quantum reports

IMPACT FACTOR 1.3

CITESCORE 3.0

Review

Theory and Applications of Quantum Hashing

Farid Ablayev, Kamil Khadiev, Alexander Vasiliev and Mansur Ziiatdinov

Special Issue
100 Years of Quantum Mechanics

Edited by
Prof. Dr. Viktor Dodonov, Prof. Dr. Margarita A. Man'ko, Prof. Dr. Salomon S. Mizrahi and Prof. Dr. Luis L. Sánchez-Soto

MDPI

https://doi.org/10.3390/quantum7020024

quantum reports

Review

Theory and Applications of Quantum Hashing

Farid Ablayev [1,2], Kamil Khadiev [1,2,*], Alexander Vasiliev [1,2,*] and Mansur Ziiatdinov [3]

[1] Zavoisky Physical-Technical Institute, FRC Kazan Scientific Center of RAS, 10/7, Sibirskiy Trakt Str., Kazan 420029, Russia; farid.ablayev@kpfu.ru
[2] Institute of Computational Mathematics and Information Technologies, Kazan Federal University, 18 Kremlevskaya Str., Kazan 420008, Russia
[3] MIFT Department, University of Messina, Viale Ferdinando Stagno d'Alcontres, 31, 98166 Messina, Italy; mansur.ziiatdinov@unime.it
[*] Correspondence: kamil.hadiev@kpfu.ru (K.K.); alexander.vasiliev@kpfu.ru (A.V.)

Abstract: We review recent results on quantum one-way functions, including quantum fingerprinting or quantum hashing (we use these two terms as synonyms even though they have very small difference). This includes the analysis of their properties, different modifications, circuit implementation on an IBM Q platform, as well as on an experimental quantum setup. We discuss computational aspects of quantum hashing, its cryptographic properties and possible usage in communication protocols and algorithms.

Keywords: quantum hashing; quantum computing; QFT; quantum algorithms; quantum cryptography; string processing

check for updates

Academic Editor: Viktor Dodonov

Received: 22 March 2025
Revised: 7 May 2025
Accepted: 7 May 2025
Published: 15 May 2025

Citation: Ablayev, F.; Khadiev, K.; Vasiliev, A.; Ziiatdinov, M. Theory and Applications of Quantum Hashing. *Quantum Rep.* **2025**, *7*, 24. https://doi.org/10.3390/quantum7020024

1. Introduction

Quantum computing [1–3] has been one of the hottest topics in computer science over the last few decades. There are many problems in which quantum algorithms outperform the best known classical ones [4]. One of the useful techniques for constructing algorithms in different computational models is quantum hashing or quantum fingerprinting. In classical computation, hash functions are widely used in algorithms and data structure design [5,6]. As an example, we can consider the Rabin–Karp algorithm for the string matching problem [7], the hash table data structure [8–10], and the comparison of objects in standard libraries of programming languages [11]. One of the popular applications is cryptography [6,12]. Cryptography applications involve the use of one-way functions as hash functions because of their cryptographic properties.

In quantum computing, hash functions also have many applications in algorithm design, including cryptography [13–19], where one-way property is crucial. Quantum one-way functions can be categorized into four types. The categorization is based on their input–output formats:

- Classical–classical: These functions retain a classical nature but satisfy the properties of quantum one-wayness. Their relevance is demonstrated in statistical zero-knowledge protocols and quantum oracle indistinguishability tasks [20,21].
- Classical–quantum: This type is extensively explored in the context of quantum hash functions [22–25]. Applications include signing classical data [26] and quantum digital signature schemes [27].
- Quantum–classical: These functions enable the authentication of quantum states via classical strings. The work of Behera et al. illustrates their utility in enhanced quantum money schemes [28].

- Quantum–quantum: These directly transform quantum states into other quantum states. The first implementation, proposed by Shang et al. [29], combines "quantum–classical" and "classical–quantum" quantum one-way functions. This approach is successfully applied in quantum identity authentication protocols, where one quantum state verifies the authenticity of another.

In this paper, we focus on the Classical–quantum type of one-way functions as quantum hash functions.

Another name for the same technique is "fingerprinting". The probabilistic (randomized) technique was developed by Frievalds [30,31]. The main idea of the technique is to randomly choose one of specially constructed hash functions from a given set and to use it for comparing objects for equality. Here, the value of a hash function is called a "fingerprint" of an object.

A thoughtful reader can see a difference between the "hash function technique" and the "fingerprinting technique". At the same time, the nature and key idea of the techniques are the same. That is why we use these terms as synonyms in this paper.

The quantum version of the fingerprinting technique was developed by Ambainis and Frievalds [32] for automata and later improved by Ambainis and Nahimovs in [33,34]. Later, Buhrman et al. in [22] provided an explicit definition of the quantum fingerprinting algorithm to construct an efficient quantum communication protocol for equality testing. The technique was applied to branching programs by Ablayev, Vasiliev, and Gainutdinova [35–37], this computational model can be considered as a non-uniform automata. They show examples of Boolean functions that can be computed by a quantum branching program with exponentially smaller complexity (width or states in terms of automata) than the deterministic ones. Based on this research, they developed the concept of cryptographic quantum hashing [23,24]. Later, they generalized the technique and suggested a family of quantum hashes that allow us to see a trade-off between one-way resistance and collision resistance of the hash function [25,38,39]. Different versions of hash functions were applied that are hash functions in the case of finite abelian groups [40] and arbitrary groups [41], as well as functions based on graphs [26,42]. An alternative approach for cryptographic ideas was presented in [43]. The connection of the approach with the quantum Fourier transform is presented in [44,45]. We define the concept of quantum hashing and discuss its properties in Section 3.

Similarly to classical hash functions, quantum hash functions have parameters that allow them to satisfy required properties and to be useful for comparing objects. We discuss procedures for computing parameters of the quantum hash functions in Section 5.

Turaykhanov, Akat'ev, and coauthors implemented a technique in a photon-based real quantum device [46]. The technique was extended to qudits, and this extension was used in the experimental implementation [47]. Alternative photon-based real quantum device implementation was developed by Plachta, Hiekkamäki, Yakaryılmaz, and Fickler [48] and Zhao, Li, Li, Zheng, and He [49]. The experiments and the qudit version of the method are presented in Section 4.

In the discussed experiments, the algorithm was embedded in this device. When we discuss the implementation of the algorithm for "universal" quantum devices such as IBMQ quantum computers or similar, we should represent the algorithm as a quantum circuit. It is important to minimize the number of quantum gates in the circuit with respect to the architecture restrictions of a quantum device. Optimization for quantum computer emulators was explored in [50]. A shallow circuit for the algorithm was suggested in [51,52]. Its implementation on a real quantum computer's noisy simulators was discussed in [45,53–55]. We present the results in Section 6.

This approach has been widely used in various areas such as stream processing algorithms [56], query model algorithms [57,58], online algorithms [59,60], quantum online algorithms with restricted size of memory [61,62], branching programs [63,64], development of quantum devices [65], automata [66–68], and others. Several applications are discussed in Section 7.

2. Preliminaries and Definitions

Here, we present short notes on quantum computation. The reader can find more information about quantum circuits in [1,3].

The notion of a quantum bit (qubit) is the basis of quantum computations. The qubit is the quantum version of the classical binary bit physically realized with a two-state device. Formally, the qubit's state is the column vector $|\psi\rangle$ of two-dimensional Hilbert space $\mathcal{H}^2$, that is, $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. Here, the pair of vectors $|0\rangle$ and $|1\rangle$ is an orthonormal basis of $\mathcal{H}^2$, where $\alpha, \beta \in \mathbb{C}$ such that $|\alpha|^2 + |\beta|^2 = 1$. The numbers α, β are called amplitudes. Consider a qubit in a state such that $|\alpha|^2 + |\beta|^2 = 1$. Then, we can represent the amplitudes as $|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle$, where $0 \le \phi < 2\pi, 0 \le \theta \le \pi$.

A quantum state $|\psi\rangle$ of a quantum n-qubit register is a complex-valued unit vector in 2^n-dimensional Hilbert space $\mathcal{H}^{2^n}$ that is described as a linear combination of basis vectors: $|i\rangle, i \in \{0, \dots, 2^n - 1\}$: $|\psi\rangle = \sum\limits_{i=0}^{2^n-1} \alpha_i|i\rangle$, with $\sum\limits_{i=0}^{2^n-1} |\alpha_i|^2 = 1$.

Quantum mechanics postulates that transformations of quantum states $|\psi\rangle \in (\mathcal{H}^2)^{\otimes n}$ (of the quantum n-qubit register) are mathematically determined by unitary operators: $|\psi'\rangle = U|\psi\rangle$, where U is a $2^n \times 2^n$ unitary matrix for transforming a vector that represents a state of a quantum n-qubit register.

There is only one way to extract information from the state of a quantum n-register to a "macro world". It is a measurement of the state of the quantum register. If we measure the quantum state $\sum_i \alpha_i|i\rangle$, then we obtain one of the basis states, $|i\rangle$, with probability $|\alpha_i|^2$.

2.1. Quantum Computational Models

In this paper, we consider different computational models.

2.1.1. Quantum Circuits

One of the most common is the quantum circuit that consists of qubits and gates (unitary transformations). Graphically, on a circuit, qubits are presented as lines.

As basic gates, we consider the following ones:

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, R_y(\xi) = \begin{pmatrix} \cos(\xi/2) & -\sin(\xi/2) \\ \sin(\xi/2) & \cos(\xi/2) \end{pmatrix},$$
$$R_z(\xi) = \begin{pmatrix} e^{-\frac{i\xi}{2}} & 0 \\ 0 & e^{\frac{i\xi}{2}} \end{pmatrix}, P(\xi) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\xi} \end{pmatrix}, CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \quad (1)$$

Additionally, we consider five nonbasic gates:

$$R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{2^{k-1}}} \end{pmatrix}, CR_k = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{\frac{i\pi}{2^{k-1}}} \end{pmatrix}, CR_z(\xi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & e^{-\frac{i\xi}{2}} & 0 \\ 0 & 0 & 0 & e^{\frac{i\xi}{2}} \end{pmatrix},$$

$$SWAP = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, CR_y(\xi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos(\xi/2) & -\sin(\xi/2) \\ 0 & 0 & \sin(\xi/2) & \cos(\xi/2) \end{pmatrix}. \quad (2)$$

The two-qubit gates (for example, the CNOT gate) is the most "expensive" for physical implementation. By "expensive", we mean that both computational error and the difficulty of implementation on a real device are high. So, we consider the number of CNOT gates as a main complexity measure for a circuit. We call it CNOT cost.

At the same time, one two-qubit gate (CNOT gate as an example) and one-qubit gate are the basis for quantum circuits [69].

2.1.2. Quantum Query Model

Another popular model is the quantum query model. Let $f : D \to \{0,1\}, D \subseteq \{0,1\}^M$ be an M variable function. We wish to compute on an input $x \in D$. We are given an oracle access to the input x, i.e., it is realized by a specific unitary transformation usually defined as $|i\rangle|z\rangle|w\rangle \to |i\rangle|z + x_i \pmod 2\rangle|w\rangle$, where the $|i\rangle$ register indicates the index of the variable we are querying, $|z\rangle$ is the output register, and $|w\rangle$ is some auxiliary work space. An algorithm in the query model consists of alternating applications of arbitrary unitary transformations independent of the input and the unitary query, and a measurement in the end. The smallest number of queries for an algorithm that outputs $f(x)$ with probability $\geq \frac{2}{3}$ on all x is called the quantum query complexity of the function f.

The model can be considered as a circuit with alternating unitary transformations of two types. The first one is a regular unitary transformation, the second one is a transformation that is controlled by input variables.

Search problem

Suppose that we have a set of objects named $\{1, 2, \ldots, M\}$, of which some are targets. Suppose that $\mathcal{O}$ is an oracle that identifies the targets. The goal of a search problem is to find a target $i \in \{1, 2, \ldots, M\}$ by making queries to the oracle $\mathcal{O}$.

In search problems, one will try to minimize the number of queries to the oracle. In the classical setting, one needs $O(M)$ queries to solve such a problem. Grover, on the other hand, constructed a quantum algorithm that solves the search problem with only $O(\sqrt{M})$ queries [70], provided that there is a unique target. When the number of targets is unknown, Brassard et al. designed a modified Grover algorithm that solves the search problem with $O(\sqrt{M/\lambda})$ queries [71], where λ is the number of targets, which is of the same order as the query complexity of the Grover search.

2.1.3. Branching Programs

A quantum branching program (QBPs) is a known model of computations, a generalization of the classical branching program (BP) model of computations. We refer to [36,64] for more information on QBPs. Quantum branching programs and the quantum query model are closely related. They can be considered as a specific variant of each other depending on the point of view. For example, in the content of this work, QBPs can be considered as a special variant of the quantum query model, which can test only one input variable on a level of computation. Here, we define a QBP as found in [3,72]. Another point of view of the model is the data stream processing algorithms.

3. Quantum Hashing Technique

In this section, we present recent results on quantum hashing developed in our research group.

3.1. One-Way δ-Resistance

We present the following definition of a quantum δ-resistant one-way function. Let the "information extracting" mechanism M be a function $M : (\mathcal{H}^2)^{\otimes s} \to \mathbb{X}$. Informally speaking, the mechanism M makes some measurement to state $|\psi\rangle \in (\mathcal{H}^2)^{\otimes s}$ and decodes the result of the measurement to $\mathbb{X}$.

Definition 1. *Let X be a random variable distributed over $\mathbb{X}$ like $\{Pr[X = w] : w \in \mathbb{X}\}$. Let $\psi : \mathbb{X} \to (\mathcal{H}^2)^{\otimes s}$ be a quantum function. Let Y be any random variable over $\mathbb{X}$ obtained by some mechanism M, causing a measurement to encode ψ of X and decoding the measurement result into $\mathbb{X}$. Let $\delta > 0$. We call a quantum function ψ a one-way δ-resistant function if the following apply:*

1. *It is easy to compute, i.e., a quantum state $|\psi(w)\rangle$ for a particular $w \in \mathbb{X}$ can be determined using a polynomial-time algorithm;*
2. *For any mechanism M, the probability $Pr[Y = X]$ that M successfully decodes Y is bounded by δ:*

$$Pr[Y = X] \leq \delta. \tag{3}$$

For cryptographic purposes, it is natural to expect (and this is applied in the rest of the paper) that the random variable X is uniformly distributed.

A quantum state of $s \geq 1$ qubits can "carry" an infinite amount of information. On the other hand, the fundamental result of quantum informatics known as Holevo's Theorem [73] states that a quantum measurement can only give $O(s)$ bits of information about the state. Here, we use the result of [74] motivated by Holevo's Theorem.

