Kazan Federal University

Computation and Cryptography with Qu-bits

Papers of the International Workshop
Kazan, June 7, 2017

Kazan, 2017

The workshop is devoted to discussions on classical and quantum computation and cryptography. CCQ workshop was held together with The 12th International Computer Science Symposium in Russia (CSR 2017). The topics include, but are not limited to: Quantum and classic computing, Algorithms ,Computational complexity ,Cryptography ,Quantum and random walks ,Automata theory

Program Committee Chairs:

- Farid Ablayev (Kazan Federal University, Russia)
- Andris Ambainis (University of Latvia, Latvia, co-chair)

Program Committee:

- Aleksandrs Belovs (University of Latvia, Latvia)
- Julien Degorre (University della Svizzera italiana, Switzerland)
- Kamil Khadiev (University of Latvia, Latvia; Kazan Federal university)
- Francois Le Gall (Kyoto University, Japan)
- Maris Ozols (University of Cambridge, UK)
- Renato Portugal (National Laboratory of Scientific Computing, Brazil)
- A. C. Cem Say (Bogazici University, Turkey)
- Dominique Unruh (University of Tartu, Estonia)
- Marcos Villagra (Universidad Nacional de Asuncion, Paraguay)
- Abuzer Yakaryilmaz (University of Latvia, Latvia)

Computation and Cryptography with Qu-bits
Papers of the International Workshop
Kazan, June 7, 2017

Contents

Branching Program Complexity of Quantum Hashing

Farid Ablayev Marat Ablayev

Kazan Federal University, Krmlevskaya 18, Kazan, 420008, Russia
fablayev@gmail.com, mablayev@gmail.com

Abstract

We investigate Branching Program complexity measure of quantum hashing. Quantum (δ, ϵ)-hash function $\psi : \mathbb{F}_q \to (\mathcal{H}^2)^{\otimes s}$ hashes elements of finite field $\mathbb{F}_q$ into s-qubit quantum states. This function is one-way δ-resistant and is collision ϵ-resistant.

We consider two complexity measures for Quantum Branching Program (QBP): a number $Width(Q)$ of QBP Q qubits and a number Time(Q) of QBP Q computational steps. We show that quantum (δ, ϵ)-hash function can be computed effectively. Namely, we present QBP Q for quantum (δ, ϵ)-hash function with the following complexity characteristics: $Width(Q) = O(\log \log q)$ and $Time(Q) = \log q$.

We prove that such QBP construction is optimal. That is, we prove lower bounds $\Omega(\log \log q)$ of QBP width and $\Omega(\log q)$ of QBP time for quantum (δ, ϵ)-hash function computation.

Keywords: Hashing, Hash-function, Quantum Hash-function, Digital Signature.

1 Introduction

In [2] we explicitly defined a notion of quantum hashing as a generalization of classical hashing and presented examples of quantum hash functions. It appeared that Gottesman-Chuang quantum signature schemes [4] are based on functions which are actually quantum hash functions. Those functions have "unconditionally one-way" property based on Holevo Theorem [5]. More information on the role of quantum hashing for the post quantum cryptography, possible application of quantum hashing for quantum signature protocols, and technological expectations for realization of quantum signature schemes are presented in [6].

Recall that in the classical setting a cryptographic hash function h should be computed effectively and should have the following properties (see for example [7]). (1) Pre-image resistance: Given $h(x)$, it should be difficult to find x, that is, these hash functions are one-way functions. (2) Second pre-image resistance: Given x_1, it should be difficult to find an x_2, such that $h(x_1) = h(x_2)$. (3) Collision resistance: It should be difficult to find any pair of distinct x_1, x_2, such that $h(x_1) = h(x_2)$. Note, that there are no one-way functions that are known to be provably more difficult to invert than to compute, the security of cryptographic hash functions is "computationally conditional".

In the paper we consider quantum (δ, ϵ)-hash functions construction based on ϵ-biased sets. Such quantum (δ, ϵ)-hash function $\psi : \mathbb{F}_q \to (\mathcal{H}^2)^{\otimes s}$ hashes elements of finite field

$\mathbb{F}_q$ into s-qubit quantum states. The notion of (δ, ϵ)-hash function combines together a notion of pre-image (one-way) quantum δ-resistance property and the notion of quantum collision ϵ-resistance properties. These properties are quantum generalization of classical one-way resistance and collision resistance properties required for classical hash functions.

Important property for hash function is a computational effectiveness. In the paper we show that considered construction of quantum (δ, ϵ)-hash function is computed effectively in the model of Quantum Branching Programs [3]. We consider two complexity measures: a number $Width(Q)$ of qubits that QBP Q uses for computation and a number Time(Q) of computational steps of QBP Q. Such QBP Q is of $Width(Q) = O(\log \log q)$ and $Time(Q) = \log q$.

We prove that such QBP construction is optimal. That is, we prove lower bounds $\Omega(\log \log q)$ for QBP width and $\Omega(\log q)$ for QBP time for quantum (δ, ϵ)-hash function presentation.

2 Preliminaries

Recall that mathematically a qubit is described as a unit vector in the two-dimensional Hilbert complex space $\mathcal{H}^2$. Let $s \geq 1$. Let $(\mathcal{H}^2)^{\otimes s}$ be the 2^s-dimensional Hilbert space, describing the states of s qubits. For an integer $j \in \{0, \ldots, 2^s - 1\}$ let $\sigma = \sigma_1 \ldots \sigma_s$ be a binary presentation of j. We use (as usual) notations $|j\rangle$ and $|\sigma\rangle$ to denote quantum state $|\sigma_1\rangle \cdots |\sigma_s\rangle = |\sigma_1\rangle \otimes \ldots \otimes |\sigma_s\rangle$.

We let q to be a prime and $\mathbb{F}_q$ be a finite field of order q. Let Σ^k be a set of words of length k over a finite alphabet Σ. Let $\mathbb{X}$ be a finite set. In the paper we let $\mathbb{X} = \Sigma^k$, or $\mathbb{X} = \mathbb{F}_q$. In the last case we will also consider that $\mathbb{X}$ is a set $\{0,1\}^k$ of binary sequences of length $k = \log q$.

We define classical-quantum (or just quantum) function ψ to be a function

$$\psi : \mathbb{X} \to (\mathcal{H}^2)^{\otimes s}.$$

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

Definition 2.1 *Let X be random variable distributed over $\mathbb{X}$ $\{Pr[X = w] : w \in \mathbb{X}\}$. Let $\psi : \mathbb{X} \to (\mathcal{H}^2)^{\otimes s}$ be a quantum function. Let Y is any random variable over $\mathbb{X}$ obtained by some mechanism $\mathcal{M}$ making measurement to the encoding ψ of X and decoding the result of measurement to $\mathbb{X}$. Let $\delta > 0$. We call a quantum function ψ a one-way δ-resistant function if for any mechanism $\mathcal{M}$, the probability $Pr[Y = X]$ that $\mathcal{M}$ successfully decodes Y is bounded by δ*

$$Pr[Y = X] \leq \delta.$$

For the cryptographic purposes it is natural to expect (and we do this in the rest of the paper) that 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, fundamental result of quantum informatics known as Holevo's Theorem

[5] states that a quantum measurement can only give s bits of information about the state. We will use here the following particular version [10] of Holevo's Theorem.

Property 2.1 *Let X be random variable uniformly distributed over a k bit binary words $\{0,1\}^k$. Let $\psi : \{0,1\}^k \to (\mathcal{H}^2)^{\otimes s}$ be a quantum function. Let Y be a random variable over $\mathbb{X}$ obtained by some mechanism $\mathcal{M}$ making some measurement of the encoding ψ of X and decoding the result of measurement to $\{0,1\}^k$. Then our probability of correct decoding is given by*

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

Collision resistance. The following definition was presented in [1].

Definition 2.2 *Let $\delta > 0$. 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 elements,*

$$|\langle \psi(w) \,|\, \psi(w') \rangle| \leq \epsilon.$$

Note that the above inequality means almost orthogonality (ϵ orthogonality) of quantum states $|\psi(w)\rangle$ and $|\psi(w')\rangle$. Well known that orthogonality of quantum states provides distinguishability of these states. In content of collision notion almost orthogonality means good collision resistant property. That is, let us denote $Pr_{\mathcal{M}}[v = w]$ a probability that some test $\mathcal{M}$ having quantum hashes $|\psi(v)\rangle$ and $|\psi(w)\rangle$ outputs the result "$v = w$" (outputs the result "$|\psi(v)\rangle = |\psi(w)\rangle$"). For example, known *SWAP*-test [4] provides

$$Pr_{swap}[v = w] \leq \frac{1}{2}(1 + \epsilon^2).$$

The *REVERSE*-test [4, 1] provides

$$Pr_{reverse}[v = w] \leq \epsilon^2.$$

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

Definition 2.3 *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)-hash function if ψ is a one-way δ-resistant and is a collision ϵ-resistant function.*

3 Computing a quantum hash $|\psi_S(x)\rangle$ by QBP

Quantum hash functions construction via small-biased sets. For an $a \in \mathbb{F}_q$ a character χ_a of $\mathbb{F}_q$ is a homomorphism $\chi_a : \mathbb{F}_q \to \mu_q$, where μ_q is the (multiplicative) group of complex q-th roots of unity, $\chi_a(x) = \omega^{ax}$. Here $\omega = e^{\frac{2\pi i}{q}}$ is a primitive complex qth root of unity. A character $\chi_0 \equiv 1$ is called a trivial character.

- A set $S \subseteq \mathbb{F}_q$ is called ϵ-biased, if for any nontrivial character $\chi \in \{\chi_a : a \in \mathbb{F}_q\}$ $\frac{1}{|S|} \left| \sum_{x \in S} \chi(x) \right| \leq \epsilon$.

We present the result of [11] in the following form.

Property 3.1 *Let $S \subseteq \mathbb{F}_q$ be an ϵ-biased set. Let $H_S = \{h_a(x) = ax \pmod q, \quad a \in S\}$. Then a quantum function $\psi_S : \mathbb{F}_q \to (\mathcal{H}^2)^{\otimes \log |S|}$*

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

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

- In the content of the definition of quantum hash generator [1] and the above consideration it is natural to call the set H_S of functions (formed from ϵ-biased set S) a *uniform quantum (δ, ϵ)-hash generator* for $\delta = O(|S|/(q \log q))$.

Note that ϵ-biased sets are interesting when $|S| \ll |\mathbb{F}_q|$ (as $S = \mathbb{F}_q$ is 0-biased). The seminal paper of Naor and Naor [9] defined these small-biased sets, gave the first explicit constructions of such sets, and demonstrated the power of small-biased sets for several applications.

- Note that a set S of $O(\log q/\epsilon^2)$ elements selected uniformly at random from $\mathbb{F}_q$ is ϵ-biased with a positive probability > 0 [8].

Many other constructions of small-biased sets followed during the last decades (see for example [8]).

As a corollary from Property 3.1 and the above consideration we can state the following.

Property 3.2 *For a small size ϵ-biased set $S = \{a_1, \ldots, a_T\} \subset \mathbb{F}_q$ with $T = O(\log q/\epsilon^2)$, for $s = \log T$, for $\delta = O(1/(q\epsilon^2))$ a quantum uniform (δ, ϵ)-hash generator H_S generates quantum (δ, ϵ)-hash function*

$$\psi_{H_S} \quad : \quad \mathbb{F}_q \to (\mathcal{H}^2)^{\otimes s} \tag{1}$$

$$|\psi_{H_S}(x)\rangle \quad = \quad \frac{1}{\sqrt{T}} \sum_{j=0}^{T-1} \omega^{a_j x} |j\rangle. \tag{2}$$

QBP for quantum hash function ψ_{H_S}. We use a QBP model defined in [3].

A Quantum Branching Program Q over the Hilbert space $(\mathcal{H}^2)^{\otimes s}$ is defined as

$$Q = \langle T, |\psi_0\rangle \rangle,$$

where T is a sequence of l instructions: $T_j = \big(x_{i_j}, U_j(0), U_j(1)\big)$ is determined by the variable x_{i_j} tested on the step j, and $U_j(0)$, $U_j(1)$ are unitary transformations in $(\mathcal{H}^2)^{\otimes s}$.

Vectors $|\psi\rangle \in (\mathcal{H}^2)^{\otimes s}$ are called states (state vectors) of Q, $|\psi_0\rangle \in (\mathcal{H}^2)^{\otimes s}$ is the initial state of Q.

