Algorithm 9. A corresponding schematic diagram is given in Figure 3.

Algorithm 9: Distributed Quantum Rainbow Online Phase

Input: A set of D target data points $\mathcal{D}$; the Rainbow table stored in QACM; the flavour space $\mathcal{F}$ of size t/D; and $K = 2^k$ parallel quantum processors.

Output: A pair (d, ℓ) corresponding to a valid match and the recovered preimage.

Partition $\mathcal{F}$ into K disjoint subsets $\mathcal{F}_0, \ldots, \mathcal{F}_{K-1}$, where $|\mathcal{F}_j| = t/(KD)$.;

foreach *processor* $j \in \{0, 1, \ldots, K-1\}$ *in parallel* **do**

  Prepare a superposition over $\mathcal{D} \times \mathcal{F}_j$.;

  Apply $F^{(j)}_{\text{Rainbow}}$ to obtain the corresponding distinguished points.;

  Apply quantum counting to estimate the number of marked pairs in the local search space.;

end

Select a processor j^* with the highest quantum-counting estimate, which indicates a nonempty local search space;

On processor j^*, apply Grover search using

$$H_{j^*}(d, \ell) = \begin{cases} 1, & \text{if } F^{(j^*)}_{\text{Rainbow}}(d, \ell) \text{ matches an entry in the QACM table,} \\ 0, & \text{otherwise,} \end{cases}$$

to obtain a marked pair (d, ℓ).;

Retrieve the corresponding chain information from the QACM table and recompute the Rainbow chain classically to recover the preimage of d.;

return (d, ℓ) *and the recovered preimage*.;

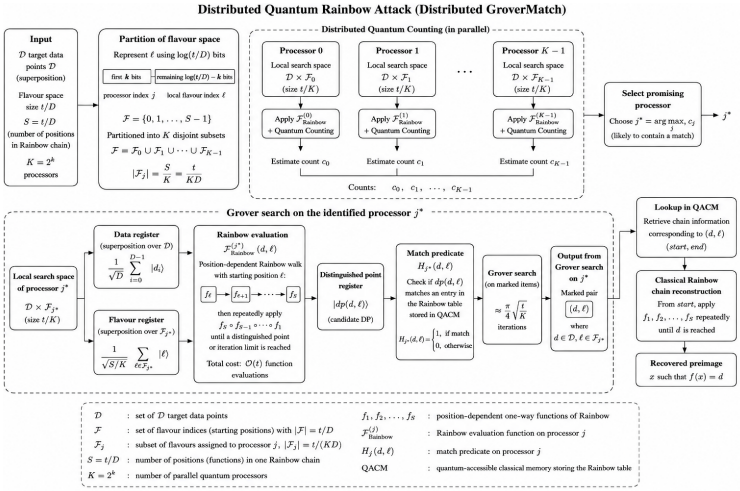

Fig. 3: Distributed GroverMatch procedure for the proposed Distributed Quantum Rainbow attack. The flavour space is partitioned among $K = 2^k$ parallel processors, followed by quantum counting to identify a promising processor j^* and Grover search over its local search space $\mathcal{D} \times \mathcal{F}_{j^*}$. The matched entry is then used for classical Rainbow chain reconstruction.

6.3 Processor-Aware TMDTO Relation for Distributed Quantum Rainbow

Proposition 2. *The Distributed Quantum Rainbow construction achieves the processor-aware relation given in Theorem 1.*