Property 1. *Let X be a random variable uniformly distributed over $\{0,1\}^k$. Let $\psi : \{0,1\}^k \to (\mathcal{H}^2)^{\otimes s}$ be a $(2^k; s)$ quantum function. Let Y be a random variable over $\{0,1\}^k$ obtained by some mechanism M making some measurement of the encoding ψ of X and decoding the measurement result to $\{0,1\}^k$. Then, the probability of correct decoding is given by*

$$Pr[Y = X] \leq \frac{2^s}{2^k}. \tag{4}$$

3.2. Collision ϵ-Resistance

The following definition was presented in [75].

Definition 2. *Let $\epsilon > 0$. We call a quantum function $\psi : \mathbb{X} \to (\mathcal{H}^2)^{\otimes s}$ a collision ϵ-resistant function if for any pair w, w' of different inputs,*

$$\big|\langle\psi(w)\,|\,\psi(w')\rangle\big| \leq \epsilon. \tag{5}$$

Testing equality

The crucial procedure for quantum hashing is an equality test for $|\psi(v)\rangle$ and $|\psi(w)\rangle$ that can be used to compare encoded classical messages v and w; see, for example, [27].

This procedure can be a well-known SWAP-test [22] or one that is adapted for specific hashing functions, like the REVERSE-test; see [23].

3.3. Balanced Quantum (δ, ϵ)-Resistance

The above two definitions and considerations lead to the following formalization of the quantum cryptographic (one-way and collision resistant) function.

Definition 3. *Let $K = |\mathbb{X}|$ and $s \geq 1$. Let $\delta > 0$ and $\epsilon > 0$. We call a function $\psi : \mathbb{X} \to (\mathcal{H}^2)^{\otimes s}$ a quantum (δ, ϵ)-resistant $(K; s)$-hash function (or just quantum (δ, ϵ)-hash function) iff ψ is a one-way δ-resistant and a collision ϵ-resistant function.*

Let $K = \Sigma^m$ for some finite alphabet Σ, with an integer $m \gg s$. In which case, we call the function (m,ϵ,s)-quantum hash function if ψ is ϵ-resistant.

We present below the following two examples to demonstrate how one-way δ-resistance and collision ϵ-resistance are correlated. The first example was presented in [32] in terms of quantum automata.

Example 1. *Let us encode numbers v from $\{0, \dots, 2^k - 1\}$ by a single qubit as follows:*

$$\psi : v \mapsto \cos\left(\frac{2\pi v}{2^k}\right)|0\rangle + \sin\left(\frac{2\pi v}{2^k}\right)|1\rangle. \tag{6}$$

Extracting information from $|\psi\rangle$ by measuring $|\psi\rangle$ with respect to the basis $\{|0\rangle, |1\rangle\}$ gives the following result. The function ψ is one-way $\frac{2}{2^k}$-resistant (see Property 1) and collision $\cos\left(\pi/2^{k-1}\right)$-resistant. Thus, the function ψ has a good one-way property, but has a bad collision resistance property for large k.

Example 2. *Let $v = \sigma_1 \dots \sigma_k \in \{0,1\}^k$. We encode v by k qubits: $\psi : v \mapsto |v\rangle = |\sigma_1\rangle \cdots |\sigma_k\rangle$.*

Extracting information from $|\psi\rangle$ by measuring $|\psi\rangle$ with respect to the basis $\{|0\dots0\rangle, \dots, |1\dots1\rangle\}$ gives the following result. The function ψ is one-way 1-resistant and collision 0-resistant. So, in contrast to Example 1, the encoding ψ from Example 2 is collision-free, that is, for different words v and w, the quantum states $|\psi(v)\rangle$ and $|\psi(v)\rangle$ are orthogonal and therefore reliably distinguished; but we lose the one-way property, i.e., ψ is easily invertible.

The following result [75] shows that a quantum collision ϵ-resistant $(K; s)$ function needs at least $\log\log K - c(\epsilon)$ qubits.

Property 2. *Let $s \geq 1$ and $K = |\mathbb{X}| \geq 4$. Let $\psi : \mathbb{X} \to (\mathcal{H}^2)^{\otimes s}$ be a collision ϵ-resistant quantum hash function. Then,*

$$s \geq \log\log K - \log\log\left(1 + \sqrt{2/(1-\epsilon)}\right) - 1. \tag{7}$$

Proof. See [75] for the proof. □

Properties 1 and 2 provide a basis for building a "balanced" one-way δ-resistance and collision ϵ-resistance properties. That is, roughly speaking, if we need to hash elements w from the domain $\mathbb{X}$ with $|\mathbb{X}| = K$ and if one can build for an $\epsilon > 0$ a collision ϵ-resistant $(K; s)$ hash function ψ with $s \approx \log\log K - c(\epsilon)$ qubits, then the function f is one-way δ-resistant with $\delta \approx (\log K/K)$. Such a function is balanced with respect to Property 2.

To summarize the above considerations, we can state the following. A quantum (δ, ϵ)-hash function is a function that satisfies all of the properties that a "classical" hash function should satisfy. Pre-image resistance follows from Property 1. The second pre-image and collision resistance follow, because all inputs are mapped to states that are nearly orthogonal. Therefore, we see that quantum hash functions can satisfy the three properties of a classical cryptographic hash function.

3.4. Quantum (δ, ϵ)-Hash Function Construction via Small-Biased Sets

This section is based on [40]. We present here a brief background on ϵ-biased sets as defined in [76] and discuss their connection to quantum hashing. Note that ϵ-biased sets are generally defined for arbitrary finite groups, but here we restrict ourselves to $\mathbb{Z}_q$.

For an $a \in \mathbb{Z}_q$, a character χ_a of $\mathbb{Z}_q$ is a homomorphism $\chi_a : \mathbb{Z}_q \to \mu_q$, where μ_q is the (multiplicative) group of complex q-th roots of unity. That is, $\chi_a(x) = \omega^{ax}$, where $\omega = e^{\frac{2\pi i}{q}}$ is a primitive q-th root of unity. The character $\chi_0 \equiv 1$ is called a trivial character.

Definition 4. *A set $S \subseteq \mathbb{Z}_q$ is called ϵ-biased if for any nontrivial character $\chi \in \{\chi_a : a \in \mathbb{Z}_q\}$,*

$$\frac{1}{|S|}\left|\sum_{x\in S}\chi(x)\right| \leq \epsilon. \tag{8}$$

These sets are interesting when $|S| \ll |\mathbb{Z}_q|$ (as $S = \mathbb{Z}_q$ is 0-biased). In their seminal paper, Naor and Naor [77] defined these small-biased sets, gave the first explicit constructions of such sets, and demonstrated the power of small-biased sets for several applications.

Remark 1. *Note that a set S of $O(\log q/\epsilon^2)$ elements selected uniformly at random from $\mathbb{Z}_q$ is ϵ-biased with positive probability [78].*

Many other constructions of small-biased sets followed over the last few decades.

Vasiliev [40] showed that ϵ-biased sets generate (δ,ϵ)-resistant hash functions. We present the result of [40] in the following form.

Property 3. *Let $S \subseteq \mathbb{Z}_q$ be an ϵ-biased set. Let*

$$H_S = \{h_a(x) = ax \pmod q, \quad a \in S, h_a : \mathbb{Z}_q \to \mathbb{Z}_q\} \tag{9}$$

be a set of functions determined by S. Then, a quantum function $\psi_S : \mathbb{Z}_q \to (\mathcal{H}^2)^{\otimes \log|S|}$

$$|\psi_S(x)\rangle = \frac{1}{\sqrt{|S|}}\sum_{a\in S}\omega^{h_a(x)}|a\rangle \tag{10}$$

is a (δ, ϵ)-resistant quantum hash function, where $\delta \leq |S|/q$.

Proof. The one-way δ-resistance property of ψ_S follows from Property 1: the probability of correctly decoding an x from a quantum state $|\psi_S(x)\rangle$ is bounded by $|S|/q$. The efficient computability of such a function follows from the fact that any quantum transformation on s qubits (including the one that creates a quantum hash) can be performed with $O(s^2 4^s)$ elementary quantum gates [1]. Whenever $s = O(\log|S|) = O(\log\log q - \log\epsilon)$, this number of steps is polynomial in $\log q$ (the binary representation of group elements) and $1/\epsilon$.

The collision ϵ-resistance property of ψ_S follows directly from the corresponding property of [40]. Note that

$$|\psi_S(x)\rangle = \frac{1}{\sqrt{|S|}}\sum_{a\in S}\omega^{h_a(x)}|a\rangle = \frac{1}{\sqrt{|S|}}\sum_{a\in S}\chi_x(a)|a\rangle. \tag{11}$$

The remainder of this proof coincides with the proof of the paper [40]. □

Remark 2. *It is natural to call the set H_S of functions a uniform ϵ-biased quantum hash generator in the context of the definition of a quantum hash generator from [24] and the above considerations.*

As a corollary of Property 3 and the above considerations, we can state the following.

Property 4. *For a small sized ϵ-biased set $S = \{s_1, \ldots, s_d\} \subset \mathbb{Z}_q$ with $d = O(\log q/\epsilon^2)$, for $\delta \leq O(\frac{\log q}{\epsilon^2 q})$, a quantum hash generator H_S generates the balanced (δ, ϵ)-resistant quantum hash function ψ_S given by*

$$|\psi_S(a)\rangle = \frac{1}{\sqrt{d}} \sum_{j=1}^{d} \omega^{as_j} |j\rangle \,. \tag{12}$$

3.5. Quantum Hashing for Finite Abelian Groups

In [40], we proposed the notion of a quantum hash function, which is defined for arbitrary finite abelian groups.

Let G be a finite abelian group with characters χ_a, indexed by $a \in G$. Let $S \subseteq G$ be an ε-biased set for some $\varepsilon \in (0, 1)$.

Definition 5. *We define a quantum hash function $\psi_S : G \to (\mathcal{H}^2)^{\otimes \log |S|}$ as follows:*

$$|\psi_S(a)\rangle = \frac{1}{\sqrt{|S|}} \sum_{j=1}^{|S|} \chi_a(s_j) |j\rangle. \tag{13}$$

We have shown that ψ_S has all the properties of a cryptographic quantum hash function (i.e., it is quantum one-way and collision-resistant), which are entirely determined by the ε-biased set $S \subseteq G$.

There are two known special cases of quantum hashing for specific finite abelian groups, which turn out to be the known quantum fingerprinting method. In particular, we are interested in hashing binary strings, and thus, it is natural to consider $G = \mathbb{Z}_2^n$ and $G = \mathbb{Z}_{2^n}$ (or, more generally, any cyclic group $\mathbb{Z}_q$).

Hashing the elements of the Boolean cube.

For $G = \mathbb{Z}_2^n$, its characters can be written in the form $\chi_a(x) = (-1)^{(a,x)}$, and the corresponding quantum hash function is the following:

$$|\psi_S(a)\rangle = \frac{1}{\sqrt{|S|}} \sum_{j=1}^{|S|} (-1)^{(a,s_j)} |j\rangle \,. \tag{14}$$

The resulting hash function is exactly the quantum fingerprinting by Buhrman et al. [22], once we consider an error-correcting code, whose matrix is built from the elements of S. Indeed, as stated in [79] an ε-balanced error-correcting code can be constructed out of an ε-biased set. Thus, the inner product (a, x) in the exponent is equivalent to the corresponding bit of the codeword, and altogether, this gives the quantum fingerprinting function, which stores information in the phase of quantum states [80].

Hashing the elements of the cyclic group

For $G = \mathbb{Z}_q$, its characters can be written as $\chi_a(x) = \exp(2\pi i a x/q)$, and the corresponding quantum hash function is given by

$$|\psi_S(a)\rangle = \frac{1}{\sqrt{|S|}} \sum_{j=1}^{|S|} \omega^{a s_j} |j\rangle \,. \tag{15}$$

The above quantum hash function is essentially equivalent to the one we have defined earlier in [23], which is in turn based on the quantum fingerprinting function from [37].

3.6. Single-Qubit Version of Quantum Hashing

In [46], we constructed an OAM-based version of the quantum hash function, which was defined as follows.

For the classical input $x \in \{0, 1, \ldots, q-1\}$, we create its quantum image composed of s single-photon states:

$$|\psi_j(x)\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\frac{2\pi b_j x}{q}}|1\rangle), \tag{16}$$

$$|\psi(x)\rangle = |\psi_1(x)\rangle \otimes \cdots \otimes |\psi_s(x)\rangle, \tag{17}$$

where $b_j \in \{0, 1, \ldots, q-1\}$ are numeric parameters of the quantum hash function that provide its collision resistance.

The main idea behind quantum hashing is to provide the minimal fidelity of different quantum hash codes (collision resistance) with the minimal possible number of qubits (that affects the one-way property). As mentioned in [25], there is always a trade-off between these two properties, so balancing them in a quantum hash function is an important task.

The fidelity of $|\psi(x_1)\rangle$ and $|\psi(x_2)\rangle$ is given by $|\langle\psi(x_1)\,|\,\psi(x_2)\rangle|^2$, and for the OAM-based hash function above, it is equal to

$$\begin{aligned}
|\langle\psi(x_1)\,|\,\psi(x_2)\rangle|^2 &= \frac{1}{2^{2s}} \prod_{j=1}^{s} \left|1 + e^{i\frac{2\pi b_j (x_2 - x_1)}{q}}\right|^2 = \\
&= \frac{1}{2^{2s}} \prod_{j=1}^{s} \left[\left(1 + \cos\frac{2\pi b_j (x_2 - x_1)}{q}\right)^2 + \left(\sin\frac{2\pi b_j (x_2 - x_1)}{q}\right)^2\right] = \\
&= \frac{1}{2^{2s}} \prod_{j=1}^{s} \left(2 + 2\cos\frac{2\pi b_j (x_2 - x_1)}{q}\right) = \\
&= \frac{1}{2^{s}} \prod_{j=1}^{s} \left(1 + \cos\frac{2\pi b_j (x_2 - x_1)}{q}\right) = \\
&= \frac{1}{2^{s}} \prod_{j=1}^{s} \left(2\cos^2\frac{\pi b_j (x_2 - x_1)}{q}\right) = \\
&= \prod_{j=1}^{s} \cos^2\frac{\pi b_j (x_2 - x_1)}{q}.
\end{aligned} \tag{18}$$

Thus, from (18), the set of parameters $B = \{b_1, \ldots, b_s\} \subset \mathbb{Z}_q$ should give the minimal fidelity of all pairs of unequal $|\psi(x_1)\rangle$ and $|\psi(x_2)\rangle$, i.e.,

$$B = \underset{b_1,\ldots,b_s \in \mathbb{Z}_q}{\operatorname{argmin}} \max_{\substack{x_1, x_2 \in \mathbb{Z}_q \\ x_1 \neq x_2}} \frac{1}{2^s} \prod_{j=1}^{s} \cos^2\frac{\pi b_j (x_2 - x_1)}{q} = \underset{b_1,\ldots,b_s \in \mathbb{Z}_q}{\operatorname{argmin}} \max_{\substack{x \in \mathbb{Z}_q \\ x \neq 0}} \frac{1}{2^s} \prod_{j=1}^{s} \cos^2\frac{\pi b_j x}{q}. \tag{19}$$

In [81], we proved the existence of the parameter set $B = \{b_1, \dots, b_s\}$ described by Formula (19). The proof develops the ideas from [32] and uses a similar notion of the "good" parameters. The following lemma has been proven.