We define a computation of Q on an input $\sigma = \sigma_1 \ldots \sigma_n \in \{0,1\}^n$ as follows:

1. A computation of Q starts from the initial state $|\psi_0\rangle$;

2. The j-th instruction of Q reads the input symbol σ_{i_j} (the value of x_{i_j}) and applies the transition matrix $U_j = U_j(\sigma_{i_j})$ to the current state $|\psi\rangle$ to obtain the state $|\psi'\rangle = U_j(\sigma_{i_j})|\psi\rangle$;

3. The final state is
$$|\psi(\sigma)\rangle = \left(\prod_{j=l}^{1} U_j(\sigma_{i_j})\right)|\psi_0\rangle.$$

Theorem 3.1 *Quantum (δ, ϵ)-hash function (1)*

$$\psi_{H_S} : \mathbb{F}_q \to (\mathcal{H}^2)^{\otimes s}$$

can be computed by quantum branching program Q composed from $s = O(\log\log q)$ qubits in $\log q$ steps.

Proof. Quantum function ψ_{H_S} (1) for an input $x \in \mathbb{F}_q$ determines quantum states (2)

$$|\psi_{H_S}(x)\rangle = \frac{1}{\sqrt{T}}\sum_{j=0}^{T-1}\omega^{a_j x}|j\rangle$$

which is a result of quantum Fourier transformation (QFT) of the initial state

$$|\psi_0\rangle = \frac{1}{\sqrt{T}}\sum_{j=0}^{T-1}|j\rangle.$$

Such a QFT is controlled by the input x. QBP Q for for computing quantum hash $|\psi_{H_S}(x)\rangle$ determined as follows. We represent an integer $x \in \{0, \ldots, q-1\}$ as the bit-string $x = x_0 \ldots x_{\log q - 1}$ that is, $x = x_0 + 2^1 x_1 + \ldots + 2^{\log q - 1} x_{\log q - 1}$. For a binary string $x = x_0 \ldots x_{\log q - 1}$ a Quantum Branching Program Q over the space $(\mathcal{H}^2)^{\otimes s}$ for computing $|\psi_S(x)\rangle$ (composed of $s = \log T$ qubits) is defined as

$$Q = \langle \mathbb{T}, |\psi_0\rangle\rangle,$$

where $|\psi_0\rangle$ is the initial state and $\mathbb{T}$ is a sequence of $\log q$ instructions:

$$\mathbb{T}_j = (x_j, U_j(0), U_j(1))$$

is determined by the variable x_j tested on the step j, and $U_j(0)$, $U_j(1)$ are unitary transformations in $(\mathcal{H}^2)^{\otimes s}$. More precise $U_j(0)$ is $T \times T$ identity matrix. $U_j(1)$ is the $T \times T$ diagonal matrix whose diagonal entries are $\omega^{a_0 2^j}, \omega^{a_1 2^j}, \ldots, \omega^{a_{T-1} 2^j}$ and the off-diagonal elements are all zero. That is,

$$U_j(1) = \begin{bmatrix} \omega^{a_0 2^j} & & & \\ & \omega^{a_1 2^j} & & \\ & & \ddots & \\ & & & \omega^{a_{T-1} 2^j} \end{bmatrix}.$$

We define a computation of Q on an input $x = x_0 \ldots x_{\log q - 1} \in \{0,1\}^{\log q}$ as follows:

1. A computation of Q starts from the initial state $|\psi_0\rangle$;

2. The j-th instruction of Q reads the input symbol x_j (the value of x) and applies the transition matrix $U_j(x_j)$ to the current state $|\psi\rangle$ to obtain the state $|\psi'\rangle = U_j(x_j)|\psi\rangle$;

3. The final state is
$$|\psi_S(x)\rangle = \left(\prod_{j=0}^{\log q-1} U_j(x_j)\right)|\psi_0\rangle.$$

□

Consider the following notations. For the QBP Q from Theorem 3.1 we let $Width(Q) = s$ and $Time(Q) = |\mathbb{T}|$. Next for quantum hash function ψ_{H_S} (1) we let

$$Width(\psi_{H_S}) = \min Width(Q), \qquad Time(\psi_{H_S}) = \min Time(Q)$$

where minimum is taken over all QBPs that compute ψ_{H_S}. Then from Theorem 3.1 we have

$$Width(\psi_{H_S}) = O(\log\log q), \tag{3}$$
$$Time(\psi_{H_S}) = O(\log q). \tag{4}$$

Lower bounds. We present here the following

Theorem 3.2

$$Width(\psi_{H_S}) = \Omega(\log\log q), \tag{5}$$
$$Time(\psi_{H_S}) = \Omega(\log q). \tag{6}$$

QBP Q is a procedure for the function ψ_{H_S} computation. ψ_{H_S} can be presented as follows

$$\psi_{H_S} : \{|\psi_0\rangle\} \times \{0,1\}^{\log q} \to (\mathcal{H}^2)^{\otimes s}.$$

The proof of the lower bound (5) is the immediate corollary from the following statement [1]. We present its proof for completeness.

Lemma 3.1 *Let $\psi : \mathbb{X} \to (\mathcal{H}^2)^{\otimes s}$ be a collision ϵ-resistant function. Then*

$$s \geq \log\log|\mathbb{X}| - \log\log\left(1 + \sqrt{2/(1-\epsilon)}\right) - 1.$$

Proof. First we observe, that from the definition $|||\psi\rangle|| = \sqrt{\langle\psi\,|\,\psi\rangle}$ of the norm it follows that

$$|||\psi\rangle - |\psi'\rangle||^2 = |||\psi\rangle||^2 + |||\psi'\rangle||^2 - 2\langle\psi\,|\,\psi'\rangle.$$

Hence for arbitrary pair w, w' of different elements from $\mathbb{X}$ we have that

$$|||\psi(w)\rangle - |\psi(w')\rangle|| \geq \sqrt{2(1-\epsilon)} \tag{7}$$

We let $\Delta = \sqrt{2(1-\epsilon)}$. For short we let $(\mathcal{H}^2)^{\otimes s} = V$ in this proof. Consider a set $\Phi = \{|\psi(w)\rangle : w \in \mathbb{X}\}$. If we draw a sphere of the radius $\Delta/2$ with the center $|\psi\rangle \in \Phi$

then all such spheres do not intersect pairwise. All these K ($K = |\mathbb{X}|$) spheres are in large sphere of radius $1 + \Delta/2$. The volume of a sphere of a radius r in V is $cr^{2^{s+1}}$ for the complex space V. Constant c depends on the metric of V. From this we have, that the number K is bonded by the number of "small spheres" in the "large sphere"

$$K \leq \frac{c(1+\Delta/2)^{2^{s+1}}}{c(\Delta/2)^{2^{s+1}}}.$$

Hence

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

□

The proof of the lower bound (6) for $Time(\psi_{H_S})$ follows from the proof of Lemma 3.1. The assumption that QBP Q for ψ_{H_S} can test less than $\log q$ (not all $\log q$) variables of inputs $x \in \mathbb{F}_q$ means existence of (at least) two different inputs $w, w' \in \mathbb{F}_q$ such that Q produces the same quantum hashes for w and w', that is, $|\psi(w)\rangle = |\psi(w')\rangle = |\psi\rangle$. The last contradicts (7).

References

[1] Farid Ablayev and Marat Ablayev. Quantum hashing via ϵ-universal hashing constructions and Freivalds' fingerprinting schemas. 16th DCFS 2014, Turku. LNCS Volume 8614 LNCS, 2014, Pages 42-52

[2] Farid Ablayev and Alexander Vasiliev. Cryptographic quantum hashing. *Laser Phys. Lett.* **11**(2):025202, Dec 2013.

[3] F. Ablayev and A. Vasiliev, *Computing Boolean functions via quantum hashing*, Computing with New Resources, Lecture Notes in Computer Science **8808**, 149–160 (2014).

[4] Daniel Gottesman and Isaac Chuang. Quantum digital signatures. arXiv:quantph/0105032, 2001.

[5] Alexander S. Holevo. Some estimates of the information transmitted by quantum communication channel (russian). *Probl. Pered. Inform. [Probl. Inf. Transm.]*, 9(3):311, 1973.

[6] A. Korol'kov. About some applied aspects of quantum cryptography in the context of development of quantum computations and emergense of quantum computations and emergence of quantum computers (russian). *Voprosy kiberbezopasnosty*, 1(9), 2015.

[7] Ryan Amiri and Erika Andersson. Unconditionally Secure Quantum Signature. *Entropy*, 17:5635–5659, 2015.

[8] Avraham Ben-Aroya, Amnon Ta-Shma. Constructing Small-Bias Sets from Algebraic-Geometric Codes. Theory of Computing 9: 253-272 (2013)

[9] J. Naor and M. Naor, *Small-bias probability spaces: Efficient constructions and applications*, Proceedings of the twenty-second annual ACM symposium on Theory of computing, 213–223 (1990).

[10] A. Nayak, *Optimal Lower Bounds for Quantum Automata and Random Access Codes*, arXiv:quant-ph/9904093v3, (1999).

[11] A. Vasiliev, *Quantum Hashing for Finite Abelian Groups*, arXiv:1603.02209 [quant-ph], (2016).

On Block Sensitivity and Fractional Block Sensitivity *

Andris Ambainis Krišjānis Prūsis Jevgēnijs Vihrovs

Faculty of Computing, University of Latvia
Raiņa bulv. 19, Rīga, LV-1586, Latvia
andris.ambainis@lu.lv, krisjanis.prusis@lu.lv, jevgenijs.vihrovs@lu.lv

Abstract

We investigate the relation between the block sensitivity $\mathsf{bs}(f)$ and fractional block sensitivity $\mathsf{fbs}(f)$ complexity measures of Boolean functions. While it is known that $\mathsf{fbs}(f) = O(\mathsf{bs}(f)^2)$, the best known separation achieves $\mathsf{fbs}(f) = \left(\frac{1}{3\sqrt{2}} + o(1)\right)\mathsf{bs}(f)^{3/2}$. We improve the constant factor and show a family of functions that give

$$\mathsf{fbs}(f) = \left(\frac{1}{\sqrt{6}} - o(1)\right)\mathsf{bs}(f)^{3/2}.$$

Keywords: Boolean functions, query complexity, block sensitivity, fractional block sensitivity

1 Introduction

The query complexity of Boolean functions is one of the simplest models of computation. In this setting, the cost of the computation is the number of the input bits one needs to query to decide the value of the function on this input. One of the main challenges is to precisely relate the computational power of the *decision tree complexity* $\mathsf{D}(f)$, *randomized decision tree complexity* $\mathsf{R}(f)$ and *quantum decision tree complexity* $\mathsf{Q}(f)$ (see [2] for the currently known relations between various complexity measures).

Block sensitivity $\mathsf{bs}(f)$ is a useful intermediate measure that has been used to show polynomial relations between the above measures. *Fractional block sensitivity* $\mathsf{fbs}(f)$ (aka fractional certificate complexity $\mathsf{fC}(f)$, randomized certificate complexity $\mathsf{RC}(f)$ [1]) is a recently introduced measure that is a relaxation of block sensitivity [7]. It has been used to show a tight relation (up to logarithmic factors) between the *zero-error randomized decision tree complexity* $\mathsf{R}_0(f)$ and *two-sided bounded error randomized decision tree complexity* $\mathsf{R}_2(f)$ [6].

The relation between $\mathsf{bs}(f)$ and $\mathsf{fbs}(f)$ has been only partially understood. On one hand, $\mathsf{bs}(f) \leq \mathsf{fbs}(f)$ and this inequality is tight. On the other hand, it

*This work was supported by the European Union Seventh Framework Programme (FP7/2007-2013) under the QALGO (Grant Agreement No. 600700) project and the RAQUEL (Grant Agreement No. 323970) project, the ERC Advanced Grant MQC, and the Latvian State Research Programme NeXIT project No. 1.

is known that $\mathsf{fbs}(f) \leq \mathsf{bs}(f)^2$ but the best known separation gives $\mathsf{fbs}(f) = \left(\frac{1}{3\sqrt{2}} + o(1)\right) \mathsf{bs}(f)^{3/2}$ [4]. We show a family of functions that give a constant factor improvement, $\mathsf{fbs}(f) = \left(\frac{1}{\sqrt{6}} - o(1)\right) \mathsf{bs}(f)^{3/2}$.