Proof. The multiple-data Quantum Rainbow construction uses t/D possible starting positions, giving an effective search space of size $D \cdot \frac{t}{D} = t$. In the distributed construction, the flavour space is partitioned among K parallel quantum processors. Each processor therefore operates on a local search space of size $D \cdot \frac{t}{KD} = \frac{t}{K}$. The evaluation of the corresponding Rainbow traversal $F^{(j)}_{\text{Rainbow}}$ requires $O(t)$ function evaluations. Consequently, the parallel wall-clock complexity of the Grover search is $T_K = O\left(t\sqrt{\frac{t}{K}}\right) = O\left(\frac{t^{3/2}}{\sqrt{K}}\right)$. Ignoring constant factors, we can write $T_K = \frac{t^{3/2}}{\sqrt{K}}$. Using the preprocessing relation $t = \frac{N}{MD}$, we obtain $T_K = \frac{1}{\sqrt{K}}\left(\frac{N}{MD}\right)^{3/2}$. Rearranging, we get $T_K^{2/3} = \frac{N}{MDK^{1/3}}$, and hence $N^2 = T_K^{4/3}M^2D^2K^{2/3}$. For $K = 1$, the processor-aware relation reduces to $N^2 = T_1^{4/3}M^2D^2$, which is the single-processor quantum Rainbow TMDTO relation.

To determine the operating point considered in this work, we impose the balancing condition. The fundamental processor constraint is that the number of parallel processors cannot exceed the available flavour space. Since the total number of flavour indices is t/D, we require $K \leq \frac{t}{D}$.

To obtain a representative balanced operating point, we choose the number of processors such that the number of processors is of the same asymptotic order as the number of Grover iterations performed by each processor. Thus, we can set $K = \Theta\left(\sqrt{\frac{t}{K}}\right)$, which gives $K = \Theta(t^{1/3})$. For simplicity, we consider the representative scaling $K = t^{1/3}$. This should be viewed as a choice of operating point rather than as a fundamental feasibility constraint.

This choice is feasible provided that $t^{1/3} \leq \frac{t}{D}$, or equivalently, $t \geq D^{3/2}$. At this operating point, substituting $K = t^{1/3}$ into the online complexity gives $T_K = t\sqrt{\frac{t}{t^{1/3}}} = t^{4/3}$. Furthermore, $K = t^{1/3} = T_K^{1/4}$. Thus, the proposed construction uses $T_K^{1/4}$ parallel quantum processors at this operating point.

Substituting $K = T_K^{1/4}$ into the processor-aware trade-off relation gives $N^2 = T_K^{4/3}M^2D^2\left(T_K^{1/4}\right)^{2/3} = T_K^{3/2}M^2D^2$. Hence, the resulting specialized time–memory–data trade-off relation is $N^2 = T_K^{3/2}M^2D^2$.

Since the operating point requires $t \geq D^{3/2}$, and $T_K = t^{4/3}$, the corresponding condition in terms of the online complexity is $T_K \geq D^2$. Therefore, the specialized trade-off $N^2 = T_K^{3/2}M^2D^2$ is considered in the parameter regime $T_K \geq D^2$, together with the underlying requirement that K be an admissible number of parallel quantum processors.

Remark 2. For asymptotic analysis, $K = t^{1/3}$ denotes the balanced scaling; in an implementation, K may be chosen as the nearest admissible power of two.

For both the Distributed Quantum Hellman and Distributed Quantum Rainbow constructions, the quantum-counting stage requires $O\left(\sqrt{\frac{t}{K}}\right)$ queries to the underlying match predicate. Since each predicate evaluation requires $O(t)$ function evaluations, the corresponding wall-clock complexity is $O\left(t\sqrt{\frac{t}{K}}\right)$.

Thus, under the present idealized model and ignoring constant-factor differences, the quantum-counting stage and subsequent GroverMatch search have the same asymptotic wall-clock order.

7 Comparison with Grover's Algorithm

In the quantum setting, the appropriate generic benchmark for function inversion is Grover's search algorithm. The purpose of this section is to determine the parameter regime in which the proposed distributed time–memory–data trade-off provides a smaller online wall-clock complexity than generic Grover search.

For the comparison, we consider the specialized operating point of the proposed distributed construction developed in Sections 5.4 and 6.3, for which $K = t^{1/3}$, $T_K = t^{4/3}$, and the processor-aware trade-off relation becomes $N^2 = T_K^{3/2} M^2 D^2$. The corresponding feasibility condition is $T_K \geq D^2$.