Lemma 1. *There is a set* $B = \{b_1, \dots, b_s\}$ *with* $|B| = s = \lceil 16 \ln q \rceil$*, which is "good" for all* $g \neq 0 \bmod q$.

In [82], the following connection between the single-qubit version and the "entangled" version was shown. Let $m = \log d$ be the number of qubits. Let $B = \{b_1, b_2, \dots, b_m\} \subseteq \mathbb{Z}_q$, and consider a special case where elements of $S = \{s_0, s_1, \dots, s_{(d-1)}\} \subseteq \mathbb{Z}_q$ are equal to sums of all possible subsets of B, i.e., for $j = j_1 j_2 \dots j_m$,

$$s_j = j_1 b_1 + j_2 b_2 + \dots + j_m b_m. \tag{20}$$

Then, the following holds:

$$\begin{gathered}
\left[\frac{1}{\sqrt{2}}(|0\rangle + e^{i\frac{2\pi b_1 x}{q}}|1\rangle)\right] \otimes \dots \otimes \left[\frac{1}{\sqrt{2}}(|0\rangle + e^{i\frac{2\pi b_m x}{q}}|1\rangle)\right] = \\
\frac{1}{\sqrt{2^m}}\left[\sum_{j_1=0}^{1} e^{i\frac{2\pi(j_1 * b_1)x}{q}}|j_1\rangle\right] \otimes \dots \otimes \left[\sum_{j_m=0}^{1} e^{i\frac{2\pi(j_m * b_m)x}{q}}|j_m\rangle\right] = \\
\frac{1}{\sqrt{2^m}}\sum_{j_1=0}^{1} \dots \sum_{j_m=0}^{1} e^{i\frac{2\pi(j_1 * b_1 + \dots + j_m * b_m)x}{q}}|j_1 \cdots j_m\rangle = \frac{1}{\sqrt{d}}\sum_{j=0}^{d-1} e^{i\frac{2\pi s_j x}{q}}|j\rangle.
\end{gathered} \tag{21}$$

Thus, the quantum hash function ψ_S can be computed by a circuit that has just two layers—the first layer contains Hadamard gates for all qubits, and the second layer contains single-qubit gates equivalent to R_z rotations. To do so, we need to choose the set of parameters $B = \{b_1, \dots, b_m\}$ that generates the ε-biased set S according to Equation (20).

3.7. Multiqudit Quantum Hashing

In [47], we defined another version of the quantum hashing technique for a cyclic group, i.e., we considered $\mathbb{X} = \mathbb{Z}_q$ and $|\mathbb{X}| = q$. It is based on small-biased sets and high-dimensional states (qudits). But first, we note the following equivalence between ε-biased sets.

Property 5. *Let* $S = \{s_1, \dots, s_d\}$ *and* $S' = \{0, (s_2 - s_1), \dots, (s_d - s_1)\}$*. Then, for any* $x \in \mathbb{Z}_q$ $bias(S, x) = bias(S', x)$*, i.e., the set* S *is equivalent (in terms of its bias) to* S'.

Proof. The proof of this statement is based on the following considerations:

$$\frac{1}{d}\left|\sum_{k=1}^{d} e^{2\pi s_k x/q}\right| = \frac{1}{d}\left|e^{2\pi s_1 x/q}\right|\left|\sum_{k=1}^{d} e^{2\pi(s_k - s_1)x/q}\right| = \frac{1}{d}\left|\sum_{k=1}^{d} e^{2\pi(s_k - s_1)x/q}\right|; \tag{22}$$

therefore,

$$bias(S, x) = \frac{1}{d}\left|\sum_{k=1}^{d} e^{2\pi s_k x/q}\right| = \frac{1}{d}\left|\sum_{k=1}^{d} e^{2\pi(s_k - s_1)x/q}\right| = bias(S', x). \tag{23}$$

□

Now, let $S_1, S_2, \dots, S_m \subset \mathbb{Z}_q$ be the ε-biased subsets of $\mathbb{Z}_q$, and we denote $S_j = \{s_{j,1}, \dots, s_{j,d}\}$ for $j = 1, \dots, m$. By Property 5, without loss of generality, we may consider all $s_{j,1}$ to be equal 0. In other words, for all $j = 1, \dots, m$ it holds that

$$\max_{x \neq 0} \frac{1}{d} \left| 1 + e^{i\frac{2\pi s_{j,2} x}{q}} + \ldots + e^{i\frac{2\pi s_{j,d} x}{q}} \right| \leq \varepsilon. \tag{24}$$

Then, for $x \in \mathbb{Z}_q$, we define a multiqudit quantum hash function in the following way:

$$|\psi_j(x)\rangle = \frac{1}{\sqrt{d}} (|\ell_1\rangle + e^{i2\pi s_{j,2} x/q} |\ell_2\rangle + \ldots + e^{i2\pi s_{j,d} x/q} |\ell_d\rangle), \tag{25}$$

$$|\psi(x)\rangle = |\psi_1(x)\rangle \otimes \cdots \otimes |\psi_m(x)\rangle, \tag{26}$$

where $|\ell_k\rangle$ are the basis states ($k = 1 \dots d$; d is the dimension of the qudit state space), q is the size of the input state space, $x \in \{0, 1, \dots, q-1\}$ is a classical input that is encoded by the relative phase of m qudit states, and $s_{i,k}$ are numeric parameters (elements of the ε-biased sets) of the quantum hash function that provide its collision resistance. The main idea of the collision resistance property is to provide minimum fidelity between different quantum hashes (quantum hash function images) with the minimum possible number of quantum information carriers. Furthermore, reaching a reasonable balance between collision resistance property and one-way property for a quantum hash function is also an important task.

Note that the formula for $|\psi(x)\rangle$ gives the classical–quantum function that transforms a classical input into the quantum state composed of m qudits (d-dimensional systems). The same state can be constructed with an appropriate number of 2-dimensional systems (qubits); however, this would imply creating entangled states, which are harder to create and maintain.

Theorem 1. *The classical–quantum function $\psi : \mathbb{Z}_q \to \mathcal{H}^{d^m}$ given by Equations (25) and (26) is a $\left(\frac{d^m}{q}, \varepsilon^m\right)$-resistant quantum hash function.*

Proof. According to Definition 3, we need to show two main properties for the function ψ:

1. δ-One-wayness: The dimension of the input space is q, and the quantum state space has dimension d^m. Thus, ψ is δ-one-way for

$$\delta = \frac{d^m}{q}. \tag{27}$$

2. ε-Collision resistance: The maximal inner product between unequal quantum hashes is bounded by

$$\begin{aligned}
\max_{x_1 \neq x_2} \left| \langle \psi(x_1) \,|\, \psi(x_2) \rangle \right| &= \max_{x_1 \neq x_2} \prod_{j=1}^{m} \left[\frac{1}{d} \left| 1 + e^{i\frac{2\pi s_{j,2}(x_2 - x_1)}{q}} + \ldots + e^{i\frac{2\pi s_{j,d}(x_2 - x_1)}{q}} \right| \right] \\
&= \max_{x \neq 0} \left| \langle \psi(x) \,|\, \psi(0) \rangle \right| \\
&= \max_{x \neq 0} \prod_{j=1}^{m} \left[\frac{1}{d} \left| 1 + e^{i\frac{2\pi s_{j,2} x}{q}} + \ldots + e^{i\frac{2\pi s_{j,d} x}{q}} \right| \right] \leq \varepsilon^m,
\end{aligned} \tag{28}$$

if all $S_j = \{s_{j,1}, \dots, s_{j,d}\}$ are ε-biased sets.

Thus, ψ corresponds to the balanced $\left(\frac{d^m}{q}, \varepsilon^m\right)$-resistant quantum hash function according to [25]. □

Note that the proof above also suggests that comparing hashes of two different values x_1 and x_2 is equivalent to comparing hashes of $x = (x_2 - x_1)$ and 0.

Remark 3. *Although ε-biased sets give guaranteed collision resistance to our multiqudit quantum hash function, for small sizes of input and output spaces, better bounds on collision resistance can be obtained by numeric optimization (see Table 1 for details).*

At the moment, this approach can be used only for relatively small values of d, m and q since it implies an exhaustive search for optimal values of $s_{j,k}$ that give minimum to the following function:

$$\min_{\{s_{j,k}\}} \max_{x \neq 0} \frac{1}{d^m} \prod_{j=1}^{m} \left| 1 + e^{i\frac{2\pi s_{j,2} x}{q}} + \ldots + e^{i\frac{2\pi s_{j,d} x}{q}} \right|. \tag{29}$$

Note that the general lower bound suggests that, for an input state space of size q, a collision probability bounded by ε^m implies that a target quantum state space should have a size of at least $\Omega((\log q)/\varepsilon^m)$. On the other hand, there exist ε-biased sets of size $d = O((\log q)/\varepsilon)$, and thus, our construction gives the output state space of $d^m = O((\log q)^m/\varepsilon^m)$.

Table 1. The worst-case values of collision probability with parameters from ε-biased sets and from the numeric optimization, $q = 256$.

Number of Qudits	ε-Biased Sets	Exhaustive Search
	$d = 2$	
1	0.9998	0.9998
2	0.9996	0.959
3	0.9994	0.7519
4	0.9992	0.4378
5	0.999	0.2031
6	0.9988	0.0806
7	0.9986	0.0279
	$d = 3$	
1	0.9681	0.9681
2	0.9372	0.5422
3	0.9073	0.1483
4	0.8784	0.0368
5	0.8504	0.0063
	$d = 4$	
1	0.8329	0.8329
2	0.6937	0.2174
3	0.5778	0.0429
4	0.4813	0.0072

4. Implementation of the Multiqudit Quantum Hashing on Real Devices

In the previous section, we suggested quantum hash functions as a sequence of independent qubits, where the classical information was encoded in the qubit phase. Now, we propose a quantum hashing protocol, where the information carriers are high-dimensional states with an orbital angular momentum. In this case, the structure of the quantum hash can be represented by Equations (25) and (26).

Here, we propose the implementation of a verification procedure whereby for a given quantum hash $|\psi(x_1)\rangle$ and a classical value x_2, it checks whether or not $x_1 = x_2$. The ideal quantum experiment that verifies a multiqubit quantum hash can be set as follows:

1. We receive a quantum hash of some generally unknown value x_1 as a sequence of m single photons in the overall state $|\psi(x_1)\rangle$:
$$|\psi(x_1)\rangle = |\psi_1(x_1)\rangle \otimes \ldots \otimes |\psi_m(x_1)\rangle, \tag{30}$$
where the j-th qudit is expected to be in the state $\left|\psi_j(x_1)\right\rangle$ as described by Equation (25).
2. Then, we check whether x_1 is equal to some predefined x_2 or not. To execute this, we perform measurements that project $\left|\psi_j(x_1)\right\rangle$ onto $\left|\psi_j(x_2)\right\rangle$ and $d-1$-phase orthogonal states:
$$\left|\psi_j^{\perp,g}(x_2)\right\rangle = \frac{1}{\sqrt{d}}(|\ell_1\rangle + e^{i\frac{2\pi s_{j,2}x_2}{q}+i\phi_{g,2}}|\ell_2\rangle + \ldots + e^{i\frac{2\pi s_{j,d}x_2}{q}+i\phi_{g,d})}|\ell_d\rangle), \tag{31}$$
where $\left|\langle\psi_j(x_2)\,|\,\psi_j^{\perp,g}(x_2)\rangle\right|^2 = 0$, $g = 1, \ldots, d-1$, and $\phi_{g,d}$ are additional phases responsible for the orthogonality of the states. For example, if we use qutrits (3-dimensional states) as information carriers, the set of $\{\phi_{g,d}\}$ is equal to $\{\{2\pi/3, -2\pi/3\}, \{-2\pi/3, 2\pi/3\}\}$.
3. The projection measurements of these states may be sequential or parallel. In the latter case, we have to prepare a complex phase mask on part 2 of SLM1, which is an appropriate superposition of detection masks for $\left|\psi_j(x_2)\right\rangle$ and $d-1$ states $\left|\psi_j^{\perp,g}(x_2)\right\rangle$. The complex mask directs the photons into d detection channels corresponding to the states $\left|\psi_j(x_2)\right\rangle, \left|\psi_j^{\perp,1}(x_2)\right\rangle, \ldots, \left|\psi_j^{\perp,d-1}(x_2)\right\rangle$, respectively. The single-photon detector click in $\left|\psi_j(x_2)\right\rangle$ or $\left|\psi_j^{\perp,g}(x_2)\right\rangle$ channels corresponds to the outcome $\left|\psi_j(x_1)\right\rangle = \left|\psi_j(x_2)\right\rangle$ or $\left|\psi_j(x_1)\right\rangle \neq \left|\psi_j(x_2)\right\rangle$, respectively.
4. If $x_1 = x_2$, the detector of the output $\left|\psi_j(x_2)\right\rangle$ would always click, while the other detectors would never click.
5. If $x_1 \neq x_2$, each of the detectors might click, but the probability of the erroneous outcome "$x_1 = x_2$" is bounded by the construction of the quantum hash function $|\psi(x)\rangle$.
6. If none of the detectors clicked, then the qudit is lost, and we either request for it to be sent again or tolerate the higher error probability.
7. If all of m measurements end up with the outcome "$\left|\psi_j(x_1)\right\rangle = \left|\psi_j(x_2)\right\rangle$", then the final result of the experiment is considered to be "$x_1 = x_2$". Otherwise, if at least one qudit leads to $\left|\psi_j(x_1)\right\rangle \neq \left|\psi_j(x_2)\right\rangle$, then the overall result is also "$x_1 \neq x_2$".