2 Definitions

Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function on n variables. We denote the input to f by a binary string $x = (x_1, \ldots, x_n)$, so that the i-th variable is x_i. For an index set $P \subseteq [n]$, let x^P be the input obtained from an input x by flipping every bit x_i, $i \in P$.

We briefly define the notions of sensitivity, certificate complexity and variations on them. For more information on them and their relations to other complexity measures (such as deterministic, probabilistic and quantum decision tree complexities), we refer the reader to the surveys by Buhrman and de Wolf [3] and Hatami et al. [5].

The *sensitivity complexity* $\mathsf{s}(f, x)$ of f on an input x is defined as

$$\mathsf{s}(f, x) = |\{i \in [n] \mid f(x) \neq f(x^{\{i\}})\}|. \tag{1}$$

The *sensitivity* $\mathsf{s}(f)$ of f is defined as $\max_{x \in \{0,1\}^n} \mathsf{s}(f, x)$.

The *block sensitivity* $\mathsf{bs}(f, x)$ of f on an input x is defined as the maximum number t such that there are t pairwise disjoint subsets $B_1, \ldots, B_t$ of $[n]$ for which $f(x) \neq f\left(x^{B_i}\right)$. We call each B_i a *block*. The *block sensitivity* $\mathsf{bs}(f)$ of f is defined as $\max_{x \in \{0,1\}^n} \mathsf{bs}(f, x)$.

The *fractional block sensitivity* $\mathsf{fbs}(f, x)$ of f on an input x is the optimal value of the following linear program, where each sensitive block of x is assigned a real valued weight w_B:

$$\max \sum_{f(x) \neq f(x^B)} w_B \qquad \text{subject to: } \forall i \in [n] : \sum_{B \ni i} w_B \leq 1,$$
$$\forall B : 0 \leq w_B \leq 1.$$

The *fractional block sensitivity* of f is defined as $\mathsf{fbs}(f) = \max_{x \in \{0,1\}^n} \mathsf{fbs}(f, x)$.

A *certificate* C of f is a partial assignment $C : P \to \{0,1\}, P \subseteq [n]$ of the input such that f is constant on this restriction. We call $|P|$ the *length* of C. If f is always 0 on this restriction, the certificate is a *0-certificate*. If f is always 1, the certificate is a *1-certificate*.

The *certificate complexity* $\mathsf{C}(f, x)$ of f on an input x is defined as the minimum length of a certificate that x satisfies. The *certificate complexity* $\mathsf{C}(f)$ of f is defined as $\max_{x \in \{0,1\}^n} \mathsf{C}(f, x)$.

The *fractional certificate complexity* $\mathsf{fC}(f, x)$ of f on an input x is the optimal value of the following linear program, where each position $i \in [n]$ is assigned a real valued weight v_i:

$$\min \sum_{i \in [n]} v_i \qquad \text{subject to: } \forall B \text{ s.t. } f(x) \neq f(x^B) : \sum_{i \in B} v_i \geq 1,$$
$$\forall i \in [n] : 0 \leq v_i \leq 1.$$

The *fractional certificate complexity* of f is defined as $\mathsf{fC}(f) = \max_{x \in \{0,1\}^n} \mathsf{fC}(f, x)$.

For any of these measures $\mathsf{M} \in \{\mathsf{s}, \mathsf{bs}, \mathsf{fbs}, \mathsf{fC}, \mathsf{C}\}$, define $\mathsf{M}_b(f) = \max_{x \in f^{-1}(b)} \mathsf{M}(f, x)$. In that way, we define the measures $\mathsf{s}_0(f), \mathsf{s}_1(f), \mathsf{bs}_0(f), \mathsf{bs}_1(f), \mathsf{fbs}_0(f), \mathsf{fbs}_1(f), \mathsf{fC}_0(f), \mathsf{fC}_1(f)$, $\mathsf{C}_0(f), \mathsf{C}_1(f)$. In particular, $\mathsf{M}(f) = \max\{\mathsf{M}_0(f), \mathsf{M}_1(f)\}$.

One can show that $\mathsf{s}(f) \le \mathsf{bs}(f) \le \mathsf{fbs}(f) \le \mathsf{fC}(f) \le \mathsf{C}(f)$ [7]. In fact, the linear programs of $\mathsf{fbs}(f, x)$ and $\mathsf{fC}(f, x)$ are duals of each other. Therefore, $\mathsf{fbs}(f) = \mathsf{fC}(f)$.

3 Separation

The separation in [4] composes a graph property Boolean function (namely, whether a given graph is a star graph) with the OR function. We build on these ideas and define a new graph property g for the composition that gives a larger separation.

Theorem 1. *There exists a family of Boolean functions such that*

$$\mathsf{fbs}(f) = \left(\frac{1}{\sqrt{6}} - o(1)\right) \mathsf{bs}(f)^{3/2}.$$

Proof. Let $N \ge 12$ be a multiple of 3. An input on $\binom{N}{2}$ variables $(x_{1,2}, x_{1,3}, \ldots, x_{N-1,N})$ encodes a graph G on N vertices. Let $x_{i,j} = 1$ iff the vertices i and j are connected by an edge in G.

We define an auxiliary function $g : \{0,1\}^{\binom{N}{2}} \to \{0,1\}$. Partition $[N]$ into three sets S_0, S_1, S_2 such that $S_r = \{i \in [N] \mid i \equiv r \pmod 3\}$. Let $g(x) = 1$ iff:

- there is some vertex i that is connected to every other vertex by an edge (a star graph);
- for any $r \in \{0,1,2\}$, no two vertices $j, k \ne i$ such that $j, k \in S_r$ are connected by an edge.

Formally, $g(x) = 1$ iff x satisfies one of the following 1-certificates $C_1, \ldots, C_N$: C_i assigns 1 to every edge in $\{x_{j,k} \mid j = i \vee k = i\}$, and assigns 0 to every edge in $\{x_{jk} \mid j \ne i, k \ne i, j \equiv k \pmod 3\}$.

Now we calculate the values of $\mathsf{bs}_0(g), \mathsf{bs}_1(g), \mathsf{fbs}_0(g)$.

- $\mathsf{bs}_0(g) = 3$.

  Consider an input x describing a triangle graph between vertices i, j, k. For this input $g(x) = 0$. Let x' be an input obtained from x by removing the edge $x_{i,j}$ and adding all the missing edges $x_{k,l}$, for all $l \ne i, j$. The corresponding graph is a star graph, therefore, $g(x') = 1$. Let B_k be the sensitive block that flips x to x'. Similarly define B_i and B_j. None of the three blocks overlap, hence $\mathsf{bs}_0(g, x) \ge 3$.

  Now we prove that $\mathsf{bs}_0(g) \le 3$. Assume the contrary, that there exists an input $x \in f^{-1}(0)$ with $\mathsf{bs}(g, x) \ge 4$. Then x has (at least) 4 non-overlapping sensitive blocks $B_1, \ldots, B_4$. Each x^{B_i} satisfies one of the 1-certificates, each a different one. There are 4 such certificates, therefore at least two of them require a star at vertices i, j belonging to the same S_r. The corresponding certificates C_i and C_j both assign 1 at the edge $x_{i,j}$. On the other hand, every other C_k assigns 0 at $x_{i,j}$. Therefore, of the 4 certificates corresponding to $B_1, \ldots, B_4$, two assign 1 to this edge and two assign 0 to this edge. Then, regardless of the value of $x_{i,j}$, we would need to flip it in two of the blocks $B_1, \ldots, B_4$: a contradiction, since the blocks don't overlap. Therefore no such x exists.

– $\mathsf{bs}_1(g) = \frac{N^2}{6} + \frac{N}{6}$.

Examine any 1-certificate C_i. Find three indices $j, k, l \equiv i \pmod 3$ (this is possible, as $N \geq 12$). Any input x that satisfies C_i has $x_{i,j} = x_{i,k} = x_{i,l} = 1$. On the other hand, any other 1-certificate C_t requires at least two of the variables $x_{i,j}, x_{i,k}, x_{i,l}$ to be 0. Hence, the Hamming distance between C_i and C_t is at least two. Therefore, flipping any position of x that is fixed in C_i changes the value of the function as well. Thus, $\mathsf{s}(f, x) = \mathsf{C}(f, x)$. As $\mathsf{s}(f, x) \leq \mathsf{bs}(f, x) \leq \mathsf{C}(f, x)$, we have

$$\mathsf{bs}(f, x) = \mathsf{C}(f, x) = |C_i| = 3\binom{N/3}{2} + \frac{2N}{3} = \frac{N^2}{6} + \frac{N}{6}.$$

– $\mathsf{fbs}_0(g) \geq \frac{N}{2}$.

Examine the all zeros input $0^{\binom{N}{2}}$. Any sensitive block B of this input flips the edges on a star from some vertex. Therefore, any position is flipped by exactly two of the sensitive blocks. The weights $w_B = \frac{1}{2}$ for each sensitive block B then give a feasible solution for the fractional block sensitivity linear program. As there are N sensitive blocks, $\mathsf{fbs}(g, 0^{\binom{N}{2}}) = \frac{N}{2}$.

To obtain the final function we use the following lemma:

Lemma 2 (Proposition 31 in [4])**.** *Let g be a non-constant Boolean function and*

$$f = \mathsf{OR}(g^{(1)}, \ldots, g^{(m)}),$$

an OR *composed with m copies of g. Then for complexity measures $\mathsf{M} \in \{\mathsf{bs}, \mathsf{fbs}\}$, we have*

$$\mathsf{M}_1(f) = \mathsf{M}_1(g)$$
$$\mathsf{M}_0(f) = m \cdot \mathsf{M}_0(g).$$

Let $m = \mathsf{bs}_1(g)/\mathsf{bs}_0(g) = \frac{N^2}{18} + \frac{N}{18}$. Then $\mathsf{bs}(f) = \mathsf{bs}_0(f) = \mathsf{bs}_1(f) = \mathsf{bs}_1(g) = \frac{N^2}{6} + \frac{N}{6}$. On the other hand, $\mathsf{fbs}(f) \geq \mathsf{fbs}_0(f) = m \cdot \mathsf{fbs}_0(g) \geq m \cdot \frac{N}{2} = \frac{N^3}{36} + \frac{N^2}{36}$. Therefore, we have $\mathsf{fbs}(f) \geq \left(\frac{N^2}{6} + \frac{N}{6}\right) \cdot \frac{N}{6} = \mathsf{bs}(f) \cdot \left(\frac{1}{\sqrt{6}} - o(1)\right) \sqrt{\mathsf{bs}(f)} = \left(\frac{1}{\sqrt{6}} - o(1)\right) \mathsf{bs}(f)^{3/2}$.

□

References

[1] Scott Aaronson. Quantum certificate complexity. *Journal of Computer and System Sciences*, 74(3):313–322, 2008.

[2] Scott Aaronson, Shalev Ben-David, and Robin Kothari. Separations in query complexity using cheat sheets. In *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing*, STOC '16, pages 863–876, New York, NY, USA, 2016. ACM.

[3] Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288(1):21–43, 2002.

[4] Justin Gilmer, Michael Saks, and Srikanth Srinivasan. Composition limits and separating examples for some Boolean function complexity measures. *Combinatorica*, 36(3):265–311, 2016.

[5] Pooya Hatami, Raghav Kulkarni, and Denis Pankratov. *Variations on the Sensitivity Conjecture*. Number 4 in Graduate Surveys. Theory of Computing Library, 2011.

[6] Raghav Kulkarni and Avishay Tal. On fractional block sensitivity. *Chicago Journal Of Theoretical Computer Science*, 8:1–16, 2016.

[7] Avishay Tal. Properties and applications of Boolean function composition. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 441–454, New York, NY, USA, 2013. ACM.