7.1 Comparison with Single-Processor Grover at a Fixed Number of Target Data Points

In this section, we first compare the proposed distributed construction with the standard single-processor Grover baseline. Grover's algorithm does not require preprocessing or memory. For multiple target data points, the corresponding time–data trade-off relation [11] is $N^2 = T_G^4 D^2$, where T_G denotes the online complexity of Grover search.

For the proposed distributed construction, the specialized time–memory–data relation is $N^2 = T_K^{3/2} M^2 D^2$. At the transition point where the two approaches have the same online wall-clock complexity, we set $T_K = T_G = \left(\frac{N}{D}\right)^{1/2}$.

Substituting this value into the proposed trade-off relation gives $M = \left(\frac{N}{D}\right)^{5/8}$. Consequently, for fixed N and D, the proposed distributed construction outperforms Grover search in online wall-clock time when $M > \left(\frac{N}{D}\right)^{5/8}$. At equality, the two approaches have the same asymptotic online wall-clock complexity.

For comparison, the corresponding transition condition for the single-processor quantum TMDTO construction of Dunkelman *et al.* [11] is $M = \left(\frac{N}{D}\right)^{2/3}$. Thus, at a fixed N and D, the proposed distributed construction requires a smaller preprocessing-memory exponent to reach the transition point with Grover search.

7.2 Comparison with an Equal Processor Budget

The distributed TMDTO construction is being allowed to use K processors, while the Grover baseline is not. Hence, one may ask what happens if generic Grover search is also parallelized across the same K processors? In this case, the wall-clock complexity would scale as $T_{G,K} = \sqrt{\frac{N}{KD}}$. At the operating point, $K = t^{1/3} = \left(\frac{N}{MD}\right)^{1/3}$. And $T_K = t^{4/3} = \left(\frac{N}{MD}\right)^{4/3}$. Combining these expressions, we obtain $T_K < T_{G,K}$ provided that $M > \left(\frac{N}{D}\right)^{2/3}$.

7.3 Boundary Transition Point under the Feasibility Constraint

We next determine the transition point that minimizes the online wall-clock complexity while remaining within the admissible parameter regime of the proposed construction.

At the transition point, $T_K = \left(\frac{N}{D}\right)^{1/2}$. And the specialized distributed construction must satisfy $T_K \geq D^2$. To obtain the smallest possible online complexity, we consider the boundary of the admissible region, $T_K = D^2$. Equating these two expressions for T_K gives $D = N^{1/5}$.Substituting this value into the expression for the Grover complexity gives $T_K = N^{2/5}$. The corresponding memory requirement at the transition point is $M = N^{1/2}$. Thus, at the transition point between the proposed distributed construction and Grover search, the relevant parameters scale as $D = N^{1/5}$, $T_K = N^{2/5}$, and $M = N^{1/2}$. The corresponding number of parallel quantum processors at the chosen operating point is $K = T_K^{1/4}$, and hence $K = N^{1/10}$.

For comparison, the single-processor quantum TMDTO construction of Dunkelman et al. [11] has the transition point $D = N^{1/4}$, and $T_1 = N^{3/8}$. The two transition points correspond to different values of the available data parameter D and therefore should not be interpreted as a direct fixed D comparison.

The principal advantage of the proposed distributed construction is instead observed at fixed N and D, for the same search space and number of target data points, the memory threshold required to outperform Grover is reduced from $\left(\frac{N}{D}\right)^{2/3}$ for the single-processor quantum TMDTO construction to $\left(\frac{N}{D}\right)^{5/8}$ for the proposed distributed construction. In addition, for fixed N, M, and D, the distributed construction reduces the online wall-clock time by the factor $\sqrt{K}$ under the idealized parallel model.

7.4 Advantage Window for Single Processor and Equal Processor Budget

We characterize the exact memory range for which the distributed section outperforms Grover search, under both single and equal processor budget baseline.

Corollary 1. *In the case of **Single Processor Advantage Window**, for a fixed N and D, at the operating point $K = t^{1/3}$, the distributed construction*

satisfies $T_K < T_G$ if and only if $\left(\frac{N}{D}\right)^{5/8} < M \leq \frac{N}{D^{5/2}}$. The window is nonempty iff $D < N^{1/5}$.