The error probability comes from the fidelity between two different quantum hashes, which is
$$\left|\langle\psi(x_1)\,|\,\psi(x_2)\rangle\right|^2 = \frac{1}{d^{2m}}\prod_{j=1}^{m}\left|1 + e^{i\frac{2\pi s_{j,2}(x_1-x_2)}{q}} + \ldots + e^{i\frac{2\pi s_{j,d}(x_1-x_2)}{q}}\right|^2. \tag{32}$$

The parameter set $\{s_{j,k}\}$ is chosen in such a way that the pairs of hashes $|\psi(x_1)\rangle$ and $|\psi(x_2)\rangle$ give the minimal fidelity for $x_1 \neq x_2$. To measure the collision probability, we compare different quantum hashes in the worst-case scenario when quantum hashes for x_1 and x_2 have the maximum fidelity for a given set $\{s_{j,k}\}$.

The protocol starts with a calibration step on which we adjust the coincidence count rate for the "yes" answer ("$x_1 = x_2$"). We perform about 200 projection measurements of equal states and calculate the average coincidence count rate between signal and idler photons. The simultaneous clicks in the idler and signal detectors mean that the idler photon has been prepared in the state $|\psi(x_1)\rangle$ and has been successfully projected on the state $|\psi(x_2)\rangle$. We pick the average value as the threshold between "yes" and "no".

In this work, we experimentally evaluate the collision probability for a multiqudit quantum hash function with $q = 256$ and three groups of parameters: (i) $d = 2$ and $m = 1, \ldots, 7$, (ii) $d = 3$ and $m = 1, \ldots, 5$, and (iii) $d = 4$ and $m = 1, \ldots, 4$, i.e., we perform experiments with varying numbers of qubits, qutrits, and ququarts. We encode the classical message x_1 into the phase of states considered in Section 3.7. Since we consider the worst-case situation, without loss of generality, we can focus on the case when $x_2 = 0$. This can be calculated from

$$x_1 = \arg\max_{x \neq 0} \frac{1}{d^{2m}} \prod_{j=1}^{m} \left| 1 + e^{i\frac{2\pi s_{j,2} x}{q}} + \ldots + e^{i\frac{2\pi s_{j,d} x}{q}} \right|^2 \tag{33}$$

for an optimal (quasioptimal) set of parameters $\left\{ s_{j,k} \right\}$, which in turn is precomputed as

$$\left\{ s_{j,k} \right\} = \arg\min_{\left\{ s_{j,k} \right\}} \max_{x \neq 0} \frac{1}{d^{2m}} \prod_{j=1}^{m} \left| 1 + e^{i\frac{2\pi s_{j,2} x}{q}} + \ldots + e^{i\frac{2\pi s_{j,d} x}{q}} \right|^2 . \tag{34}$$

Figure 1 shows the comparison of experimental and theoretical error rates for the worst-case scenario depending on different dimensions of quantum states. The experimental results suggest that the proposed technique can be useful even for small sizes of input and output states. Moreover, it can be seen that the number of information carriers decreases with an increase in their quantum state dimension for an optimal relation between the collision probability and the decoding probability (the probability of extracting the classical input x from the quantum hash). For instance, if we limit the collision probability by 0.25 and the decoding probability by 0.15, the optimal number of qudits to "compress" an 8-bit classical information proves to be $m = 5$ for $d = 2$, $m = 3$ for $d = 3$, and $m = 2$ for $d = 4$. Moreover, according to the Holevo theorem [73], no more information can be extracted from a quantum d-level system than from a classical d-level system, which means that we have a bounded probability (below 1) to extract the information about classical input x.

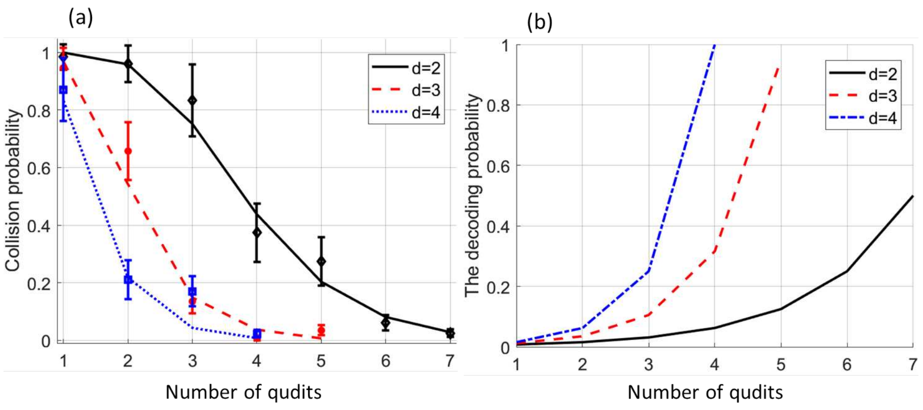

Figure 1. (**a**) Comparing experimental and theoretical collision probabilities for the worst-case scenario for different dimensions of quantum states. Vertical lines demonstrate experimental results: the average value of the error in the series of measurements and its standard deviation. (**b**) The theoretical probability of recovering the original message from its quantum hash.

5. Searching Coefficients

The main properties of the quantum hash function are defined by the set of coefficients S; therefore, it is important to explore methods to find the sets of coefficients that lead to quantum hash functions with a small probability ε of error. Although there are explicit methods of construction, these are usually slow or their results are worse than those of heuristic methods. This section covers several methods of search, such as brute-force algorithms, random search, genetic algorithms, simulated annealing, and deterministic methods.

Let us define the problem formally. Consider $\varepsilon_0 > 0$ and $q \in \mathbb{N}$. First, we choose the size $d = d(\varepsilon_0, q)$ of the parameter set $S = \{s_1, \ldots, s_t\}$, for example, using Property 2 or Remark 1. Second, we search for the parameter set S such that $\varepsilon = \varepsilon(S) \leq \varepsilon_0$, where

$$\varepsilon(S) = \frac{1}{d}\left|\sum_{j=1}^{t} \cos\frac{2\pi s_j}{q}\right|, \tag{35}$$

or is defined in a similar way for the other variants of quantum hash functions. We compare different methods in terms of their running time expressed as a function of q, d, and ε_0. Note that we include the time of computing $\varepsilon(S)$ into the whole running time of the algorithm.

In the brute-force method, we systematically check all possible assignments of $s_j \in \{0, \ldots, q-1\}$, $j = 1, \ldots, d$. For example, we can start from the assignment $s_1 = 0, s_2 = 0, \ldots, s_d = 0$, and then move on to the next assignment by increasing the rightmost coefficient s_k that is not equal to $q-1$ along with setting all $s_j = 0$ for $j > k$. The running time of the brute-force method is $O(dq^d)$.

The random search method takes advantage of Remark 1, and repeatedly chooses a random set S until it satisfies the condition $\varepsilon(S) \leq \varepsilon_0$. While this process could be infinite, Remark 1 ensures that for properly chosen d, the expected running time of the algorithm is $O(dq)$.

Heuristic algorithms, such as genetic algorithms or simulated annealing, reduce the amount of search space, but may not find the best solution. For example, genetic algorithms use analogies from the evolutionary process. First, they generate the pool of "organisms" (several random sets S); then, they select those that are best suited (i.e., those with smaller $\varepsilon(S)$) and "breed" them (i.e., for two sets $S^{(1)} = \{s_1^{(1)}, \ldots, s_d^{(1)}\}$ and $S^{(2)} = \{s_1^{(2)}, \ldots, s_d^{(2)}\}$, they generate the set $S = \{s_1^{(1)}, \ldots, s_k^{(1)}, s_{k+1}^{(2)}, \ldots, s_d^{(2)}\}$ for some random $k = 1, \ldots, t-1$).

Simulated annealing, another heuristic algorithm, likens the search to the heat treatment of steel. It starts with a set $S = \{s_1, \ldots, s_d\}$, and it sets a parameter T (a "temperature") to a high value, and then starts to slowly decrease it. At each iteration, the probability of changing a random coefficient s_k to another value depends on temperature and $\varepsilon(S)$, with higher temperature permitting larger changes. The heuristic algorithms are presented in [38].

Deterministic methods are usually based on derandomization techniques. For example, Khadieva and Ziiatdinov [68] implemented the algorithm described in [83] to compute the parameter set and performed numerical experiments.

6. Circuit Representation of Quantum Hashing and Implementation on Noisy Simulators

6.1. Full Quantum Circuit for Quantum Hashing

The basic implementation part of quantum hashing circuits is a uniformly controlled rotation operator (see Figure 2). We denote the uniformly controlled rotation operator for n control qubits and one target qubit as the UCR_d operator.

Here, R_a^j is the rotation to a $\frac{2\pi s_j}{q}$ angle by the a axis, that is, z or y. It depends on the type of hashing function: y for the amplitude form, and z for the phase form.

Many types of quantum computers (for example, quantum devices based on superconductors) do not allow us to apply two-qubit gates to an arbitrary pair of qubits but have a graph that represents such a restriction. Vertices of the graph correspond to qubits, and two-qubit gates can be applied only to qubits corresponding to vertices connected by an edge. We say that such a graph represents the qubit connection graph for a device.

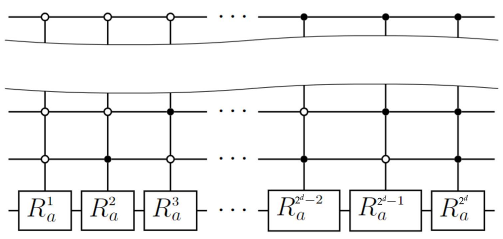

Figure 2. Quantum circuit for the uniformly controlled rotation UCR_d operator with d control qubits and 2^d rotations by the a axis, that is, z or y.

If the qubit connection graph is complete, i.e., we do not have any restriction on the two-qubit gate application, then the CNOT cost of the uniformly controlled rotation operator is as presented in the following lemma.

Theorem 2 ([84,85])**.** *The CNOT cost of the d-qubit uniformly controlled rotation operator* UCR_d *is* 2^d.

One of the most popular architectures considered by researchers is the linear nearest neighbor (LNN) architecture. The graph for LNN architecture is a chain where the i-th vertex is connected only with vertices $i-1$ and $i+1$ (see Figure 3).

Figure 3. The linear nearest neighbor (LNN) architecture for 5 qubits.

A circuit for the LNN architecture was presented by Möttönen, Vartiainen, Bergholm, and Salomaa in [84,85] (see Figure 4).

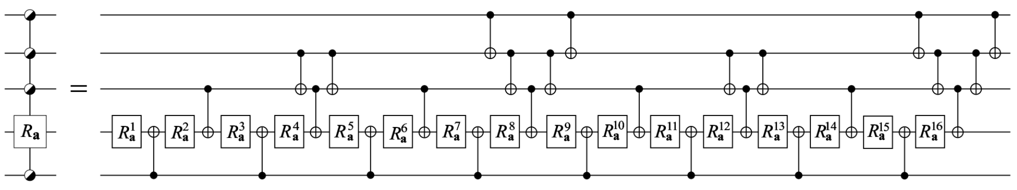

Figure 4. The quantum circuit for the uniformly controlled rotation gate with 4 control qubits for a linear nearest neighbor (LNN) architecture.

The restriction on architecture slightly affects the CNOT cost, which is discussed in the next theorem.

Theorem 3 ([84,85])**.** *The CNOT cost of the d-qubit uniformly controlled rotation operator* UCR_d *for the LNN architecture is* $\frac{10}{6}2^d + 3d - 9$.

Zinnatullin, Khadiev, and Khadieva optimized the previous circuit for more complex architectures [50]. A 5-qubit IBM Falcon r4T (Figure 5), 7-qubit IBM Falcon r5.11H (Figure 6), 16-qubit IBM Falcon r4P (Figure 7), and 27-qubit Falcon r5.11(Figure 8) were considered.

The CNOT cost for optimized circuits are 20 for a 5-qubit machine, 92 for a 7-qubit machine, 60,716 for a 16-qubit machine, and 100,909,180 for a 27-qubit machine.

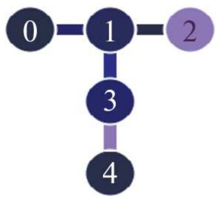

Figure 5. A 5-qubit IBM Falcon r4T architecture.

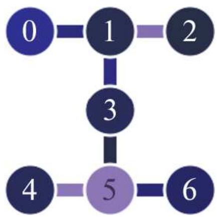

Figure 6. A 7-qubit IBM Falcon r5.11H architecture.

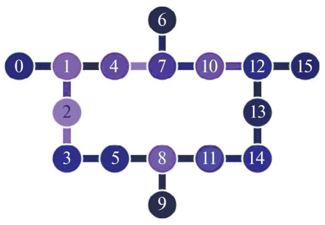

Figure 7. A 16-qubit IBM Falcon r4P architecture.

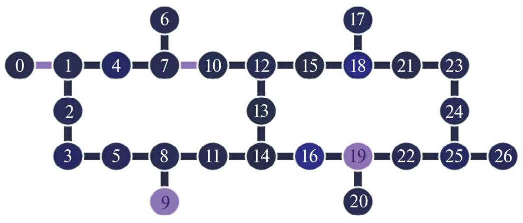

Figure 8. A 27-qubit IBM Falcon r5.11 architecture.

An application of the presented technique for the representation of the UCR_d operator as a quantum circuit requires a better precision for the rotation angles than the original one. The original angles require the $O(\log q)$ bit precision (precisely, it is $\frac{2\pi}{q}$). At the same time, the presented circuits use angles with $O(\log q \log \log q)$ bit precision (precisely, it is $\frac{2\pi}{q \log_2 q}$). Note that preposition of a rotation angle is a challenging issue for hardware [86]. Khadieva, Salehi, and Yakaryılmaz in [53] proposed a technique that provides a trade-off between CNOT cost and precision for angles. It allows us to obtain any intermediate result between 2^d CNOT cost and $\frac{2\pi}{q \log_2 q}$ precision, and $d2^d$ CNOT cost and $\frac{2\pi}{q}$ precision.

Each CNOT gate gives us an error probability. Even if the error is very small, the exponential number of CNOT gates dramatically affects the error probability. That is why researchers pay attention to shallow versions of circuits for quantum hashing.

6.2. Shallow Circuit

Kālis (with Ambainis as a supervisor) in his master thesis [51] suggested a shallow circuit for the quantum hashing in amplitude form for 3 qubits. This idea was investigated by Salehi and Yakaryılmaz [87]. Later, the approach was developed in detail and presented in a general way for quantum hashing by Ziatdinov, Khadieva, and Yakaryılmaz [52]. They suggested to use a shallow circuit presented in Figure 9.