New Size Hierarchies for Two Way Automata*

Kamil Khadiev[1,2] Rishat Ibrahimov[2] Abuzer Yakaryılmaz[1]

[1]Faculty of Computing, University of Latvia
Raiņa bulv. 19, Rīga, LV-1586, Latvia
[2]Kazan Federal University, Krmlevskaya 18, Kazan, 420008, Russia
kamilhadi@gmail.com, rishat.ibrahimov@yandex.ru, abuzer@lu.lv

Abstract

We introduce a new type of nonuniform two–way automaton that can use a different transition function for each tape square. We also enhance this model by allowing to shuffle the given input at the beginning of the computation. Then we present some hierarchy and incomparability results on the number of states for the types of deterministic, nondeterministic, and bounded-error probabilistic models. For this purpose, we provide some lower bounds for all three models based on the numbers of subfunctions and we define two witness functions.
Keywords: Two–way nonuniform automaton, size hierarchy, deterministic and nondeterministic models, probabilistic computation

1 Introduction

Nonuniform models (like circuits, branching programs, uniform models using advice, etc.) have played significant roles in computational complexity, and, naturally they have also been investigated in automata theory (e.g. [7, 12, 14, 6]). The main computational resource for nonuniform automata is the number of internal states that depends on the input size. Thus we can define linear, polynomial, or exponential size automata models. In this way, for example, nonuniform models allow us to formulate the analog of "P versus NP problem" in automata theory: Sakoda and Sipser [24] conjectured that simulating a two–way nondeterministic automaton by two–way deterministic automata requires exponential number of states in the worst case. But, the best known separation is only quadratic ($O(n^2)$) [9, 13] and the researchers have succeeded to obtain slightly better bounds only for some modified models, e.g. [21, 15, 11, 16].

In this paper, we mainly present some hierarchy results for deterministic, nondeterministic, and bounded-error probabilistic nonuniform two–way automata models, which can also be seen as a "two–way" version of ordered binary decision diagrams (OBDDs) [26]. For each input length (n), our models can have different number of states, and, like Branching programs or the data-independent models defined by Holzer [12], the transition functions can be changed during the computation.

*A preliminary version of this paper was presented in NCMA2014 (6th Workshop on Non-Classical Models of Automata and Applications) as a short paper.

Holzer's model can use a different transition function for each step. We restrict this property so that the transition function is the same for the same tape positions, and so, we can have at most n different transition functions. Moreover, we enhance our models by shuffling the input symbols at the beginning of the computation. We give the definitions and related complexity measures in Section 2. Hierarchies for related model OBDD was explored in [3],[17], [8], [18], [20], [4], [2], [1], [19].

In order to obtain our main results, we start with presenting some generic lower bounds (Section 3), which are follow [25] and [10]. Then, we define two witness Boolean functions in Section 3.2.1: *Shuffled Address Function*, denoted $\texttt{2-SAF}_\texttt{t}$, which is a modification of Boolean functions given in [22],[8],[17], [5], [13], and its uniform version $\texttt{2-USAF}_\texttt{t}$. Moreover, regarding these functions, we provide two deterministic algorithms. In our results, we also use the well known *Equality function* $\mathsf{EQ(X)} = \bigvee_0^{\lfloor n/2 \rfloor - 1} x_i = x_{i+\lfloor n/2 \rfloor}$.

In Sections 4 and 5, we present our main results based on the size (the number of states) of models. We obtain linear size separations for deterministic models and quadratic size separations for nondeterministic and probabilistic models. Moreover, we investigate the effect of shuffling for all three types of models, and, we show that in some cases shuffling can save huge amount of states and in some other cases shuffling cannot be size efficient. We also show that the constant number of states does not increase the computational power of deterministic and nondeterministic nonuniform models without shuffling.

All missing proofs are given in Appendices, which are referred just before the statements.

2 Definitions

Our alphabet is binary, $\Sigma = \{0, 1\}$. We mainly use the terminologies of Branching programs: Our decision problems are solving/computing Boolean functions: The automaton solving a function accepts the inputs where the function gets the value true and rejects the inputs where the function gets the value false. For uniform models, on the other hand, our decision problems are recognizing languages: The automaton recognizing a language accepts any member and rejects any non-members.

A nonuniform head–position–dependent two–way deterministic automaton working on the inputs of length/size $n \geq 0$ (2DA_n) D_n is a 6-tuple $D_n = (\Sigma, S, s_1, \delta = \{\delta_1, \ldots, \delta_n\}, s_a, s_r)$, where (i) $S = \{s_1, \ldots, s_d\}$ is the set of states (d can be a function in n) and $s_1, s_a, s_r \in S$ ($s_a \neq s_r$) are the initial, accepting, and rejecting states, respectively; and, (ii) δ is a collection of n transition functions such that $\delta_i : S \setminus \{s_a, s_r\} \times \Sigma \to S \times \{\leftarrow, \downarrow, \rightarrow\}$ is the transition function that governs behaviour of D_n when reading the ith symbol/variable of the input, where $1 \leq i \leq n$. Any given input $u \in \Sigma^n$ is placed on a read-only tape with a single head as $u_1 u_2 \cdots u_n$ from the squares 1 to $|u| = n$. When D_n is in $s \in S \setminus \{s_a, s_r\}$ and reads $u_i \in \Sigma$ (the ith symbol of u) on the tape, it switches to state $s' \in S$ and updates the head position with respect to $a \in \{\leftarrow, \downarrow, \rightarrow\}$ if $\delta_i(s, u_i) \to (s', a)$. If $a =$ " $\leftarrow$ " (" $\rightarrow$ "), the head moves one square to the left (the right), and, it stays on the same square, otherwise. The transition functions δ_1 and δ_n must be defined to guarantee that the head never leaves u during the computation. Moreover, the automaton enters s_a or s_r only on the right most symbol and then the input is accepted or rejected, respectively.

The nondeterministic counterpart of $2DA_n$, denoted $2NA_n$, can choose from more than one transition in each step. So, the range of each transition function is $\mathcal{P}(S \times \{\leftarrow, \downarrow, \rightarrow\})$, where $\mathcal{P}(\cdot)$ is the power set of any given set. Therefore, a $2NA_n$ can follow more than one computational path and the input is accepted only if one of them ends with the decision of "acceptance". Note that some paths end without any decision since the transition function can yield the empty set for some transitions.

The probabilistic counterpart of $2DA_n$, denoted $2PA_n$, is a $2NA_n$ such that each transition is associated with a probability. Thus, $2PA_n$s can be in a probability distribution over the deterministic configurations (the state and the position of head forms a configuration) during the computation. To be a well-formed machine, the total probability must be 1, i.e. the probability of outgoing transitions from a single configuration must be always 1. Thus, each input is accepted and rejected by a $2PA_n$ with some probabilities. An input is said to be accepted/rejected by a (bounded-error) $2PA_n$ if the accepting/rejecting probability by the machine is at least $1/2 + \varepsilon$ for some $\varepsilon \in (0, 1/2]$.

A function $f_n : \Sigma^n \to \Sigma$ is said to be computed by a $2DA_n$ D_n (a $2NA_n$ N_n, a $2PA_n$ P_n) if each member of $f_n^{-1}(1)$ is accepted by D_n (N_n, P_n) and each member of $f_n^{-1}(0)$ is rejected by D_n (N_n, P_n). The class $\mathsf{2DSIZE(d(n))}$ is formed by the functions $f = \{f_0, f_1, f_2, \ldots\}$ such that each f_i is computed by a $2DA_i$ D_i, the number of states of which is no more than $d(i)$, where i is a non-negative integer. We can similarly define nondeterministic and probabilistic counterparts of this class, denoted $\mathsf{2NSIZE(d(n))}$ and $\mathsf{2PSIZE(d(n))}$ respectively.

We also introduce a generalization of our nonuniform models that can shuffle the input at the beginning of the computation with respect to a permutation. A nonuniform head–position–dependent *shuffling* two–way deterministic automaton working on the inputs of length/size of $n \geq 0$ ($2DA_n^{\Theta}$), say D_n^{θ}, is a $2DA_n$ that shuffles the symbols of input with respect to θ, a permutation of $\{1, \ldots, n\}$, i.e. the j-th symbol of the input is placed on $\theta(j)$-th place on the tape ($1 \leq j \leq n$), and then execute the $2DA_n$ algorithm on this new input. The nondeterministic and probabilistic models can be respectively abbreviated as $2NA_n^{\Theta}$ and $2PA_n^{\Theta}$. The class $\mathsf{2D\Theta SIZE(d(n))}$ is formed by the functions $f = \{f_0, f_1, f_2, \ldots\}$ such that each f_i is computed by a $2DA_i^{\Theta}$ D_i^{θ} whose number of states is no more than $d(i)$, where i is a non-negative integer and θ is a permutation of $\{1, \ldots, n\}$. The nondeterministic and probabilistic classes are respectively represented by $\mathsf{2N\Theta SIZE(d(n))}$ and $\mathsf{2P\Theta SIZE(d(n))}$.

Moreover we consider uniform versions of two-way automata, respectively 2DFA and 2NFA. We can define 2DFA in the same way as $2DA_n$, but it is identical for all n, $|S| = const$ and $\delta_i = \delta$ for any $i \in \{1, \ldots, n\}$. Moreover, they can use end-markers, between which the given input is placed on the input tape. We can define 2NFA similarly. The corresponding classes of languages defined by 2DFAs and 2NFAs of size d are denoted $\mathsf{2DFASIZE(d)}$ and $\mathsf{2NFASIZE(d)}$, respectively.

3 Lower bounds, Boolean functions, and algorithms

Our key complexity measure behind our results is the number of subfunctions for a given function. It can be seen as the counterpart of "the equivalence classes of a language" with respect to Myhill-Nerode Theorem [23].

Let f be a Boolean function defined on $X = \{x_1, \ldots, x_n\}$. We define the set of all permutations of $(1, \ldots, n)$ as $\Theta(n)$. Let $\theta \in \Theta(n)$ be a permutation. We can order the elements of X with respect to θ, say $(x'_1, \ldots, x'_n)$, and then we can split them into two disjoint non-empty (ordered) sets by picking an index $i \in \{1, \ldots, n-1\}$: $X_A = (x'_1, \ldots, x'_i)$ and $X_B = (x'_{i+1}, \ldots, x'_n)$. Let ρ be a mapping $\{x'_1, \ldots, x'_i\} \to \{0,1\}^i$ that assigns a value to each $x'_j \in (x'_1, \ldots, x'_i)$. Then, we define function $f|_\rho : X_B \to \{0,1\}$ that returns the value of f where the values of the input from X_A are fixed by ρ. The function $f|_\rho$ is called a subfunction.

The total number of different subfunctions with respect to θ and i is denoted by $N_i^\theta(f)$. Then, we focus on the maximum value by considering all possible indices:

$$N^\theta(f) = \max_{i \in \{1,\ldots,n-1\}} N_i^\theta(f).$$

After this, we focus on the best permutation that minimizes the number of subfunctions:

$$N(f) = \min_{\theta \in \Theta(n)} N^\theta(f).$$

Now, we represent the relation between the number of subfunctions for Boolean function and the number of equivalence classes of a language.

Let L be a language defined on $\Sigma = \{0,1\}$. For a given non-negative integer n, L_n is the language composed by all members of L with length n, i.e. $L_n = L \cap \Sigma^n = \{w \mid w \in L, |w| = n\}$. For $r < n$, two strings $u \in \Sigma^r$ and $v \in \Sigma^r$ are said to be equivalent if for any $y \in \Sigma^{n-r}$, $uy \in L_n$ if and only if $vy \in L_n$.

We denote the number of non-equivalent strings of length r as $R^r(L_n)$. Then, similar to the number of subfunctions,

$$R(L_n) = \max_{r \in \{1,\ldots,n-1\}} R^r(L_n) \text{ and } R_n(L) = R(L_n).$$

The function $f_n^L(X)$ denotes the characteristic Boolean function for language L_n. Thus, we can say that

$$N^{id}(f_n^L) = R_n(L).$$

for $id = (1, \ldots, n)$ is natural order.

3.1 Lower bounds

First we give our lower bounds on the sizes of models in terms of $N(f)$. Note that all of lower bounds for shuffling models are valid also for non-shuffling models, but in that case lower bounds for $N^{id}(f)$ not for $N(f)$.

Theorem 1. *(Appendix A) If the function $f(X)$ is computed by a $2DA_n^\Theta$ of size d for some permutations θ, then*

$$N(f) \leq (d+1)^{d+1}.$$

Corollary 1. *If the language L is recognized by a 2DFA of size d, then*

$$R_n(L) \leq (d+1)^{d+1},$$

Theorem 2. *(Appendix B) If the function $f(X)$ is computed by a $2NA_n^\Theta$ of size d for some permutations θ, then*

$$N(f) \leq 2^{(d+1)^2}.$$

Corollary 2. *If the language L is recognized by a 2NFA of size d, then*

$$R_n(L) \leq 2^{(d+1)^2}.$$

Based on Theorems 1 and 2, we can obtain the following result.