Corollary 2. *In the case of **Equal Processor Budget Advantage Window**, Under the fair baseline $T_{G,K} = \left(\frac{N}{KD}\right)^{1/2}$,the distributed construction satisfies $T_K < T_{G,K}$ iff $\left(\frac{N}{D}\right)^{2/3} < M \leq \frac{N}{D^{5/2}}$. The window is nonempty iff $D < N^{2/11}$.*

Remark 1 *Writing $y = \log_N D$,the two window width is $\Delta_{\text{single}}(y) = \frac{3}{8}(1-5y)$ and $\Delta_{\text{fair}}(y) = \frac{1}{3}(1-\frac{11}{2}y)$. Two observations follow directly:*

1. *At $y = 0(D = 1)$, $\Delta_{\text{fair}} < \Delta_{\text{single}}$.This matches the intuition that beating a K processor baseline is strictly harder than beating the single processor and the memory required to do so is higher.*
2. *The distributed construction provides a wall-clock advantage over equal-budget Grover when the available data satisfies $D < N^{2/11}$.This condition is more restrictive than the $D < N^{1/5}$ required to outperform the weaker single-processor baseline. When D exceeds $N^{2/11}$, the wall-clock advantage over equal-budget Grover disappears, although the comparison with the single-processor baseline may remain favorable up to $D = N^{1/5}$.*

8 Conclusion

In this paper, we proposed distributed variants of the quantum Hellman and quantum Rainbow time–memory–data trade-off attacks by incorporating a distributed Grover search over the flavour space. The proposed construction does not reduce the total quantum query complexity of the underlying *GroverMatch* procedure; rather, it reduces the parallel online wall-clock time by distributing the search over K parallel quantum processors.

For both the Distributed Quantum Hellman and Distributed Quantum Rainbow constructions, we first derived the processor-aware relation $N^2 = T_K^{4/3} M^2 D^2 K^{2/3}$, where T_K denotes the parallel online wall-clock complexity and K denotes the number of parallel quantum processors. At the operating point considered in this work, $K = t^{1/3}$, the online complexity becomes $T_K = t^{4/3}$, and the processor-aware relation specializes to $N^2 = T_K^{3/2} M^2 D^2$.

We also answered a natural question that the advantages of distributed settings with equal processor budget.In that case a genuine advantage survives but in a smaller data regime where a nonempty range of M yields a wall clock advantage over equal budget Grover.

This specialized relation should be interpreted as a wall-clock time–memory–data trade-off obtained through parallel quantum computation, rather than as an improvement of the single-processor quantum TMDTO bound.

We further compared the proposed construction with generic Grover search. At the transition point between the two approaches, the required memory satisfies $M = \left(\frac{N}{D}\right)^{5/8}$. Consequently, within the parameter regime of the proposed distributed construction, the distributed TMDTO attack has a smaller online wall-clock complexity than Grover search when $M > \left(\frac{N}{D}\right)^{5/8}$. This memory threshold

is lower than the corresponding threshold $\left(\frac{N}{D}\right)^{2/3}$ for the single-processor quantum TMDTO construction considered in Dunkelman *et al.* [11].

It is important, however, to distinguish this reduction in the memory threshold from the dependence of online time on memory. From $N^2 = T_K^{3/2} M^2 D^2$, the online wall-clock complexity scales as $T_K \propto M^{-4/3}$. Thus, doubling the memory reduces the online wall-clock time by a factor of $2^{4/3} \approx 2.52$. For comparison, the corresponding factor for the single-processor quantum TMDTO relation $N^2 = T^{4/3} M^2 D^2$ is $2^{3/2} \approx 2.83$, while the classical relation $N^2 = T M^2 D^2$ gives a factor of $2^2 = 4$.

Therefore, the principal advantage of the proposed construction is not a stronger time reduction per doubling of memory, but the reduction in the memory threshold required to outperform generic Grover search, together with the reduction in parallel online wall-clock time obtained by using multiple quantum processors.