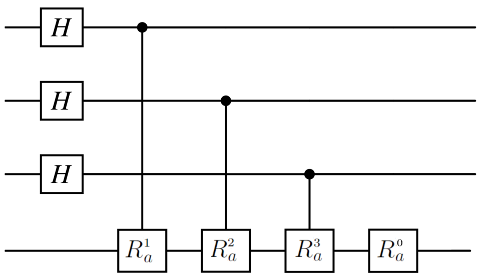

Figure 9. Shallow quantum circuit for quantum hashing algorithm with 3 control qubits.

If we use the same number of qubits $d = \frac{2}{\varepsilon}\log_2 q$ as for the standard circuit with UCR_d operator, then only $d = O(\log q)$ independent angles can be chosen. For the angles $\beta_0, \ldots, \beta_d$ used in the shallow circuit, we obtain the angles $\alpha_i = \sum_{j=0}^{\log_2 i} bin_i[j] \cdot \beta_j$, where $bin_i[j]$ is the j-th bit in the binary representation of the integer i. According to [52], it is possible to use generalized arithmetic progressions to find the angles β_i such that the error probability of the quantum hash is comparable to the full quantum circuit, but that requires a large circuit width. The method of finding the angles β_i such that both error probability and circuit width are small remains an open problem.

At the same time, numerical experiments show that we can find a set of angles $\beta_0, \ldots, \beta_d$ for $d = \frac{2}{\varepsilon}\log_2 q$ such that the error probability of the method is at most 2ε. This research is presented in [52].

Ziatdinov, Khadieva, and Khadiev in [54] invoked both circuits on a noisy simulator of an IBM machine for four qubits (three control and one target) for computing $MOD_{17} = \{1^x : x \bmod 17 = 0\}$ unary language, where 1^x means a string of x symbols 1. By recognizing, we mean returning 1 if the input belongs to MOD_{17}, and 0 otherwise. Note that a classical algorithm requires at least $\lceil \log_2 17 \rceil = 5$ bits of memory for recognizing the language [32].

For solving the problem, the authors did not search for angles that minimize the error in an "ideal" (not noisy) device. They used another strategy; they invoked a brute-force algorithm to find the sequence $\beta_0, \ldots, \beta_n$ with two properties:

1. The probability of accepting member inputs on the noisy simulator should be as high as possible.
2. The probability of accepting non-member inputs on the noisy simulator should be as small as possible.

For $q = 17$, the authors minimized a value $diff = P(6 \cdot q) - max\{P(r) : r \mod q \neq 0, 1 \leq r \leq 6 \cdot q + 9\}$, where $P(i)$ is the probability of returning 1 for an input 1^i. Then, the circuit was invoked for 1^i, where $i \leq 7 \cdot q + 9 = 128$. The program was executed 10,000 times and the number of shots that returned the accepting state was the result of computation. We use the notation $\tilde{P}(i)$ for this number, where i is the length of the word. The numbers $\tilde{P}(i)$ for each length i for $1 \leq i \leq 128$ are shown in Figure 10. We can say that it is a statistical representation of probability $P(i)$.

We can see that each member i of MOD_{17} has $\tilde{P}(i) \geq 1021$. At the same time, each non-member i has $\tilde{P}(i) \leq 925$. So, we can choose a threshold $\tilde{\lambda} = 1000$. For any length i, we can say that it is a member iff $\tilde{P}(i) > \tilde{\lambda}$. Here, we can use the concept of an isolated cutpoint from automata theory [88]. For a cutpoint $\lambda = 0.1$ and constant $\varepsilon = 0.001$, we claim that an input 1^i is a member of the language if $\tilde{P}(i) > \lambda + \varepsilon$. If an input 1^i is a non-member, then $\tilde{P}(i) < \lambda - \varepsilon$.

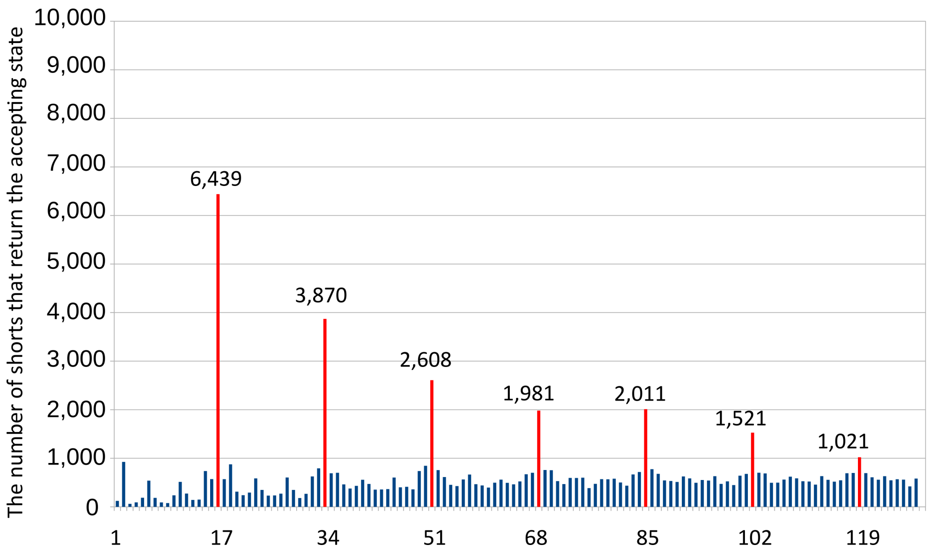

Figure 10. The number of shots that return the accepting state (Y axis) in the case of a shallow circuit. For different lengths i of the input word, the length of the word (X axis) is $1 \le i \le 128$.

The authors invoked the same series of experiments for a standard circuit, that is, UCR_d for $d = 3$. The number of parameters (angles) was too large (it is $2^3 = 8$) for the brute-force search because the invocation of the circuit on an IBMQ simulator is time-consuming. That is why the coordinate descent method was used for the first six parameters and brute force for the last two parameters. Then, we invoked the circuit for the same inputs 1^i for $i \le 7 \cdot p + 9 = 128$. The result is presented in Figure 11.

We can see that only a few members like $i = q$, $i = 2q$, $i = 4q$, and $i = 7q$ can be separated from non-members. Other members of MOD_q cannot be separated from members, for example, $i = 3p$, $i = 5p$, and $i = 6q$. In addition, the result for 1^1 is larger than for 1^q, where $q = 17$. The searching algorithm cannot find a set S that violates this condition. At the same time, we can ignore $i = 1$ because it can be checked with additional conditions. Even in this case, the algorithm is only useful for detecting some members with a threshold (or cutpoint) of $\lambda = 1000$, but we cannot detect other members.

At the same time, in the case of the shallow circuit, $\tilde{P}(q)$ is much higher than for the standard circuit. For instance, $\tilde{P}(q) > 5100$ for the shallow circuit, which corresponds to provability $P(q) > 0.51$, while for the standard circuit, the corresponding probability would be much smaller than 0.5 (approximately 0.1).

These numerical experiments show us that even if for an "ideal" device, the standard circuit with the UCR_d operator has a smaller error probability than the shallow circuit. In contrast, we have the opposite situation for noisy simulators, where the shallow circuit shows much better results than the standard one.

Vasiliev in [82] considered the phase version of quantum hashing. In this work, R_z rotation was used. The author used the equivalence of R_z and P gates, which are presented in Figure 12.

We can use the fact that the target qubit is initially in the $|1\rangle$ state and remove one of two P gates; the remaining P gate can be applied without control. See Figure 13 for the final equivalence of gates.

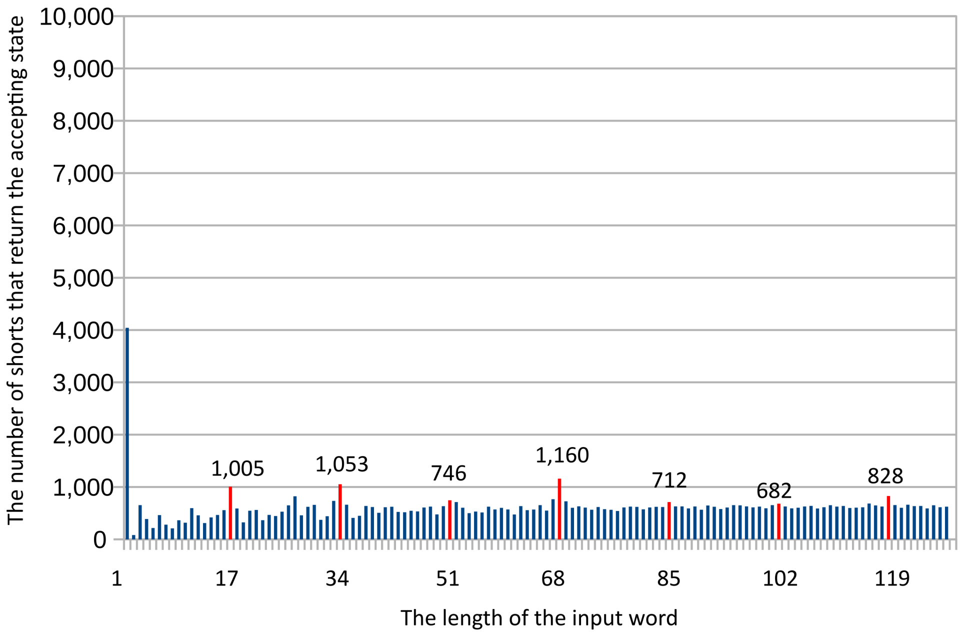

Figure 11. The number of shots that return the accepting state (Y axis) in the case of a standard circuit. For different lengths i of the input word, the length of the word (X axis) is $1 \leq i \leq 128$.

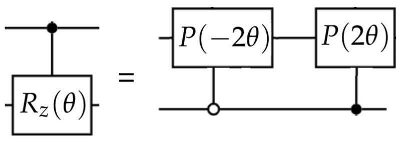

Figure 12. Equivalence of control R_z and control P gates.

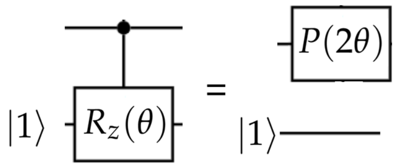

Figure 13. Equivalence of control R_z with $|1\rangle$ value for the control qubit and the P gate.

This allows us to convert the shallow circuit for quantum hashing to the circuit without any CNOT gates (see Figure 14).

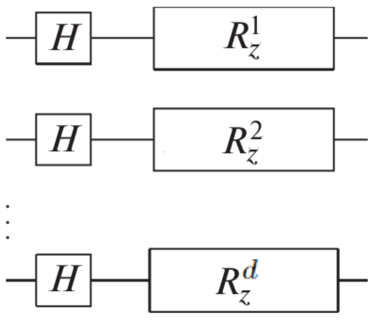

Figure 14. Shallow circuit for quantum hashing algorithm in phase form in the case of d control qubits.

6.3. Shallow Circuit for LNN Architecture and for Arbitrary Qubit Connection Graphs

Khadieva, Salehi, and Yakaryılmaz [53] suggested a quantum circuit for an LNN architecture (see Figure 15).

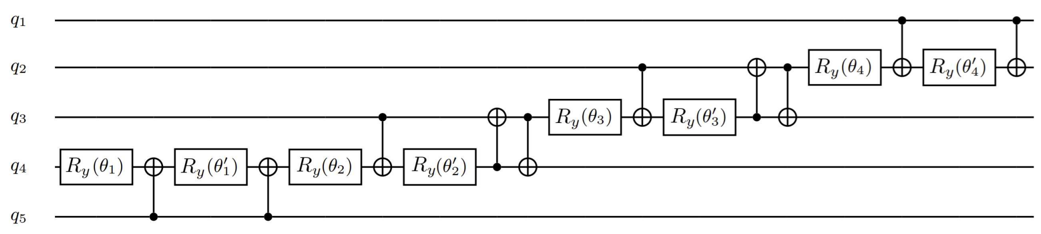

Figure 15. The shallow quantum circuit for quantum hashing with 4 control qubits for linear nearest neighbor (LNN) architecture.

The main idea is to move the target qubit by the chain using the SWAP gate. The control R_y gate has 2 CNOT cost, the SWAP gate has 3 CNOT cost, but control R_y and SWAP together also have 3 CNOT cost (see Figure 16).

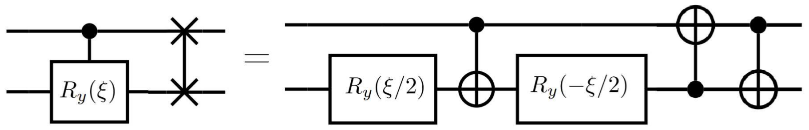

Figure 16. Representation of sequential control R_y and SWAP gates using only basis gates.

So, this idea allowed the authors to reduce the CNOT cost and obtain the CNOT cost of the circuit that applies quantum hashing ℓ times for d qubits, as presented in the next lemma.

Theorem 4 ([53])**.** *In the case of the LNN architecture, the CNOT cost of a shallow quantum circuit for ℓ applications of quantum hashing with d control qubits is $(3d-7)\ell+2$.*

Later, Khadieva [45] generalized the technique for more a complex architecture that involves a cycle with tails (like a "sun" and "two joint suns") represented by a 16-qubit Falcon r4P (see Figure 7) and a 27-qubit Falcon r5.11 IBMQ (see Figure 7) architectures. The CNOT cost of the shallow circuit for the quantum hashing algorithm for these devices is presented in the next lemma.

Theorem 5 ([45])**.** *The CNOT cost of the shallow quantum circuit for ℓ applications of quantum hashing with 15 control qubits is $37\ell+4$ (for $\ell \geq 3$) in the case of the 16-qubit Falcon r4P IBMQ device. The CNOT cost for 26 control qubits is $67\ell+4$ (for $\ell \geq 3$) in the case of the 27-qubit Falcon r4P IBMQ device.*

The method was generalized for arbitrary qubit connection graphs by Khadiev, Khadieva, Chen, and Wu in [55]. The authors built the shortest path that visits all vertices at least once. Then, they moved the target qubit by this path using the same technique as for an LNN architecture. They obtained the following complexity for ℓ applications of quantum hashing on d qubits:

Theorem 6 ([55])**.** *Let G be a connected qubit connection graph. The CNOT cost of the circuit for the application of the quantum fingerprinting (quantum hashing) algorithm ℓ times is at most $(3k-4)\ell+2$, where k is the length of the shortest path in the graph G that visits all vertices at least once.*

Note that the CNOT cost is between $(3d-4)\ell+2$ and $(\frac{3}{2}d^2+\frac{3}{2}d-4)\ell+2$ depending on k. If the graph G has a Hamiltonian path, then the cost is minimal.