Theorem 3. *For constant integer d,* $\mathsf{2DSIZE(d)}$ *and* $\mathsf{2NSIZE(d)}$ *contain only characteristic functions of regular languages.*

Proof. If d is constant and $2DA_n$ (or $2NA_n$) A_n computes f_n then $N(f_n)$ is constant and it is same for each n. Hence the $R(L_n)$ of the corresponding language L_n is constant too, so the language $L = \bigcup L_n$ is regular, because it have constant number of "the equivalence classes of a language" with respect to Myhill-Nerode Theorem [23]. □ □

Theorem 4. *(Appendix C) If the function $f(X)$ is computed by a $2PA_n^{\Theta}$ of size d for some permutations θ, with expected running time to finish computation T and error probability ε, then*

$$N(f) \leq \left\lceil \frac{4d\,(8 + 3\log T)}{\log\,(1+2\varepsilon)\,(1+\varepsilon)} \right\rceil^{(d+1)^2}.$$

Corollary 5. *If the function $f(X)$ is computed by a $2PA_n^{\Theta}$ of size d for some permutations θ, with expected running time to finish computation $T \geq 256$ and error probability $\varepsilon = \frac{1}{5}$, then*

$$N(f) \leq (32d \log T)^{(d+1)^2}.$$

Proof. For $\varepsilon = \frac{1}{5}$, $\log\,(1+2\varepsilon)\,(1+\varepsilon) > \frac{1}{2}$, and, for $T \geq 256$, $\log T \geq 8$. □ □

3.2 Boolean Functions

We define two Boolean functions: (1) A modification of Boolean function given in [3, 17, 22, 8, 13] *Shuffled Address Function*, denoted $\mathtt{2\text{-}SAF}_t$, and (2) *Uniform Shuffled Address Function* $\mathtt{2\text{-}USAF}_t$ as a modification of $\mathtt{2\text{-}SAF}_t$. We also use the language $\mathtt{L2USAF}_t$, the characteristic function of which is $\mathtt{2\text{-}USAF}_t$.

3.2.1 Boolean Function $\mathtt{2\text{-}SAF}_t$:

We divide all input into two parts, and each part into t blocks. Each block has *address* and *value*. Formally, Boolean function $\mathtt{2\text{-}SAF}_t(X) : \{0,1\}^n \to \{0,1\}$ for integer $t = t(n)$ such that

$$2t(2t + \lceil \log 2t \rceil) < n. \tag{1}$$

We divide the input variables (the symbols of the input) into $2t$ blocks. There are $\lfloor \frac{n}{2t} \rfloor = q$ variables in each block. After that, we divide each block into *address* and *value* variables (see Figure 1). The first $\lceil \log 2t \rceil$ variables of block are *address* and the other $q - \lceil \log 2t \rceil = b$ variables of block are *value*. We call $x_0^p, \ldots, x_{b-1}^p$ and $y_0^p, \ldots, y_{\lceil \log 2w \rceil}^p$ are the *value* and the *address* variables of the pth block, respectively, for $p \in \{0, \ldots, 2t-1\}$.

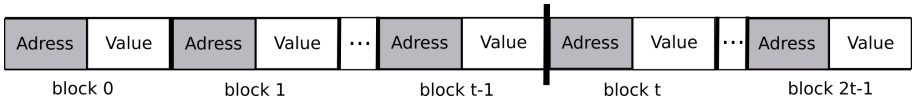


Figure 1: *Address* and *value* bits of blocks.

Function $\texttt{2-SAF}_t(X)$ is calculated based on the following five sub-functions:

1. $Adr : \{0,1\}^n \times \{0,\ldots,2t-1\} \to \{0,\ldots,2t-1\}$ gets the address of a block:
$$Adr(X,p) = \sum_{j=0}^{\lceil \log 2t \rceil - 1} y_j^p \cdot 2^j (mod\ 2t).$$
2. $Ind : \{0,1\}^n \times \{0,\ldots,2t-1\} \to \{-1,\ldots,2t-1\}$ gets the number of block by address:
$$Ind(X,a) = \begin{cases} p & , \text{ where } p \text{ is the minimal number such that } Adr(X,p) = a, \\ -1 & , \text{ if there are no such } p \end{cases}.$$
3. $Val : \{0,1\}^n \times \{0,\ldots,2t-1\} \to \{-1,\ldots,t-1\}$ gets the value of the block with address i:
$$Val(X,a) = \begin{cases} \sum_{j=0}^{b-1} x_j^p (mod\ t) & , \text{ where } p = Ind(X,a) \text{ for } p \geq 0, \\ -1 & , \text{ if } Ind(X,a) < 0 \end{cases}.$$

Suppose that we are at the i-th step of iteration.

4. $Step_1 : \{0,1\}^n \times \{0,\ldots,1\} \to \{-1,t\ldots,2t-1\}$ gets the first part of the ith step of iteration:
$$Step_1(X,i) = \begin{cases} -1 & , \text{ if } Step_2(X,i-1) = -1, \\ Val(X, Step_2(X,i-1)) + t & , \text{ otherwise} \end{cases}.$$
5. $Step_2 : \{0,1\}^n \times \{-1,\ldots,1\} \to \{-1,\ldots,t-1\}$ gets the second part of the ith step of iteration:
$$Step_2(X,i) = \begin{cases} -1 & , \text{ if } Step_1(X,i) = -1, \\ 2 & , \text{ if } i = -1 \\ Val(X, Step_1(X,i)) & , \text{ otherwise} \end{cases}.$$

Function $\texttt{2-SAF}_t(X)$ is computed iteratively: $\texttt{2-SAF}_t(X) = \begin{cases} 0, & \text{if } Step_2(X,1) \leq 0, \\ 1, & \text{otherwise} \end{cases}$:

1. We find the block with address 2 in the first part and compute the value of this block, which is the address of the block for the second part.
2. We take the block from the second part with the computed address and compute value of the block, which is the address of the new block for the first part.
3. We find the block with new address in the second part and check value of this block. If value greats 0, then value of $\texttt{2-SAF}_t$ is 1, and 0 otherwise.

If we do not find block with searching address n in any phase then value of $\texttt{2-SAF}_t$ is also 0. See the Figure 2 for the iterations of the function.

Theorem 6. *(Appendix D) For integer $t = t(n)$, $N(\texttt{2-SAF}_t) \geq t^{t-2}$, where t satisfies $2t(2t + \lceil \log 2t \rceil) < n$.*

Theorem 7. *(Appendix E) There is a $2DA_n$ of size $13t + 4$ that computes $\texttt{2-SAF}_t$.*

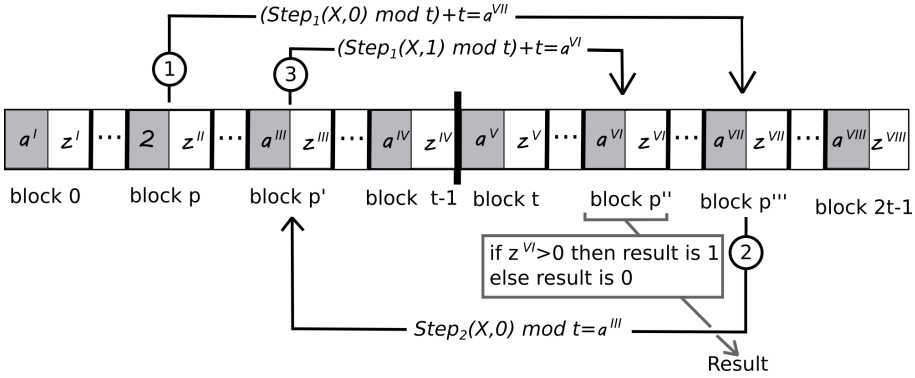


Figure 2: 2-$\mathtt{SAF}_t(X)$, $Step_1(X,i)$ and $Step_2(X,i)$ functions. In the picture we write a in address bits of p-th block (gray box) if $A(X,p)=a$.

3.2.2 Boolean Function 2-$\mathtt{USAF}_t$:

The definition of 2-$\mathtt{USAF_t}$ is as follows:

$$\text{2-}\mathtt{USAF}_t(X):\{0,1\}^n\to\{0,1\} \text{ for integer } t=t(n) \text{ satisfying that}$$

$$4t(2t+\lceil\log 2t\rceil)<n. \tag{2}$$

We denote its language version as $\mathtt{L2USAF}_t$.

We divide the input variables (the symbols of the input) into $2t$ blocks. There are $\lfloor\frac{n}{2t}\rfloor=q$ variables in each block. After that, we divide each block into *mark*, *address* and *value* variables. All variables that are in odd positions are *mark*. The type of the bit on an even position is determined by the value of previous *mark* bit. The bit is *address*, if the previous *mark* bit's value is 0, and *value* otherwise. The first $\lceil\log 2t\rceil=c$ variables of block that are in the even positions denote the *address* and the other $q/2-\lceil\log 2t\rceil=b$ variables of block denote the *value*.

We call $x_0^p,\ldots,x_{b-1}^p$, $y_0^p,\ldots,y_{\lceil\log 2t\rceil}^p$ and $z_0^p,\ldots,z_{q/2}^p$ are the *value*, *address* and the *mark* variables of the pth block, respectively, for $p\in\{0,\ldots,2t-1\}$.

Function 2-$\mathtt{USAF}_t(X)$ is calculated based on the following five sub-functions:

1. $Adr:\{0,1\}^n\times\{0,\ldots,2t-1\}\to\{0,\ldots,2t-1\}$ gets the address of a block:

$$Adr(X,p)=\sum_{j=0}^{c-1}y_j^p\cdot 2^{c-j-1}(mod\ 2t).$$

2. $Ind:\{0,1\}^n\times\{0,\ldots,2t-1\}\to\{-1,\ldots,2t-1\}$ gets the number of block by address:

$$Ind(X,a)=\begin{cases} p & , \text{ where } p \text{ is the minimal number such that } Adr(X,p)=a,\\ -1 & , \text{ if there are no such } p\end{cases}.$$

3. $Val:\{0,1\}^n\times\{0,\ldots,2t-1\}\to\{-1,\ldots,t-1\}$ gets the value of the block with address i:

$$Val(X,a)=\begin{cases}\sum_{j=0}^{b-1}x_j^p(mod\ t) & , \text{ where } p=Ind(X,a) \text{ for } p\geq 0,\\ -1 & , \text{ if } Ind(X,i)<0\end{cases}.$$

Suppose that we are at the i-th step of iteration.

4. $Step_1 : \{0,1\}^n \times \{0,\ldots,1\} \to \{-1, t \ldots, 2t-1\}$ gets the first part of the ith step of iteration:

$$Step_1(X,i) = \begin{cases} -1 & , \text{ if } Step_2(X,i-1) = -1, \\ Val(X, Step_2(X,i-1)) + t & , \text{ otherwise} \end{cases}.$$

5. $Step_2 : \{0,1\}^n \times \{-1,\ldots,1\} \to \{-1,\ldots,t-1\}$ gets the second part of the ith step of iteration:

$$Step_2(X,i) = \begin{cases} -1 & , \text{ if } Step_1(X,i) = -1, \\ 2 & , \text{ if } i = -1 \\ Val(X, Step_1(X,i)) & , \text{ otherwise} \end{cases}.$$

Remark that the address of the current block is computed on the previous step. Function $\texttt{2-USAF}_t(X)$ is computed as: $\texttt{2-USAF}_t(X) = \begin{cases} 0, & \text{if } Step_2(X,1) \leq 0, \\ 1, & \text{otherwise} \end{cases}$.

Theorem 8. *(Appendix F) For integer $t = t(n)$, $R_n(\texttt{L2USAF}_t) \geq t^{t-2}$, where t satisfies $4t(2t + \lceil \log 2t \rceil) < n$.*

Theorem 9. *(Appendix G) There is a $2DFA$ of size $23t + 2(1+3t)\log t + 6$ recognizing $\texttt{L2USAF}_t$. For simplicity, we can also use the followings:*

- *For $t > 2906$, there is a $2DFA$ of size $8t\log(t)$ recognizing $\texttt{L2USAF}_t$.*
- *For $t > 26$, there is a $2DFA$ of size $11t\log(t)$ recognizing $\texttt{L2USAF}_t$.*
- *For $t > 3$, there is a $2DFA$ of size $19t\log(t)$ recognizing $\texttt{L2USAF}_t$.*

4 Hierarchies results

Now, we can follow our hierarchies and incomparability results.