However, one should note that when generic Grover search is provided with the same number of parallel quantum processors, the corresponding transition condition becomes $M > (N/D)^{2/3}$. Thus, the reduced 5/8 threshold reflects the comparison with the standard single-processor Grover baseline, whereas the principal benefit under an equal processor budget is the processor-aware reduction in parallel wall-clock time.

References

1. Martin E. Hellman. A cryptanalytic time-memory trade-off. *IEEE Transactions on Information Theory*, 26(4):401–406, 1980.
2. Steven H. Babbage. Improved "exhaustive search" attacks on stream ciphers. In *European Convention on Security and Detection*, pages 161–166, 1995.
3. Jovan Dj. Golić. Cryptanalysis of alleged A5 stream cipher. In *Advances in Cryptology–EUROCRYPT 1997*, volume 1233 of *Lecture Notes in Computer Science*, pages 239–255. Springer, 1997.
4. Alex Biryukov and Adi Shamir. Cryptanalytic time/memory/data tradeoffs for stream ciphers. In *Advances in Cryptology–ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2000.
5. Alex Biryukov, Adi Shamir, and David Wagner. Real time cryptanalysis of A5/1 on a PC. In *Fast Software Encryption 2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2000.
6. Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, pages 212–219. ACM, 1996.
7. Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002.
8. Aran Nayebi, Scott Aaronson, Aleksandrs Belovs, and Luca Trevisan. Quantum lower bound for inverting a permutation with advice. *Quantum Information and Computation*, 15(11–12):901–913, 2015.
9. Minki Hham, Keita Xagawa, and Takashi Yamakawa. Quantum random oracle model with auxiliary input. In *Advances in Cryptology–ASIACRYPT 2019*, volume 11921 of *Lecture Notes in Computer Science*, pages 584–614. Springer, 2019.

10. Kai-Min Chung, Tsung-Han Liao, and Li Qian. Lower bounds for function inversion with quantum advice. In *1st Conference on Information-Theoretic Cryptography (ITC 2020)*, volume 163 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 8:1–8:15. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020.
11. Orr Dunkelman, Nathan Keller, Eyal Ronen, and Adi Shamir. Quantum time/memory/data tradeoff attacks. *Designs, Codes and Cryptography*, 92(1):159–177, 2024. doi:10.1007/s10623-023-01300-x.
12. Daowen Qiu, Lili Luo, and Ligang Xiao. Distributed Grover's algorithm. *Theoretical Computer Science*, 993:114461, 2024. doi:10.1016/j.tcs.2024.114461.
13. Hugo Jacinto, Élie Gouzien, and Nicolas Sangouard. Network requirements for distributed quantum computation. *Physical Review Research*, 8:013205, 2026. doi:10.1103/v9ln-c4v2.
14. J. R. Jiang, T. Y. Wang, W. H. Huang, and J. Z. Zhang. Distributed Shor's algorithm with sequential quantum teleportation. In *2023 IEEE 5th Eurasia Conference on IoT, Communication and Engineering (ECICE)*, pages 479–484. IEEE, 2023.
15. Ligang Xiao, Daowen Qiu, Le Luo, and Paulo Mateus. Distributed Shor's algorithm. *Quantum Information and Computation*, 23(1–2):27–44, 2023. doi:10.26421/QIC23.1-2-3.
16. Mohamadali Khajeian. A hybrid classical-quantum rainbow table attack on human passwords. *Journal of Algorithms and Computation*, 57(1):134–145, 2025. doi:10.22059/jac.2025.398812.1237.
17. Philippe Oechslin. Making a faster cryptanalytic time-memory trade-off. In *Advances in Cryptology–CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 617–630. Springer, 2003.
18. Elad Barkan, Eli Biham, and Adi Shamir. Rigorous bounds on cryptanalytic time/memory tradeoffs. In *Advances in Cryptology–CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 1–21. Springer, 2006.
19. Dorothy E. Denning. *Cryptography and Data Security*. Addison-Wesley, Reading, MA, 1982.