The shallow quantum circuit for quantum hashing has a structure very similar to the circuit for the quantum Fourier transform (QFT) algorithm [89]. The technique is used in quantum addition [90], quantum phase estimation (QPE) [89,91], quantum amplitude estimation (QAE)[92], the algorithm for solving linear systems of equations [93], Shor's factoring algorithm [94], 1D wave equation [95], Quantum walks [96–98] with applications to quantum backtracking [99,100], and others. The best known representation of the QFT algorithm for LNN architectures is [101], which is an improvement of a series of results [102–106]. At the same time, the presented technique was not applied to general graphs. The authors of [45] and [55] applied the ideas used for quantum hashing to the QFT algorithm and obtained the CNOT cost for arbitrary qubit connection graphs. Another interesting technique for several specific graphs was discussed in [107].

7. Applications of Quantum Hashing Technique

The quantum hashing technique has been widely used in various areas. Here, we mention only some of the results.

- Stream processing algorithms. Le Gal [56] considered an automata-like model with non-constant size of memory. The technique allowed him to obtain an advantage in memory size for a quantum version of the model. The technique was used for checking the equality of two strings using a logarithmic size of memory.
- Automata. The technique was introduced for an automata model [32] and later improved in [33,34]. The technique allows us to recognize a unary language $MOD_p = \{a^i : i \bmod p = 0\}$ for some prime p. It allows the authors to demonstrate an example of language that can be recognized by a quantum model with an exponentially smaller size of memory compared to the classical (deterministic or probabilistic) counterparts. At the same time, the same technique for the constant number of qubits was used for two-way automata with classical and quantum states [108–111]. It allows authors to show a language that can be recognizable by the model but cannot be recognizable by the probabilistic two-way automata. Similar results were obtained for one-way models [112,113].
  Later, the same idea was used for one-way automata and promise problems [66,67,114–117]. The technique allows showing more languages that can be recognized by the quantum model but cannot be recognized by a classical model [68,118].
- Branching programs. The technique was used for the branching program model by Ablayev, Vasiliev, Gainutdinova, and co-authors [35–37]. They showed a family of Boolean functions that can be computed by quantum branching programs with polynomial width (logarithmic size of memory) but cannot be computed by deterministic and probabilistic counterparts. Most of the Boolean functions have an equality of objects (binary strings or other objects) as a base.
  Later, Khadiev and Khadieva [63] presented an especially constructed Boolean function that allowed them to show a hierarchy of complexity classes for quantum read-k-time branching programs. The upper bound was proven using the quantum fingerprinting technique.
  Nondeterministic quantum branching programs were investigated by Gainutdinova and co-authors [64,66,119–121]. The authors suggested several Boolean functions such that nondeterministic quantum branching programs based on specific versions of quantum hashing can compute them, but classical counterparts cannot.
- Online algorithms with restricted memory size. Quantum algorithms with restricted memory size is a computational model similar to automata but used for online minimization problems [61,62]. Khadiev and Khadieva [59,60] presented a problem

that can be solved by quantum online algorithms with a restricted size of memory, but cannot be solved by randomized or deterministic counterparts in the case of logarithmic size of memory. The technique based on one qubit allowed the authors to show a similar result for a constant size of memory [122,123]. The algorithms also used the quantum hashing algorithm for checking the equality of binary strings using a logarithmic size of memory.
Similar approaches were used in [124].

- Development of quantum devices.
  Vasiliev [65] used the technique for developing communication protocols between parts of quantum devices.
- Query model algorithms. Quantum hashing has numerous applications in addressing the string matching problem. The circuit implementation is discussed in [57,58] and the algorithm is presented in detail in the next section.

7.1. Quantum Search in a Dictionary and String Matching Problem

The hashing technique for searches in a database is widely used for compressing data, which allows us to accelerate search procedures. We demonstrate compressing effects and possibilities of the quantum hashing technique based on searches for words in a text.

In this section, we consider the string matching problem. The problem is to search for a word w in a text V. Formally, we should find any index i such that the substring $V_i, \ldots, V_{i+m-1} = w$, where V_i is the i-th symbol of V, m is the length of w, and n is the length of V. Here, $1 \leq i \leq n$.

The task of searching for information in a database has been considered in many works and has some specific features. First, the oracle f_w, defined by the task, is used to find the occurrence of the word w in the dictionary V, namely $f_w(x) = 1$ if and only if $w = x$. Secondly, to speed up the search for a word in a text in the 1970s and 1980s, including hashing-based algorithms were proposed. The Knuth–Morris–Pratt (KMP) algorithm [9,125,126], the Rabin–Karp algorithm [7,8], and the Matiyasevich algorithm [127] can solve the problem in a linear number of comparisons (in linear time), with $O(n+m)$ complexity. Other possible solutions to the problem are Backward Oracle Matching [128], the Wu–Manber algorithm [129], the Boyer–Moore algorithm [130], the algorithms in [131,132] based on suffix arrays [8,131,133,134] and suffix trees [135,136], and others [128,137–139].

It is clear that quantum algorithms for this task, based on Grover's idea [70,92], provide a quadratic saving in the number of calls to the oracle in the quantum query model. Note that to describe Grover's algorithm, we first need to specify an efficient oracle circuit for f_w for such an algorithm. Several similar studies in recent decades in the field of developing algorithms for the string matching problem have been devoted to the construction of efficient oracles f_w. A number of constructions of such oracles in terms of quantum circuits have been proposed, and the complexity of such circuits have been studied. These include the Practical implementation of a Quantum String Matching Algorithm [140] and the algorithms of Ramesh, Hariharan, and Vinay [141], Montanaro [142], Soni and Rasool [143], and Khadiev and Serov [144]. Note that all of these listed results require $O(\log n + m)$ qubits.

7.2. Hashing Technique for Quantum Search in the Text

The first idea of using hashing methods for quantum information retrieval in an unordered database was realized in [58]. In this paper, a hybrid classical–quantum (probabilistic-quantum) algorithm is proposed. The key idea of the hybrid classical–quantum algorithm is as follows: (a) choose a universal hash family $\mathcal{F}$ of hash functions, (b) uniformly randomly choose a hash function $h \in \mathcal{F}$, (c) hash the elements of V with the hash function h, and (d) apply the quantum amplification technique to search for the hash image $h(w)$ of the word w in the hashed dictionary $h(V)$. It has been proven that using hashing methods can exponentially reduce the number of qubits required. Specifically, $\log n + \log m$ qubits are sufficient when using hashing, instead of $\log n + m$ qubits in algorithms without hashing.

7.3. Search in a Dictionary Based on the Quantum Hashing Technique

The next step in developing a quantum search in a dictionary is based on quantum fingerprinting–hashing and it is presented in [145].

We begin with a discussion of quantum fingerprinting–hashing features that we explored for search in a dictionary. It is important for the above construction that the size of the quantum hash $s = O(\log m)$, that is, we have an exponential compression of information.

The quantum hash function ψ defines for the word $w \in \Sigma^m$ a unitary transformation:

$$U_\psi : (\mathcal{H}^2)^{\otimes s} \xrightarrow{w} (\mathcal{H}^2)^{\otimes s}. \tag{36}$$

That is, the operator U_ψ acts on the space $(\mathcal{H}^2)^{\otimes s}$, having $w \in \Sigma^m$ as an external control word. It is also convenient to think of this as ψ defining a set $\{U_\psi(w) : w \in \Sigma^m\}$ of unitary transformations:

$$U_\psi(w) : (\mathcal{H}^2)^{\otimes s} \to (\mathcal{H}^2)^{\otimes s}. \tag{37}$$

In this paper, we assume that the unitary transformation $U_\psi(w)$ always starts from an initial basis state $|0\rangle$, that is, we have

$$U_\psi(w) : |0\rangle \mapsto |\psi(w)\rangle \quad \text{or} \quad U_\psi(w)|0\rangle = |\psi(w)\rangle. \tag{38}$$

For further consideration, we also need the inverse transformation U_ψ^{-1} of U_ψ. Thus, accordingly, for the inverse transformation U_ψ^{-1}, the following must be true:

$$U_\psi^{-1}(w) : |\psi(w)\rangle \mapsto |0\rangle \quad \text{or} \quad U_\psi^{-1}(w)|\psi(w)\rangle = |0\rangle. \tag{39}$$

Remark 4. *So, the above definition can be seen as a quantum generalization of the information compression function. On the other hand, we have a quantum hash function ψ and its implementation, the operator U_ψ. Since the unitary operator U_ψ is invertible, this means that there are no collisions "in the quantum world". That is, two different words w and w' have different hash images $|\psi(w)\rangle$ and $|\psi(w')\rangle$. But we can obtain a collision "in the macro world" (that is, we can decide that the state $|\psi(w)\rangle$ is generated by the word w') when extracting information about these two different quantum states $|\psi(w)\rangle$ and $|\psi(w')\rangle$. Thus, the probability of such an event (the collision probability) is small when ψ is the collision ϵ-resistant function (see Definition 2).*

Now, we present the algorithm based on the technique of (m, ϵ, s)-quantum hash function works for a quantum solution of the search in a dictionary problem. The sequence $V = \{w_0, \dots, w_{n-1}\}$ of binary words with length m and a binary word w with length m are given. We want to find any index k, which denotes the word number k such that $w = w_k$.

Note that the string matching problem can be easily represented as the search in a dictionary problem by taking w_i as a substring of the text string with index i and finishing in $i + m - 1$.

Thus, the algorithm $\mathcal{A}$ implements the mapping

$$\mathcal{A} : V, w \longmapsto k. \tag{40}$$

The first part of the algorithmconsists of preparing the initial state.

According to this sequence V, using the transformation U_ψ, which defines the (m,ϵ,s)-quantum hash function ψ, a state is prepared:

$$|V,\psi\rangle = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} |j\rangle \otimes |\psi(w_j)\rangle \otimes |1\rangle, \tag{41}$$

The second part of the algorithm $\mathcal{A}$—amplification—consists of reading a word to be searched for, w, and finding a number k such that $w = w_k$. The steps are as follows:

1. **Conversion hash transformation—first level of amplification**
The operator $I^{\otimes \log n} \otimes U_\psi^{-1}(w) \otimes I$ is applied to the state $|V,\psi\rangle$,
where $U_\psi^{-1}(w)$ is the inverse transformation controlled by the searched word w. We obtain the state

$$|V,\psi,w\rangle = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} |j\rangle \otimes |\phi(w_j,w)\rangle \otimes |1\rangle, \tag{42}$$

where $|\phi(w_j,w)\rangle = U^{-1}(w)|\psi(w_j)\rangle$ for $j \in \{0,\ldots,n-1\}$.
2. **Amplification**
The amplification of basic states $|j\rangle|0\rangle^{\otimes s}|1\rangle$ is when "good states" are applied to state $|V,\psi,w\rangle$. The procedure is described in [92], and is also discussed in the book [146].
3. **Measurement**
The first $\log n + s$ qubits of the final state are measured in a computational basis. If the last s qubits are all zero, then the measurement result k of the first $\log n$ qubits is declared as the index of the word w_k in the sequence V, for which $w_k = w$.

The algorithm has the following complexity characteristics: searching for a word w of length m in a dictionary V of size n requires $O(\log n + s)$ qubits. The number of queries (the number of Grover operators applied) is $O(\sqrt{n})$. The key point of the algorithm is the use of the (m,ϵ,s)-fingerprinting–hash function to obtain the correct result with high probability.

The steps of the algorithm are as follows: In the first step, the quantum state is quantum-parallel inverted by the mapping $\psi^{-1}(w)$ defined by w. This step provides the first level of amplitude gain. Then, in the second step, the known sequence of Grover "amplitude amplification operators" is applied. Finally, the resulting quantum state is measured and the algorithm produces the result.

The high probability of a correct result is based on the properties of the quantum hash function—it provides the first level of amplitude amplification before applying the sequence of Grover amplitude amplification operators.

For the formal proof, we refer to [145].

Funding: The research in Section 6 is supported by the Russian Science Foundation Grant 24-21-00406 (https://rscf.ru/en/project/24-21-00406/). M.Z. acknowledges financial support under the National Recovery and Resilience Plan (PNRR), Mission 4, Component 2, Investment 1.4, Call for tender No. 1031 published on 17/06/2022 by the Italian Ministry of University and Research (MUR), funded by the European Union—NextGenerationEU, Project Title "National Centre for HPC, Big Data and Quantum Computing (HPC)"—Code National Center CN00000013—CUP D43C22001240001.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Nielsen, M.A.; Chuang, I.L. *Quantum Computation and Quantum Information*, 1st ed.; Cambridge University Press: Cambridge, UK, 2000. [CrossRef]
2. Ambainis, A. Understanding Quantum Algorithms via Query Complexity. In Proceedings of International Congress of Mathematicians'2018, Rio de Janeiro, Brazil, 1–9 August 2018; Volume 4, pp. 3283–3304.