Theorem 10. *Let $d : \mathbb{N} \to \mathbb{N}$ be a function satisfying $13d + 43 < \sqrt{\frac{n}{6}}$. Then,* $\mathsf{2D\Theta SIZE}(\mathrm{d}) \subsetneq \mathsf{2D\Theta SIZE}(13\mathrm{d} + 43)$.

Proof. By Theorem 6, we can follow that $(d+1)^{d+1} < (d+3)^{d+1} \leq N(\texttt{2-SAF}_{d+3})$. Suppose that there is a $2DA^{\Theta}_n$ with size d computing $\texttt{2-SAF}_{d+3}$. Then, by Theorem 1, we can have $N(\texttt{2-SAF}_{d+3}) \leq (d+1)^{d+1}$, which is a contradiction and so $\texttt{2-SAF}_{d+3} \notin \mathsf{2D\Theta SIZE}(\mathrm{d})$.

Moreover, by Theorem 7, we have that $\texttt{2-SAF}_{d+3}$ is in $\mathsf{2DSIZE}(13\mathrm{d} + 43)$.

Lastly, the relating between d and n can be followed from the definition of $\texttt{2-SAF}_d$ by also taking into account the parameter $(13d + 43)$. For $\texttt{2-SAF}_t$, we have inequality $2t(2t + \lceil \log(2t) \rceil) < n$. For $t \geq 2$, we always have $t \geq \lceil \log(2t) \rceil$. Therefore, the following inequality also works $2t(3t) < n$, which gives us that $t < \sqrt{\frac{n}{6}}$. □ □

Theorem 11. *Let $d : \mathbb{N} \to \mathbb{N}$ be a function satisfying $121 < 13d(n) + 4 < \sqrt{\frac{n}{6}}$. Then,* $\mathsf{2N\Theta SIZE}(\lfloor\sqrt{\mathrm{d}}\rfloor) \subsetneq \mathsf{2N\Theta SIZE}(13\mathrm{d} + 4)$.

Proof. By Theorem 6 we have $2^{\log_2 d(d-2)} = d^{d-2} \leq N(\texttt{2-SAF}_d)$. For $d > 9$, we can follow that

$$2^{(\sqrt{d}+1)^2} < 2^{\log_2 d(d-2)}.$$

Suppose that there is a $2NA^{\Theta}_n$ with size $\lfloor\sqrt{d}\rfloor$ computing $\texttt{2-SAF}_d$. Then, by Theorem 2, we can have $N(\texttt{2-SAF}_d) \leq 2^{(\lfloor\sqrt{d}\rfloor+1)^2}$, which is a contradiction and so $\texttt{2-SAF}_d \notin \mathsf{2N\Theta SIZE}(\lfloor\sqrt{\mathrm{d}}\rfloor)$.

Moreover, by Theorem 7, we have that $\texttt{2-SAF}_d$ is in $\mathsf{2DSIZE}(13\mathrm{d}+4)$.

The relating between d and n can be followed similar to the previous proof and the condition $d > 9$. □ □

Theorem 12. *Let* $d : \mathbb{N} \to \mathbb{N}$ *be a function satisfying* $1330 < \lceil 11d(n)\log d(n)\rceil < \sqrt{\frac{n}{12}}$. *Then,* $\mathsf{2DFASIZE}(\mathrm{d}-3) \subsetneq \mathsf{2DFASIZE}(\lceil 11\mathrm{d}\log\mathrm{d}\rceil)$.

Proof. By Theorem 8, we have $(d-2)^{d-2} < d^{d-2} \leq N(\texttt{2-USAF}_d)$ and so we can also have

$$(d-2)^{d-2} < R_n(\texttt{L2USAF}_d).$$

Suppose that there is a 2DFA with size $d-3$ recognizing $\texttt{L2USAF}_d$. Then, by Corollary 1, we can have $R_n(\texttt{L2USAF}_d) \leq (d-2)^{d-2}$, which is a contradiction and so $\texttt{L2USAF}_d \notin \mathsf{2DFASIZE}(\mathrm{d}-3)$.

By Theorem 9 and for $d > 26$, we have $\texttt{L2USAF}_d \in \mathsf{2DFASIZE}(\lceil 11\mathrm{d}\log\mathrm{d}\rceil)$.

The relating between d and n can be followed by the definition of $\texttt{2-USAF}_t$: $4t(2t+\lceil\log(2t)\rceil) < n$. For $t \geq 2$, $t \geq \lceil\log(2t)\rceil$ and so we can also use the inequality $4t(3t) < n$, which implies $t \leq \sqrt{\frac{n}{12}}$. □ □

Theorem 13. *Let* $d : \mathbb{N} \to \mathbb{N}$ *be a function satisfying* $1330 < \lceil 11d\log d\rceil < \sqrt{\frac{n}{12}}$. *Then,* $\mathsf{2NFASIZE}(\lfloor\sqrt{\mathrm{d}}\rfloor) \subsetneq \mathsf{2NFASIZE}(\lceil 11\mathrm{dlogd}\rceil)$.

Proof. By using the facts given in the two above proofs, for $d > 9$, we know that $\texttt{L2USAF}_d \notin \mathsf{2NFASIZE}(\lfloor\sqrt{\mathrm{d}}\rfloor)$ and $\texttt{L2USAF}_d \in \mathsf{2DFASIZE}(\lceil 11\mathrm{d}\log\mathrm{d}\rceil)$. □ □

Theorem 14. *Let* $d : \mathbb{N} \to \mathbb{N}$ *be a function satisfying* $30 < 13d(n)+4 < \sqrt{\frac{n}{6}}$ *and* $T \geq 256$ *be the expected running time to finish computation. Then*

$$\mathsf{2B\Theta SIZE}\left(\left\lfloor\frac{\sqrt{\mathrm{d}}}{32\log\mathrm{T}}\right\rfloor\right) \subsetneq \mathsf{2B\Theta SIZE}\,(13\mathrm{d}+4),$$

where the error bound is at least $\frac{1}{5}$ *for the classes.*

Proof. By using Theorem 6, we can obtain the following for $d > 2$.

$$\left(\frac{32\log(T)\sqrt{d}}{32\log(T)}\right)^{\left(\frac{\sqrt{d}}{32\log T}+1\right)^2} < \left(\frac{32\log(T)d}{32\log(T)}\right)^{d-2} = d^{d-2} \leq N(\texttt{2-SAF}_d).$$

Then $\texttt{2-SAF}_d$ cannot be solved by a $2PA_n^{\Theta}$ with size $\lfloor\frac{\sqrt{d}}{32\log T}\rfloor$. Otherwise, by Corollary 5, we obtain the following contradiction.

$$N(\texttt{2-SAF}_d) < \left(32\log T\left\lfloor\frac{\sqrt{d}}{32\log T}\right\rfloor\right)^{\left(\lfloor\frac{\sqrt{d}}{32\log T}\rfloor+1\right)^2}.$$

By Theorem 7, we have that $\texttt{2-SAF}_d$ is in $\mathsf{2DSIZE}(13\mathrm{d}+4)$.

The relating between d and n can be followed similar to the previous proofs and the condition $d > 2$. □ □

5 Incomparability results

In this section, we give evidences how shuffling can reduce the size of models. For this purpose, we use the well-known *Equality function*

$$\texttt{EQ}(X) = \bigvee_{0}^{\lfloor n/2 \rfloor - 1} x_i = x_{i+\lfloor n/2 \rfloor}.$$

The nice property of $\texttt{EQ}(X)$ for our purpose is that $N^{id}(\texttt{EQ}) = 2^{\lfloor n/2 \rfloor}$ and $N^{\theta}(\texttt{EQ}) \leq 4$ for $\theta = (0, \lfloor n/2 \rfloor, 1, \lfloor n/2 \rfloor + 1, \ldots)$. Therefore, we can follow that $\texttt{EQ} \in \mathsf{2D\Theta SIZE}(4)$.

Suppose that there is a $2NA_n$ with size $\lfloor \sqrt{\frac{n}{2}} \rfloor - 2$ solving EQ. Then, by Theorem 2, we have

$$N^{id}(\texttt{EQ}) \leq 2^{(\lfloor \sqrt{\frac{n}{2}} \rfloor - 1)^2},$$

which is a contradiction. Therefore, $\texttt{EQ} \notin \mathsf{2NSIZE}(\lfloor \sqrt{\frac{n}{2}} \rfloor - 2)$. We use this lower bound also for deterministic case since trying to find a better bound in a nice form for the class $\mathsf{2DSIZE}(\cdot)$ is not easy.

Let $d = d(n)$ be a function such that $d \leq \lfloor \sqrt{\frac{n}{2}} \rfloor - 2$, and d' be a function such that $4 \leq d' < d$. Then, due to function EQ, we can immediately follow that

$$\mathsf{2D\Theta SIZE}(\mathrm{d}') \notin \mathsf{2DSIZE}(\mathrm{d}) \quad \text{and} \quad \mathsf{2N\Theta SIZE}(\mathrm{d}') \notin \mathsf{2NSIZE}(\mathrm{d}).$$

On the other hand, due to the results given in Section 4, the shuffling classes cannot contain the non-shuffling classes under certain size bounds, which lead us to our incomparability results.

Theorem 15. *As a further restriction, if* $13d+43 < \sqrt{\frac{n}{6}}$*, then for* $94 \leq 13d'+43 < d$*, the classes* $\mathsf{2D\Theta SIZE}(\mathrm{d}')$ *and* $\mathsf{2DSIZE}(\mathrm{d})$ *are not comparable.*

Proof. By the proof of Theorem 10, we know that $\mathsf{2D\Theta SIZE}(\mathrm{d}')$ does not have a function that is in $\mathsf{2DSIZE}(13\mathrm{d}' + 43)$. Since $13d' + 43 < d$, this function is also in $\mathsf{2DSIZE}(\mathrm{d})$. □ □

Theorem 16. *As a further restriction, if* $121 < 13d + 4 < \sqrt{\frac{n}{6}}$*, then for* $58 \leq 13d' + 4 < d$*, the classes* $\mathsf{2N\Theta SIZE}(\lfloor \sqrt{\mathrm{d}'} \rfloor)$ *and* $\mathsf{2NSIZE}(\mathrm{d})$ *are not comparable.*

Proof. By the proof of Theorem 11, we know that $\mathsf{2N\Theta SIZE}(\lfloor \sqrt{\mathrm{d}'} \rfloor)$ does not have a function that is in $\mathsf{2DSIZE}(13\mathrm{d}' + 4)$. Since $13d' + 4 < d$, this function is also in $\mathsf{2NSIZE}(\mathrm{d})$. □ □

Any $2PA_n$ with size d satisfying $(32d \log T)^{(d+1)^2} < 2^{\lfloor \frac{n}{2} \rfloor}$ cannot solve EQ due to Theorem 4. We bound $T < 2^{2^d}$. Then, for $d \geq 2$, we can follow

$$(32d \log T)^{(d+1)^2} < \left(2^5 2^{\log_2 d} 2^d\right)^{(d+1)^2} \leq \frac{2^{d^3}}{2} < \frac{2^{\frac{n}{2}}}{2} < 2^{\lfloor \frac{n}{2} \rfloor}.$$

Then, for $4 \leq d' \leq d$ where $d < \sqrt[3]{\frac{n}{2}}$, $\mathsf{2B\Theta SIZE}\,(\mathrm{d}') \notin \mathsf{2BSIZE}(\mathrm{d})$. Now, we can state the result for probabilistic case similar to the other cases.

Theorem 17. *As a further restriction, if* $30 < 13d + 4 < \sqrt[3]{\frac{n}{2}}$*, then for* $4 \leq 13d' + 4 < d$ *and* $256 \leq T < 2^{2^{d'}}$ *be the expected running time to finish computation, the classes* $\mathsf{2B\Theta SIZE}\left(\left\lfloor \frac{\sqrt{\mathrm{d}'}}{32 \log \mathrm{T}} \right\rfloor\right)$ *and* $\mathsf{2BSIZE}(\mathrm{d})$ *are not comparable.*

Acknowledgements. We thank Farid Ablayev and Aida Gainutdinova for their helpful comments on the models. We also thank anonymous reviewers for their valuable comments. The work is partially supported by ERC Advanced Grant MQC. The work is performed according to the Russian Government Program of Competitive Growth of Kazan Federal University

References

[1] F. Ablayev, A. Gainutdinova, K. Khadiev, and A. Yakaryılmaz. Very narrow quantum obdds and width hierarchies for classical obdds. *Lobachevskii Journal of Mathematics*, 37(6):670–682, 2016.