3. Ablayev, F.; Ablayev, M.; Huang, J.Z.; Khadiev, K.; Salikhova, N.; Wu, D. On quantum methods for machine learning problems part I: Quantum tools. *Big Data Min. Anal.* **2019**, *3*, 41–55. [CrossRef]
4. Jordan, S. Quantum Algorithms Zoo, 2025. Available online: http://quantumalgorithmzoo.org/ (accessed on 1 April 2025).
5. Konheim, A.G. *Hashing in Computer Science: Fifty Years of Slicing and Dicing*; John Wiley & Sons: Hoboken, NJ, USA, 2010.
6. Mehta, D.P.; Sahni, S. *Handbook of Data Structures and Applications*; Chapman and Hall/CRC: Boca Raton, FL, USA, 2004.
7. Karp, R.M.; Rabin, M.O. Efficient randomized pattern-matching algorithms. *IBM J. Res. Dev.* **1987**, *31*, 249–260. [CrossRef]
8. Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C. *Introduction to Algorithms*; McGraw-Hill: New York, NY, USA, 2001.
9. Knuth, D. *Searching and Sorting, the Art of Computer Programming*; Reading: Addison-Wesley: Boston, MA, USA, 1973; Volume 3.
10. Mehlhorn, K.; Sanders, P. Hash tables and associative arrays. In *Algorithms and Data Structures: The Basic Toolbox*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 81–98.
11. Oracle Corporation Java® Platform, Standard Edition and Java Development Kit Version 17 API Specification. Class Object, 2025. Available online: https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Object.html#hashCode() (accessed on 1 April 2025).
12. Menezes, A.J.; Van Oorschot, P.C.; Vanstone, S.A. *Handbook of Applied Cryptography*; CRC Press: Boca Raton, FL, USA, 1997.
13. Gisin, N.; Ribordy, G.; Tittel, W.; Zbinden, H. Quantum cryptography. *Rev. Mod. Phys.* **2002**, *74*, 145. [CrossRef]
14. Li, J.; Li, N.; Zhang, Y.; Wen, S.; Du, W.; Chen, W.; Ma, W. A survey on quantum cryptography. *Chin. J. Electron.* **2018**, *27*, 223–228. [CrossRef]
15. Kumar, A.; Garhwal, S. State-of-the-art survey of quantum cryptography. *Arch. Comput. Methods Eng.* **2021**, *28*, 3831–3868. [CrossRef]
16. Moizuddin, M.; Winston, J.; Qayyum, M. A comprehensive survey: Quantum cryptography. In Proceedings of the 2017 2nd International Conference on Anti-Cyber Crimes (ICACC), Abha, Saudi Arabia, 26–27 March 2017; pp. 98–102.
17. Ablayev, F.; Ablayev, M.; Vasiliev, A.; Ziatdinov, M. Quantum Fingerprinting and Quantum Hashing. Computational and Cryptographical Aspects. *Balt. J. Mod. Comput.* **2016**, *4*, 860–875. [CrossRef]
18. Rozenman, G.G.; Kundu, N.K.; Liu, R.; Zhang, L.; Maslennikov, A.; Reches, Y.; Youm, H.Y. The quantum internet: A synergy of quantum information technologies and 6G networks. *IET Quantum Commun.* **2023**, *4*, 147–166. [CrossRef]
19. Radanliev, P. Artificial intelligence and quantum cryptography. *J. Anal. Sci. Technol.* **2024**, *15*, 4. [CrossRef]
20. Kashefi, E.; Kerenidis, I. Statistical Zero Knowledge and quantum one-way functions. *Theor. Comput. Sci.* **2007**, *378*, 101–116. [CrossRef]
21. Hosoyamada, A.; Yasuda, K. Building quantum-one-way functions from block ciphers: Davies–Meyer and Merkle–Damgard constructions. In Proceedings of Advances in Cryptology—ASIACRYPT 2018, Brisbane, Australia, 2–6 December 2018; Volume 11272, pp. 275–304. [CrossRef]
22. Buhrman, H.; Cleve, R.; Watrous, J.; de Wolf, R. Quantum Fingerprinting. *Phys. Rev. Lett.* **2001**, *87*, 167902. [CrossRef]
23. Ablayev, F.; Vasiliev, A. Cryptographic quantum hashing. *Laser Phys. Lett.* **2014**, *11*, 025202. [CrossRef]
24. Ablayev, F.; Ablayev, M. On the concept of cryptographic quantum hashing. *Laser Phys. Lett.* **2015**, *12*, 125204. [CrossRef]
25. Ablayev, F.; Ablayev, M.; Vasiliev, A. On the balanced quantum hashing. *J. Phys. Conf. Ser.* **2016**, *681*, 012019. [CrossRef]
26. Ziatdinov, M. From Graphs to Keyed Quantum Hash Functions. *Lobachevskii J. Math.* **2016**, *37*, 704–711. [CrossRef]
27. Gottesman, D.; Chuang, I. Quantum Digital Signatures. *arXiv* **2001**, arXiv:quant-ph/0105032.
28. Behera, A.; Paul, G. Quantum to classical one-way function and its applications in quantum money authentication. *Quantum Inf. Process.* **2018**, *17*, 1–24. [CrossRef]
29. Shang, T.; Tang, Y.; Chen, R.; Liu, J. Full quantum one-way function for quantum cryptography. *Quantum Eng.* **2020**, *2*, e32. [CrossRef]
30. Freivalds, R. Fast probabilistic algorithms. In *Mathematical Foundations of Computer Science 1979*; Becvar, J., Ed.; Springe: Berlin/Heidelberg, Germany, 1979; Volume 74, pp. 57–69. [CrossRef]
31. Freivalds, R. Probabilistic Two-Way Machines. In Proceedings of the International Symposium on Mathematical Foundations of Computer Science, Pleso, Czechoslovakia, 31 August–4 September 1981; pp. 33–45.
32. Ambainis, A.; Freivalds, R. 1-way quantum finite automata: Strengths, weaknesses and generalizations. In Proceedings of the 39th IEEE Conference on Foundation of Computer Science, Washington, DC, USA, 8–11 November 1998; pp. 332–342. [CrossRef]
33. Ambainis, A.; Nahimovs, N. Improved Constructions of Quantum Automata. In *Theory of Quantum Computation, Communication, and Cryptography*; Kawano, Y.; Mosca, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5106, pp. 47–56. [CrossRef]
34. Ambainis, A.; Nahimovs, N. Improved constructions of quantum automata. *Theor. Comput. Sci.* **2009**, *410*, 1916–1922. [CrossRef]
35. Gainutdinova, A. On relative complexity of quantum and classical branching programs. *Discret. Math. Appl.* **2002**, *12*, 515–526. [CrossRef]
36. Ablayev, F.; Gainutdinova, A.; Karpinski, M.; Moore, C.; Pollett, C. On the computational power of probabilistic and quantum branching program. *Inf. Comput.* **2005**, *203*, 145–162. [CrossRef]

37. Ablayev, F.; Vasiliev, A. Algorithms for Quantum Branching Programs Based on Fingerprinting. *Electron. Proc. Theor. Comput. Sci.* **2009**, *9*, 1–11. [CrossRef]
38. Vasiliev, A.; Latypov, M.; Ziatdinov, M. Minimizing collisions for quantum hashing. *J. Eng. Appl. Sci.* **2017**, *12*, 877–880. [CrossRef]
39. Ablayev, F.; Ablayev, M.; Vasiliev, A. Quantum hashing and fingerprinting for quantum cryptography and computations. In Proceedings of the International Computer Science Symposium in Russia, Ekaterinburg, Russia, 29 June–3 July 2020; pp. 1–15.
40. Vasiliev, A. Quantum Hashing for Finite Abelian Groups. *Lobachevskii J. Math.* **2016**, *37*, 751–754. [CrossRef]
41. Ziatdinov, M. Quantum hashing. Group approach. *Lobachevskii J. Math.* **2016**, *37*, 222–226. [CrossRef]
42. Zinnatullin, I. Cryptographic Properties of the Quantum Hashing Based on Expander Graphs. *Lobachevskii J. Math.* **2023**, *44*, 776–787. [CrossRef]
43. Gavinsky, D.; Ito, T. Quantum fingerprints that keep secrets. *Quantum Inf. Comput.* **2013**, *13*, 583–606. [CrossRef]
44. Ablayev, F.; Vasiliev, A. Quantum hashing and Fourier transform. In *Journal of Physics: Conference Series*; IOP Publishing: Philadelphia, PA, USA, 2020; Volume 1680, p. 012001.
45. Khadieva, A. Quantum hashing algorithm implementation. *arXiv* **2024**, arXiv:quant-ph/2024.
46. Turaykhanov, D.A.; Akat'ev, D.O.; Vasiliev, A.V.; Ablayev, F.M.; Kalachev, A.A. Quantum hashing via single-photon states with orbital angular momentum. *Phys. Rev. A* **2021**, *104*, 052606. [CrossRef]
47. Akat'ev, D.; Vasiliev, A.; Shafeev, N.; Ablayev, F.; Kalachev, A. Multiqudit quantum hashing and its implementation based on orbital angular momentum encoding. *Laser Phys. Lett.* **2022**, *19*, 125205. [CrossRef]
48. Plachta, S.Z.; Hiekkamäki, M.; Yakaryılmaz, A.; Fickler, R. Quantum advantage using high-dimensional twisted photons as quantum finite automata. *Quantum* **2022**, *6*, 752. [CrossRef]
49. Zhao, Y.Y.; Li, K.; Li, C.; Zheng, S.; He, Z. Experimental Demonstration Advantage of Photonic Finite Automata. In Proceedings of the 2023 Asia Communications and Photonics Conference/2023 International Photonics and Optoelectronics Meetings (ACP/POEM), Wuhan, China, 4–7 November 2023; pp. 01–03.
50. Zinnatullin, I.; Khadiev, K.; Khadieva, A. Efficient Implementation of Amplitude Form of Quantum Hashing Using State-of-the-Art Quantum Processors. *Russ. Microelectron.* **2023**, *52*, S390–S394. [CrossRef]
51. Kālis, M. Kvantu Algoritmu Realizācija Fiziskā Kvantu Datorā. Master's Thesis, University of Latvia, Riga, Latvia, 2018.
52. Ziiatdinov, M.; Khadieva, A.; Yakaryılmaz, A. GAPs for Shallow Implementation of Quantum Finite Automata. In Proceedings of the 16th International Conference on Automata and Formal Languages (AFL 2023), Eger, Hungary, 5–7 September 2023; Zsolt Gazdag, S.I., Kovasznai, G., Eds.; Open Publishing Association: Waterloo, Australia, 2023; Volume 386, pp. 269–280. [CrossRef]
53. Khadieva, A.; Salehi, O.; Yakaryı lmaz, A. A Representative Framework for Implementing Quantum Finite Automata on Real Devices. In Proceedings of the UCNC 2024, Pohang, Republic of Korea, 17–21 June 2024; Volume 14776, pp.163–177.
54. Ziiatdinov, M.; Khadieva, A.; Khadiev, K. Shallow Implementation of Quantum Fingerprinting with Application to Quantum Finite Automata. *Front. Comput. Sci.* **2025**, *7*.
55. Khadiev, K.; Khadieva, A.; Chen, Z.; Wu, J. Implementation of Quantum Fourier Transform and Quantum Hashing for a Quantum Device with Arbitrary Qubits Connection Graphs. *arXiv* **2025**, arXiv:2501.18677.
56. Le Gall, F. Exponential separation of quantum and classical online space complexity. *Theory Comput. Syst.* **2009**, *45*, 188–202. [CrossRef]
57. Ablayev, F.; Ablayev, M.; Khadiev, K.; Salihova, N.; Vasiliev, A. Quantum Algorithms for String Processing. In Proceedings of the Mesh Methods for Boundary-Value Problems and Applications, Kazan, Russia, 20–25 October 2020; Volume 141, pp. 1–14.
58. Ablayev, F.; Salikhova, N.; Ablayev, M. Hybrid Classical–Quantum Text Search Based on Hashing. *Mathematics* **2024**, *12*, 1858. [CrossRef]
59. Khadiev, K.; Khadieva, A. Quantum Online Streaming Algorithms with Logarithmic Memory. *Int. J. Theor. Phys.* **2021**, *60*, 608–616. [CrossRef]
60. Khadiev, K.; Khadieva, A. Quantum and Classical Log-Bounded Automata for the Online Disjointness Problem. *Mathematics* **2022**, *10*, 143. [CrossRef]
61. Khadiev, K.; Khadieva, A.; Mannapov, I. Quantum Online Algorithms with Respect to Space and Advice Complexity. *Lobachevskii J. Math.* **2018**, *39*, 1210–1220. [CrossRef]
62. Khadiev, K.; Khadieva, A.; Ziatdinov, M.; Mannapov, I.; Kravchenko, D.; Rivosh, A.; Yamilov, R. Two-Way and One-Way Quantum and Classical Automata with Advice for Online Minimization Problems. *Theor. Comput. Sci.* **2022**, *920*, 76–94. [CrossRef]
63. Khadiev, K.; Khadieva, A.; Knop, A. Exponential separation between quantum and classical ordered binary decision diagrams, reordering method and hierarchies. *Nat. Comput.* **2023**, *22*, 723–736. [CrossRef]
64. Ablayev, F.; Gainutdinova, A.; Khadiev, K.; Yakaryılmaz, A. Very narrow quantum OBDDs and width hierarchies for classical OBDDs. *Lobachevskii J. Math.* **2016**, *37*, 670–682. [CrossRef]
65. Vasiliev, A. A model of quantum communication device for quantum hashing. *J. Phys. Conf. Ser.* **2016**, *681*, 012020. [CrossRef]

66. Gainutdinova, A.; Yakaryılmaz, A. Nondeterministic Unitary OBDDs. In Proceedings of the Computer Science—Theory and Applications—12th International Computer Science Symposium in Russia, CSR 2017, Kazan, Russia, 8–12 June 2017; Weil, P., Ed.; Springer: Cham, Switzerland, 2017; Volume 10304, pp. 126–140. [CrossRef]
67. Yakaryılmaz, A.; Say, A.C.C. Languages recognized by nondeterministic quantum finite automata. *Quantum Inf. Comput.* **2010**, *10*, 747–770. [CrossRef]
68. Khadieva, A.; Ziatdinov, M. Deterministic Construction of QFAs Based on the Quantum Fingerprinting Technique. *Lobachevskii J. Math.* **2023**, *44*, 713–723. [CrossRef]
69. Barenco, A.; Bennett, C.H.; Cleve, R.; DiVincenzo, D.P.; Margolus, N.; Shor, P.; Sleator, T.; Smolin, J.A.; Weinfurter, H. Elementary gates for quantum computation. *Phys. Rev. A* **1995**, *52*, 3457. [CrossRef]
70. Grover, L.K. A Fast Quantum Mechanical Algorithm for Database Search. In Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, Philadelphia, PA, USA, 22–24 May 1996; pp. 212–219. [CrossRef]
71. Boyer, M.; Brassard, G.; Høyer, P.; Tapp, A. Tight bounds on quantum searching. *Fortschritte Der Phys.* **1998**, *46*, 493–505. [CrossRef]
72. Zheng, S.; Qiu, D. From quantum query complexity to state complexity. In *Computing with New Resources: Essays Dedicated to Jozef Gruska on the Occasion of His 80th Birthday*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 231–245.
73. Holevo, A.S. Some estimates of the information transmitted by quantum communication channel (Russian). *Probl. Pered. Inform. [Probl. Inf. Transm.]* **1973**, *9*, 3–11.
74. Nayak, A. Optimal Lower Bounds For Quantum Automata And Random Access Codes. In Proceedings of the Foundations of Computer Science, New York, NY, USA, 17–18 October 1999; pp. 369–376. [CrossRef]
75. Ablayev, F.; Ablayev, M. Quantum hashing via ϵ-universal hashing constructions and classical fingerprinting. *Lobachevskii J. Math.* **2015**, *36*, 89–96. [CrossRef]
76. Chen, S.; Moore, C.; Russell, A. Small-Bias Sets for Nonabelian Groups. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*; Raghavendra, P.; Raskhodnikova, S., Jansen, K., Rolim, J.D., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; Volume 8096, pp. 436–451. [CrossRef]
77. Naor, J.; Naor, M. Small-bias Probability Spaces: Efficient Constructions and Applications. In Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, 13–17 May 1990; STOC '90, pp. 213–223. [CrossRef]
78. Alon, N.; Roichman, Y. Random Cayley graphs and expanders. *Random Struct. Algorithms* **1994**, *5*, 271–284. [CrossRef]
79. Ben-Aroya, A.; Ta-Shma, A. Constructing Small-Bias Sets from Algebraic-Geometric Codes. In Proceedings of the Foundations of Computer Science 2009, Atlanta, GA, USA, 24–27 October 2009; pp. 191–197. [CrossRef]
80. de Wolf, R. Quantum Computing and Communication Complexity. Ph.D. Thesis, University of Amsterdam, Amsterdam, The Netherlands, 2001.
81. Vasiliev, A. Collision Resistance of the OAM-based Quantum Hashing. *Lobachevskii J. Math.* **2023**, *44*, 758–761. [CrossRef]
82. Vasiliev, A. Constant-Depth Algorithm for Quantum Hashing. *Russ. Microelectron.* **2023**, *52*, S399–S402. [CrossRef]
83. Wigderson, A.; Xiao, D. Derandomizing the Ahlswede-Winter matrix-valued Chernoff bound using pessimistic estimators, and applications. *Theory Comput.* **2008**, *4*, 53–76. [CrossRef]
84. Möttönen, M.; Vartiainen, J.J. Decompositions of general quantum gates. *arXiv* **2006**, arXiv:quant-ph/0504100.
85. Bergholm, V.; Vartiainen, J.J.; Möttönen, M.; Salomaa, M.M. Quantum circuits with uniformly controlled one-qubit gates. *Phys. Rev. A At. Mol. Opt. Phys.* **2005**, *71*, 052330. [CrossRef]
86. Maldonado, T.J.; Flick, J.; Krastanov, S.; Galda, A. Error rate reduction of single-qubit gates via noise-aware decomposition into native gates. *Sci. Rep.* **2022**, *12*, 6379. [CrossRef]
87. Salehi, Ö.; Yakaryılmaz, A. Cost-efficient QFA Algorithm for Quantum Computers. *arXiv* **2021**, arXiv:2107.02262.
88. Say, A.C.C.; Yakaryılmaz, A. Quantum finite automata: A modern introduction. In *Computing with New Resources*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 208–222. [CrossRef]
89. Kitaev, A.Y. Quantum measurements and the Abelian stabilizer problem. *arXiv* **1995**, arXiv:quant-ph/9511026
90. Draper, T.G. Addition on a quantum computer. *arXiv* **2000**, arXiv:quant-ph/0008033.
91. Slepnev, V.; Gubaydullin, A.; Vinokur, V. Fluxonium-based superconducting qubit magnetometer: Optimization of phase estimation algorithms. *Phys. Rev. B* **2024**, *110*, 214423. [CrossRef]
92. Brassard, G.; Høyer, P.; Mosca, M.; Tapp, A. Quantum amplitude amplification and estimation. *Contemp. Math.* **2002**, *305*, 53–74.
93. Harrow, A.W.; Hassidim, A.; Lloyd, S. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.* **2009**, *103*, 150502. [CrossRef] [PubMed]
94. Shor, P.W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Rev.* **1999**, *41*, 303–332. [CrossRef]
95. Wright, L.; Mc Keever, C.; First, J.T.; Johnston, R.; Tillay, J.; Chaney, S.; Rosenkranz, M.; Lubasch, M. Noisy intermediate-scale quantum simulation of the one-dimensional wave equation. *Phys. Rev. Res.* **2024**, *6*, 043169. [CrossRef]

96. Lee, T.; Mittal, R.; Reichardt, B.W.; Špalek, R.; Szegedy, M. Quantum query complexity of state conversion. In Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science, Palm Springs, CA, USA, 22–25 October 2011; pp. 344–353.
97. Belovs, A. Quantum walks and electric networks. *arXiv* **2013**, arXiv:1302.3143.
98. Belovs, A.; Childs, A.M.; Jeffery, S.; Kothari, R.; Magniez, F. Time-efficient quantum walks for 3-distinctness. In Proceedings of the International Colloquium on Automata, Languages, and Programming, Riga, Latvia, 8–12 July 2013; Springer: Berlin/Heidelberg, Germany, 2013; pp. 105–122.
99. Montanaro, A. Quantum speedup of branch-and-bound algorithms. *Phys. Rev. Res.* **2020**, *2*, 013056. [CrossRef]
100. Montanaro, A. Quantum-Walk Speedup of Backtracking Algorithms. *Theory Comput.* **2018**, *14*, 1–24. [CrossRef]
101. Park, B.; Ahn, D. Reducing CNOT count in quantum Fourier transform for the linear nearest-neighbor architecture. *Sci. Rep.* **2023**, *13*, 8638. [CrossRef]
102. Fowler, A.; Devitt, S.; Hollenberg, L. Implementation of Shor's algorithm on a linear nearest neighbour qubit array. *Quantum Inf. Comput.* **2004**, *4*, 237–251. [CrossRef]
103. Saeedi, M.; Wille, R.; Drechsler, R. Synthesis of quantum circuits for linear nearest neighbor architectures. *Quantum Inf. Process.* **2011**, *10*, 355–377. [CrossRef]
104. Wille, R.; Lye, A.; Drechsler, R. Exact reordering of circuit lines for nearest neighbor quantum architectures. *IEEE Trans. -Comput. Des. Integr. Circuits Syst.* **2014**, *33*, 1818–1831. [CrossRef]
105. Kole, A.; Datta, K.; Sengupta, I. A new heuristic for *N*-dimensional nearest neighbor realization of a quantum circuit. *IEEE Trans. -Comput. Des. Integr. Circuits Syst.* **2017**, *37*, 182–192. [CrossRef]
106. Bhattacharjee, A.; Bandyopadhyay, C.; Wille, R.; Drechsler, R.; Rahaman, H. Improved look-ahead approaches for nearest neighbor synthesis of 1D quantum circuits. In Proceedings of the 2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSID), Delhi, India, 5–9 January 2019; pp. 203–208.
107. Dreier, F.; Fleckenstein, C.; Aigner, G.; Fellner, M.; Stahn, R.; Lanthaler, M.; Lechner, W. Connectivity-aware Synthesis of Quantum Algorithms. *arXiv* **2025**, arXiv:2501.14020.
108. Ambainis, A.; Nayak, A.; Ta-Shma, A.; Vazirani, U. Dense quantum coding and quantum finite automata. *J. ACM* **2002**, *49*, 496–511. [CrossRef]
109. Ambainis, A.; Watrous, J. Two–way finite automata with quantum and classical states. *Theor. Comput. Sci.* **2002**, *287*, 299–311. [CrossRef]
110. Yakaryilmaz, A. Log-space counter is useful for unary languages by help of a constant-size quantum register. *arXiv* **2013**, arXiv:1309.4767.
111. Yakaryılmaz, A.; Say, A.C. Efficient probability amplification in two-way quantum finite automata. *Theor. Comput. Sci.* **2009**, *410*, 1932–1941. [CrossRef]
112. Zheng, S.; Qiu, D.; Gruska, J.; Li, L.; Mateus, P. State succinctness of two-way finite automata with quantum and classical states. *Theor. Comput. Sci.* **2013**, *499*, 98–112. [CrossRef]
113. Zheng, S.; Gruska, J.; Qiu, D. On the state complexity of semi-quantum finite automata. *RAIRO-Theor. Informatics -Appl. Théor. Appl.* **2014**, *48*, 187–207. [CrossRef]
114. Ambainis, A.; Yakaryılmaz, A. Automata: From Mathematics to Applications. Chapter Automata and quantum computing. In *Handbook of Automata Theory*; EMS Press: Berlin, Germany, 2001; pp. 1457–1493. [CrossRef]
115. Gainutdinova, A.; Yakaryılmaz, A. Unary probabilistic and quantum automata on promise problems. *Quantum Inf. Process.* **2018**, *17*, 28. [CrossRef]
116. Hu, Y.; Melnyk, D.; Wang, Y.; Wattenhofer, R. Space complexity of streaming algorithms on universal quantum computers. In Proceedings of the Theory and Applications of Models of Computation: 16th International Conference, TAMC 2020, Changsha, China, 18–20 October 2020; pp. 275–286.
117. Nakanishi, M.; Yakaryılmaz, A. Classical and quantum counter automata on promise problems. In Proceedings of the International Conference on Implementation and Application of Automata, Umeå, Sweden, 18–21 August 2015; pp. 224–237.
118. Nakanishi, M.; Yakaryılmaz, A.; Gainutdinova, A. New results on classical and quantum counter automata. *Discret. Math. Theor. Comput. Sci.* **2019**, *21*. [CrossRef]
119. Gainutdinova, A. Quantum and classical nondeterministic OBDDs. *Uchenye Zap. Kazan. Univ. Seriya -Fiz. Nauk.* **2024**, *166*, 470–484. [CrossRef]
120. Gainutdinova, A.F. On the complexity of computing the "Shuffled Inequality" function in classical and quantum NOBDDs. *Izv. Vyss. Uchebnykh Zaved. Mat.* **2025**, *69*, 3–14. [CrossRef]
121. Gainutdinova, A.F. Comparative complexity of quantum and classical OBDDs for total and partial functions. *Russ. Math.* **2015**, *59*, 26–35. [CrossRef]

122. Khadiev, K.; Khadieva, A. Two-way quantum and classical machines with small memory for online minimization problems. In Proceedings of the International Conference on Micro- and Nano-Electronics 2018, Zvenigorod, Russia, 1–5 October 2018; Volume 11022, p. 110222T. [CrossRef]
123. Khadiev, K.; Khadieva, A. Two-Way Quantum and Classical Automata with Advice for Online Minimization Problems. In Proceedings of the Formal Methods. FM 2019 International Workshops, Porto, Portugal, 7–11 October 2019; pp. 428–442.
124. Yuan, Q. *Quantum Online Algorithms*; University of California: Santa Barbara, CA, USA, 2009.
125. Knuth, D.E.; Morris, J.H., Jr.; Pratt, V.R. Fast pattern matching in strings. *SIAM J. Comput.* **1977**, *6*, 323–350. [CrossRef]
126. Knuth, D.E. The Dangers of Computer-Science Theory. In Proceedings of the Fourth International Congress for Logic, Methodology and Philosophy of Science, Bucharest, Romania, 29 August–4 September 1971; Suppes, P., Henkin, L., Joja, A., Moisil, G.C., Eds.; Elsevier: Amsterdam, The Netherlands, 1973; Volume 74, pp. 189–195. [CrossRef]
127. Matiyasevich, Y.V. Real-time recognition of the inclusion relation. *J. Sov. Math.* **1973**, *1*, 64–70. [CrossRef]
128. Allauzen, C.; Crochemore, M.; Raffinot, M. Factor oracle: A new structure for pattern matching. In Proceedings of the SOFSEM'99: Theory and Practice of Informatics: 26th Conference on Current Trends in Theory and Practice of Informatics, Milovy, Czech Republic, 27 November–4 December 1999; pp. 295–310.
129. Wu, S.; Manber, U. *A Fast Algorithm for Multi-Pattern Searching*; Technical Report TR-94-17; Department of Computer Science, University of Arizona: Tucson, AZ, USA, 1994.
130. Boyer, R.S.; Moore, J.S. A fast string searching algorithm. *Commun. ACM* **1977**, *20*, 762–772. [CrossRef]
131. Manber, U.; Myers, G. Suffix Arrays: A New Method for on-Line String Searches. In Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, San Francisco, CA, USA, 22–24 January 1990; pp. 319–327.
132. Abouelhoda, M.I.; Kurtz, S.; Ohlebusch, E. Replacing suffix trees with enhanced suffix arrays. *J. Discret. Algorithms* **2004**, *2*, 53–86. [CrossRef]
133. Li, Z.; Li, J.; Huo, H. Optimal In-Place Suffix Sorting. In Proceedings of the String Processing and Information Retrieval, Lima, Peru, 9–11 October 2018; pp. 268–284.
134. Landau, G.M.; Kasai, T.; Lee, G.; Arimura, H.; Arikawa, S.; Park, K. Linear-time longest-common-prefix computation in suffix arrays and its applications. In Proceedings of the Combinatorial Pattern Matching: 12th Annual Symposium, CPM 2001, Jerusalem, Israel, 1–4 July 2001; Volume 2089, pp. 181–192.
135. Weiner, P. Linear pattern matching algorithms. In Proceedings of the 14th Annual Symposium on Switching and Automata Theory (swat 1973), Iowa City, IA, USA, 15-17 October 1973; pp. 1–11. [CrossRef]
136. Aho, A.V.; Hopcroft, J.E. *The Design and Analysis of Computer Algorithms*; Pearson Education India: Tamil Nadu, India, 1974.
137. Amir, A.; Landau, G.M.; Lewenstein, M.; Sokol, D. Dynamic text and static pattern matching. *ACM Trans. Algorithms* **2007**, *3*, 19-es. [CrossRef]
138. Aho, A.V.; Corasick, M.J. Efficient string matching: An aid to bibliographic search. *Commun. ACM* **1975**, *18*, 333–340. [CrossRef]
139. Commentz-Walter, B. A string matching algorithm fast on the average. In Proceedings of the International colloquium on automata, languages, and programming, Ulrichland, Austria, 16–20 July 1979; pp. 118–132.
140. Marino, F.P.; Faro, S.; Scardace, A. Practical implementation of a quantum string matching algorithm. In Proceedings of the Proceedings of the 2024 Workshop on Quantum Search and Information Retrieval, Pisa, Italy, 3 June 2024; pp. 17–24.
141. Ramesh, H.; Vinay, V. String matching in $O(\sqrt{n} + \sqrt{m})$ quantum time. *J. Discret. Algorithms* **2003**, *1*, 103–110. [CrossRef]
142. Montanaro, A. Quantum pattern matching fast on average. *Algorithmica* **2017**, *77*, 16–39. [CrossRef]
143. Soni, K.K.; Rasool, A. Pattern matching: A quantum oriented approach. *Procedia Comput. Sci.* **2020**, *167*, 1991–2002. [CrossRef]
144. Khadiev, K.; Serov, D. Quantum Algorithm for the Multiple String Matching Problem. In Proceedings of the International Conference on Current Trends in Theory and Practice of Computer Science, Bratislava, Slovak Republic, 20–23 January 2025; Volume 15539, pp. 58–69.
145. Ablayev, F.; Salikhova, N.; Ablayev, M. Quantum search in a dictionary based on fingerprinting-hashing. *arXiv* **2024**, arXiv:2412.11422.
146. Kaye, P.; Laflamme, R.; Mosca, M. *An Introduction to Quantum Computing*; Oxford University Press: Oxford, UK, 2006.