[2] Farid Ablayev, Aida Gainutdinova, Kamil Khadiev, and Abuzer Yakarylmaz. Very narrow quantum obdds and width hierarchies for classical obdds. In *DCFS*, volume 8614 of *LNCS*, pages 53–64. Springer, 2014.

[3] Farid Ablayev and Kamil Khadiev. Extension of the hierarchy for k-OBDDs of small width. *Russian Mathematics*, 53(3):46–50, 2013.

[4] Farid Ablayev, Kamil Khadiev, and Aliya Khadieva. Lower bound and hierarchies for quantum ordered read-k-times branching programs. *arXiv preprint arXiv:1703.05015*, 2017. Accepted by CSR 2017.

[5] Farid M. Ablayev and Marek Karpinski. On the power of randomized branching programs. In *ICALP*, volume 1099 of *LNCS*, pages 348–356. Springer, 1996.

[6] Kaspars Balodis. One alternation can be more powerful than randomization in small and fast two-way finite automata. In *FCT*, volume 8070 of *LNCS*, pages 40–47, 2013.

[7] David A. Mix Barrington, Howard Straubing, and Denis Thérien. Non-uniform automata over groups. *Information and Computation*, 89(2):109–132, 1990.

[8] Beate Bollig, Martin Sauerhoff, Detlef Sieling, and Ingo Wegener. Hierarchy theorems for kobdds and kibdds. *Theoretical Computer Science*, 205(1):45–60, 1998.

[9] Marek Chrobak. Finite automata and unary languages. *Theoretical Computer Science*, 47(0):149 – 158, 1986.

[10] Cynthia Dwork and Larry Stockmeyer. A time complexity gap for two-way probabilistic finite-state automata. *SIAM Journal on Computing*, 19(6):1011–1123, 1990.

[11] Viliam Geffert, Bruno Guillon, and Giovanni Pighizzini. Two-way automata making choices only at the endmarkers. In *Language and Automata Theory and Applications*, volume 7183 of *LNCS*, pages 264–276. Springer Berlin Heidelberg, 2012.

[12] Markus Holzer. Multi-head finite automata: data-independent versus data-dependent computations. *Theoretical Computer Science*, 286(1):97–116, 2002.

[13] Christos A. Kapoutsis. Removing bidirectionality from nondeterministic finite automata. In *MFCS*, volume 3618 of *LNCS*, pages 544–555. Springer, 2005.

[14] Christos A. Kapoutsis. Size complexity of two-way finite automata. In *Developments in Language Theory*, volume 5583 of *LNCS*, pages 47–66. Springer, 2009.

[15] Christos A. Kapoutsis. Nondeterminism is essential in small two-way finite automata with few reversals. *Information and Computation*, 222(0):208 – 227, 2013. 38th International Colloquium on Automata, Languages and Programming (ICALP 2011).

[16] Christos A. Kapoutsis, Richard Královic, and Tobias Mömke. Size complexity of rotating and sweeping automata. *Journal of Computer and System Sciences*, 78(2):537–558, 2012.

[17] K. Khadiev. Width hierarchy for k-obdd of small width. *Lobachevskii Journal of Mathematics*, 36(2):178–183, 2015.

[18] K. Khadiev. On the hierarchies for deterministic, nondeterministic and probabilistic ordered read-k-times branching programs. *Lobachevskii Journal of Mathematics*, 37(6):682–703, 2016.

[19] Kamil Khadiev and Rishat Ibrahimov. Width hierarchies for quantum and classical ordered binary decision diagrams with repeated test. *arXiv preprint arXiv:1703.07891*, 2017. Submitted to RuFiDim 2017.

[20] Kamil Khadiev and Aliya Khadieva. Reordering method and hierarchies for quantum and classical ordered binary decision diagrams. 2017. Accepted by CSR 2017.

[21] Hing Leung. Tight lower bounds on the size of sweeping automata. *Journal of Computer and System Sciences*, 63(3):384 – 393, 2001.

[22] Noam Nisan and Avi Widgerson. Rounds in communication complexity revisited. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 419–429. ACM, 1991.

[23] M.O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3:114–125, 1959.

[24] William J. Sakoda and Michael Sipser. Nondeterminism and the size of two way finite automata. In *STOC'78: Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 275–286, 1978.

[25] John C. Shepherdson. The reduction of two–way automata to one-way automata. *IBM Journal of Research and Development*, 3:198–200, 1959.

[26] Ingo Wegener. *Branching Programs and Binary Decision Diagrams.* SIAM, 2000.

A Proof of Theorem 1

This result follows from [25]. We present proof for completeness.

Let $\pi \in \Pi(\theta)$ be any partition such that $\theta \in \Theta(n)$ is the order of the inputs for A_n^θ, and, $\pi = (X_A, X_B)$ and $|X_A| = u$. Let pair (s, u), for $s \in S$ is state and $u \in \{1, \ldots, n\}$ is position of reading head on input tape, called configuration. We specify each configuration b_i for $i \in \{0, 2d+2\}$ as follows:

- b_0 is the initial configuration $(s_1, 1)$;
- b_i is the configuration $(s_i, u-1)$ for $i \in \{1, \ldots, d\}$;
- b_{d+1} is the configuration (s_a, n);
- b_{d+2} is the configuration (s_r, n);
- b_i is the configuration (s_j, u) for $j = i - d - 2$ and $i \in \{1, \ldots, d\}$.

We define matrix $M_A(\sigma, \gamma)$ that represents the computation of A_n^θ on input $\nu = (\sigma, \gamma)$ with respect to π:

$$M_A(\sigma, \gamma) = \left(\begin{array}{c|c} 0 & M_A(\sigma) \\ \hline M_A(\gamma) & 0 \end{array} \right),$$

where the sub-matrices $M_A(\sigma)$ and $M_A(\gamma)$ are defined as follows:

- $M_A(\sigma)$ is a $(d+3) \times d$-dimensional matrix that represents the computation on σ. Its (i, j)th entry, say $m_{i,j}$, is defined as follows:

$$m_{ij} = \begin{cases} 1, & i \leq d \text{ and if } A_n^\theta \text{ begins the computation from the configuration } b_i, \text{ then} \\ & \text{it reaches the configuration } b_{j+d+3} \text{ early than another } b_r \text{ for } r \geq d+3 \\ 0, & \text{otherwise} \end{cases}$$

- $M_A(\gamma)$ is a $d \times (d+3)$-dimensional matrix that represents the computation on γ. Its (i, j)th entry, say $m'_{i,j}$, is defined as follows:

$$m'_{ij} = \begin{cases} 1, & j \geq 0 \text{ and if } A_n^\theta \text{ begins the computation from the configuration } b_{i+d+3}, \\ & \text{then it reaches the configuration } b_j \text{ early than another } b_r \text{ for } r \leq d \\ 0, & \text{otherwise} \end{cases}$$

Let $p^0 = (1, 0, \ldots, 0)$ be the vector for initial configuration and q be a $(2d+3)$-dimensional column vector whose all entries are 0 except the $(d+1)$th entry which is 1 (corresponding to the accepting configuration). Now, we can represent the whole computation of A_n^θ on a given input by using the vectors p^0 and q and the matrix $M_A(\sigma, \gamma)$.

Lemma 18. *If the function $f(X)$ is computed by $2DA_n^\Theta$ A_n^θ of size d, then there is a non-negative integer t' for any $\nu \in \{0,1\}^n$ satisfying $f(\nu) = 1$ such that*

$$p^0 \cdot \Big(M_A(\sigma, \gamma) \Big)^{t'} \cdot q = 1$$

On the other hand, for any $\nu \in \{0,1\}^n$ satisfying $f(\nu) = 0$, there is no such t'.

Proof. The computation of the automaton can be traced by a $(2d+3)$-dimensional configuration (row) vectors, $p^0, p^1, p^2, \ldots$, where p^i represents the current configurations during the computation, i.e. the entry with value of 1 represents the

configuration to be in. The initial configuration vector is p^0, and, p^i is obtained by applying $M_A(\sigma,\gamma)$ to p^{i-1}: $p^i = p^{i-1} \cdot M_A(\sigma,\gamma)$, where $i > 0$.

Let us prove claim by contradiction. Let for the inputs ν, such that $f(\nu) = 1$, there are not t' such that $p^0 \cdot \Big(M_A(\sigma,\gamma)\Big)^{t'} \cdot q = 1$. A_n^θ computes f means result of A_n^θ computation on ν is 1. Hence there are t'' such that after t'' times visiting u-th variable the automata never visit it and accepts the input. But it means in this moment A_n^θ will be in configuration b_j such that $p_j^{t''} = 1$ and $p^{t''} \cdot q = 1$, because the automata accepts the input. Therefore, for any $t' \geq t''$, $p^0 \cdot \Big(M_A(\sigma,\gamma)\Big)^{t'} \cdot q$ always equals 1.

Let for the inputs ν, such that $f(\nu) = 0$, there are t' such that $p^0 \cdot \Big(M_A(\sigma,\gamma)\Big)^{t'} \cdot q = 1$. Then due definition of q, $M_A(\sigma,\gamma)$ and q after t' times visiting u-th variable the automata never visit this variable and accepts the input, but it is contradiction, because A_n^θ should returns 0 on this input. □ □

Let $\mathcal{M}$ be the set of all possible matrices $M_A(\sigma)$ such that $\mathcal{M} = \{M_A(\sigma) : \sigma \in \{0,1\}^{|X_A|}\}$. Now, we show that $N^\pi(f) \leq |\mathcal{M}|$. Assume that $N^\pi(f) > |\mathcal{M}|$. By Pigeonhole principle, there are $\sigma, \sigma' \in \{0,1\}^{|X_A|}$ and the corresponding mappings τ, τ' such that $\sigma \neq \sigma'$, $f|_\tau \neq f|_{\tau'}$ but $M_A(\sigma) = M_A(\sigma')$. Thus, there are $\gamma \in \{0,1\}^{|X_B|}$ such that $f|_\tau(\gamma) \neq f|_{\tau'}(\gamma)$, and so, $f(\nu) \neq f(\nu')$ for $\nu = (\sigma,\gamma)$ and $\nu' = (\sigma',\gamma)$.

Let $f(\nu) = 1$ and $f(\nu') = 0$. Due to Lemma 18, there is a t' such that $p^0 \cdot \Big(M_A(\sigma,\gamma)\Big)^{t'} \cdot q = 1$. However, $M_A(\sigma) = M_A(\sigma')$ and so $M_A(\sigma,\gamma) = M_A(\sigma',\gamma)$. That means $p^0 \cdot \Big(M_A(\sigma',\gamma)\Big)^{t'} \cdot q = 1$. But, this is a contradiction. Therefore, $N^\pi(f) \leq |\mathcal{M}|$. Moreover, since our computation is deterministic, there no more than one 1 in each ith row of $M \in \mathcal{M}$ for $i \leq d$. So, $|\mathcal{M}| \leq (d+1)^{d+1}$ and $N^\pi(f) \leq (d+1)^{d+1}$.

By definition of $N^\theta(f)$, we have $N^\theta(f) \leq (d+1)^{d+1}$ and, by definition of $N(f)$, we have $N(f) \leq (d+1)^{d+1}$. The proof of Theorem 1 is completed.

B Proof of Theorem 2

This result follows from [25]. We present proof for completeness.

Let $\pi \in \Pi(\theta)$ be any partition such that $\theta \in \Theta(n)$ is the order of the inputs for A_n^θ, and, $\pi = (X_A, X_B)$ and $|X_A| = u$. We define $M_A(\sigma)$ and $M_A(\gamma)$ with the following modification:

$$m_{ij} = \begin{cases} 1, & i \leq d \text{ and if } A_n^\theta \text{ begins the computation from the configuration } b_i \text{ and} \\ & \text{reaches } x_u \text{ in set of states } s, \text{ then } l \in s \text{ for } b_{j+d+3} = (l,u) \\ 0, & \text{otherwise} \end{cases}.$$

$$m'_{ij} = \begin{cases} 1, & j \geq 0 \text{ and if } A_n^\theta \text{ begins the computation from the configuration } b_{i+d+3} \text{ and} \\ & \text{reaches } x_{u-1} \text{ in set of states } s, \text{ then } l \in s \text{ for } b_j = (l,u) \\ 0, & \text{otherwise} \end{cases}.$$

Then, we can obtain the counterpart of Lemma 18.

Lemma 19. *If the function $f(X)$ is computed by A_n^θ of size d, then for any $\nu \in \{0,1\}^n$ satisfying $f(\nu) = 1$, there is a t' such that*

$$p^0 \cdot \Big(M_A(\sigma,\gamma)\Big)^{t'} \cdot q \geq 1$$

On the other hand, for any $\nu \in \{0,1\}^n$ satisfying $f(\nu) = 0$ there is no such t'.

Proof. The proof is the same except that the entry corresponding to the accepting configurations can get the a value greater than 1 for the inputs the function takes the value of 1 since the computation is nondeterministic and there can be more than one accepting path. And, again this entry is always 0 for the inputs the function take the value of 0. □ □

The inequality $N^\pi(f) \leq |\mathcal{M}|$ can be obtained in the same way.[1] Since the computation is nondeterministic, any position of ith row of $M \in \mathcal{M}$ can be 0 or 1 for $0 \leq i \leq d$. Therefore, $|\mathcal{M}| \leq 2^{(d+1)^2}$ and $N^\pi(f) \leq 2^{(d+1)^2}$. Moreover, by the definition of $N^\theta(f)$, we have $N^\theta(f) \leq 2^{(d+1)^2}$ and, by the definition of $N(f)$, we have $N(f) \leq 2^{(d+1)^2}$. The proof of Theorem 2 is completed.

C Proof of Theorem 4

This result follows from [10]. We present proof for completeness.

We need some additional definitions to present our lower bound for $2\mathrm{PA}_n^\Theta$. Let $\beta \geq 1$. Two numbers p and p' are said to be $\beta - close$ if either

- $p = p' = 0$ or
- $p \geq 0$, $p' \geq 0$, and $\beta^{-1} \leq p/p' \leq \beta$.

And, two numbers p and p' are said to be $\beta - close$ mod λ if either

- $p \leq \lambda$ and $p' \leq \lambda$ or
- $p \geq \lambda$, $p' \geq \lambda$ and p and p' are $\beta - close$.

Let $P = \{p_{i,j}\}_{i,j=1}^m$ be a m-state Markov chain with starting state 1 and two absorbing states $m-1$ and m; $a(P)$ denote the probability that Markov chain P is absorbed in state m when started in state 1; and, $T(P)$ denote the expected time to absorption into one of the states $m-1$ or m. Two Markov chains $P = \{p_{i,j}\}_{i,j=1}^m$ and $P' = \{p'_{i,j}\}_{i,j=1}^m$ are said to be $\beta - close$ mod λ if, for each pair i, j, $p_{i,j}$ and $p'_{i,j}$ are $\beta - close$ mod λ.

Let $\pi \in \Pi(\theta)$ be any partition such that $\theta \in \Theta(n)$ is the order of the inputs for A_n^θ, and, $\pi = (X_A, X_B)$ and $|X_A| = u$.

Let us consider configurations b_i for $i \in \{0, \ldots, 2d+2\}$, matrix $M_A(\sigma,\gamma)$ and vectors p^0, q similar to ones from Theorem 1. But $M_A(\sigma)$ and $M_A(\gamma)$ will be defined little different.

We define $M_A(\sigma)$ and $M_A(\gamma)$ with the following modifications:

[1] $M(\sigma)$ includes all information about the behavior of automaton on σ and so if there are two different σ and σ' with different subfunctions but the same matrix $M(\sigma)$, it should be different on some γ. But the behaviour of the automaton on this input is the same since $M(\sigma) = M(\sigma')$ and $M(\sigma,\gamma) = M(\sigma,\gamma')$. This is a contradiction.

- If A_n^θ begins the computation from the configuration b_i, then it reaches the configuration b_{j+d+3} early than another b_r for $r \geq d+3$ within probability m_{ij}
- if A_n^θ begins the computation from the configuration b_{i+d+3}, then it reaches the configuration b_j early than another b_r for $r \leq d$ within probability m'_{ij}.

Then, we can obtain the counterpart of Lemma 18.

Lemma 20. *If the function $f(X)$ is computed by A_n^θ of size d within error probability ε, then for any $\nu \in \{0,1\}^n$ satisfying $f(\nu) = 1$, there is a t' such that*

$$p^0 \cdot \Big(M_A(\sigma,\gamma)\Big)^{t'} \cdot q \geq \frac{1}{2} + \varepsilon$$

On the other hand, for any $\nu \in \{0,1\}^n$ satisfying $f(\nu) = 0$ there is no such t'.

Proof. Since computation is probabilistic, there can be more than one path from the configuration b_i, where it reaches the configuration b_{j+d+3} early than another b_r for $r \geq d+3$. But by construction of matrix M_A, we consider all of them. In [10] have shown that we can model probabilistic computation in this way. □ □

We have shown that we model computation of $2\mathrm{PA}_n^\Theta$ by Markov chain specified by matrix $M_A(\sigma,\gamma)$.

Lemma 21. *Let P and P' are two m-state Markov chains and $a(P) \geq \frac{1}{2} + \varepsilon$. Let $T = max\left(T(P), T(P'), m\right)$, $\lambda = \varepsilon^2/256T^3$ $\beta = \sqrt[2m]{\frac{1+\varepsilon+\varepsilon^2}{1+\varepsilon}}$. If P and P' are $\beta - close\ mod\ \lambda$, then $a(P') \geq \frac{1}{2} + \varepsilon/4$.*

Proof. Dwork and Stockmeyer [10] have shown that

$$a\left(P'\right) \geq \left(1 - 2\lambda m^3\right)\beta^{-2m} a\left(P\right) - 4\sqrt{\lambda m T}.$$

Since by substituting the values of λ and β,

$$a\left(P'\right) \geq \left(1 - 2\lambda T^3\right)\beta^{-2m} a\left(P\right) - 4\sqrt{\lambda T^3} \geq$$

$$\geq \left(1 - \varepsilon^2/128\right)\frac{1+\varepsilon}{1+\varepsilon+\varepsilon^2}\left(\frac{1}{2} + \varepsilon\right) - \varepsilon/4.$$

Since, for any $\varepsilon < 1/2$,

$$(1+2\varepsilon)\left(1 - \varepsilon^2/128\right) = 1 + \varepsilon + \varepsilon\left(1 - \varepsilon/128 - \varepsilon^2/64\right) \geq 1 + \varepsilon + \varepsilon^2$$

we have

$$\left(1 - \varepsilon^2/128\right)\frac{1+\varepsilon}{1+\varepsilon+\varepsilon^2}\left(\frac{1}{2} + \varepsilon\right) - \varepsilon/4 \geq$$

$$\geq \left(1 - \varepsilon^2/128\right)\frac{\frac{1}{2} + \varepsilon/2}{\left(\frac{1}{2} + \varepsilon\right)\left(1 - \varepsilon^2/128\right)}\left(\frac{1}{2} + \varepsilon\right) - \varepsilon/4 = \frac{1}{2} + \varepsilon/4.$$

□ □

Let $\mathcal{M}$ be the set of all possible matrices $M_A(\sigma)$ such that $\mathcal{M} = \{M_A(\sigma) : \sigma \in \{0,1\}^{|X_A|}\}$ and, for any $P \in \mathcal{M}$ and $P' \in \mathcal{M}$, P and P' are $\beta - close$ mod λ.

The inequality $N^{\pi}(f) \leq |\mathcal{M}|$ can be obtained in the same way.[2] To estimate $|\mathcal{M}|$ we use technique similar to one from [10]. Let $c = d(d+1)$. Define an equivalence relation on matrices in $\mathcal{M}$ as follows: $Q(w) \equiv Q'(w) \Leftrightarrow \forall i,j \quad (q_{i,j}(w) \leq \lambda \Leftrightarrow q'_{i,j}(w) \leq \lambda)$. Let E be a largest equivalence class. Since there are at most 2^c equivalence classes, $|\mathcal{M}| \leq 2^c|E(w)|$. Let d' be the number of entries $q_{i,j}(w)$ of $Q(w) \in E(w)$, such as $q_{i,j}(w) \geq \lambda$. By projecting to these entries, map Q(w) to an element $q(w) \in [\lambda, 1]^{d'}$. Let $\log q(w)$ be the componentwise log of $q(w)$, so $\log q(w) \in [\log \lambda, 0]^{d'}$. If two matrices are $\beta - close$ mod λ, then $|\log q_{i,j}(w) - \log q'_{i,j}(w)| \leq \log \beta$. By dividing each coordinate interval $[log\lambda, 0]$ into $\lceil -\log \lambda / \log \beta \rceil$ subintervals of length $\log \beta$ (with possibly one subinterval of length less than $\log \beta$), we have:

$$N^{id}(f) \leq |\mathcal{M}| \leq 2^c|E(w)| \leq$$

$$\leq \left\lceil \frac{-2\log\lambda}{\log\beta} \right\rceil^c \leq$$

$$\leq \left\lceil \frac{4d(8+3\log T)}{\log(1+2\varepsilon)/(1+\varepsilon)} \right\rceil^c .$$

The proof of Theorem 4 is completed.

D Proof of Theorem 6

The proof is based on two technical Lemmas 22 and 23.

Lemma 22. *Let $t = t(n)$ be some integers satisfying Inequality (1) and $\pi = (X_A, X_B)$ be a partition such that X_A contains at least t* value *variables from exactly t blocks. Then, X_B contains at least t* value *variables from exactly t blocks.*

Proof. We define $I_A = \{i : X_A$ contains at least t *value* variables from ith block$\}$. Let $i' \notin I_A$. Then, X_A contains at most $t-1$ *value* variables from i'th block, so X_B contains at least $b-(t-1)$ *value* variables from i'th block. By (1), we can get

$$b - (t-1) = \left\lfloor \frac{n}{2t} \right\rfloor - \lceil \log 2t \rceil - (t-1)$$

which is bigger than

$$(2t + \lceil \log 2t \rceil) - \lceil \log 2t \rceil - (t-1) = 2t - (t-1) = t+1.$$

Let $I = \{0, \ldots, 2t-1\}$ be the numbers of all blocks and $i' \in I \backslash I_A$. Then, we can follow that $|I \backslash I_A| = 2t - t = t$. □ □

[2] $M(\sigma)$ includes all information about the behaviour of automaton on σ and so if there are two different σ and σ' with different subfunctions but their matrices $M(\sigma)$ and $M(\sigma')$ are $\beta - close$ mod λ, it should be different on some γ. But the behaviour of the automaton on this input is the same and therefore matrices $M(\sigma, \gamma)$ and $M(\sigma, \gamma')$ are $\beta - close$ mod λ. This is a contradiction.

Let $\theta \in \Theta(n)$ be any order. Then, we pick a partition $\pi = (X_A, X_B) \in \Pi(\theta)$ such that X_A contains at least t *value* variables from exactly t blocks. We define $I_A = \{i :$ X_A contains at least t *value* variables from ith block$\}$ and $I_B = \{0, \ldots, 2t-1\} \backslash I_A$. By the proof of Lemma 22, we know that $|I_B| = t$.

Let (σ, γ) be the partition for the input ν with respect to π. We define the sets $\Psi \subset \{0,1\}^{|X_A|}$ and $\Gamma \subset \{0,1\}^{|X_B|}$ for the input ν with respect to π that satisfies the following conditions. For $\sigma, \sigma' \in \Psi$, $\gamma \in \Gamma$, $\nu = (\sigma, \gamma)$, and $\nu' = (\sigma', \gamma)$:

- For any $z \in \{0, \ldots, t-1\}$, $Ind(\nu, z) \in I_A$;
- For any $z \in \{w, \ldots, 2t-1\}$, $Ind(\nu, z) \in I_B$;
- There is $z \in \{2, \ldots, t-1\}$ such that $Val(\nu', z) \neq Val(\nu, z)$;
- The value of x_j^p is 0 for any $p \in I_B$ and $x_j^p \in X_A$;
- The value of x_j^p is 0 for any $p \in I_A$ and $x_j^p \in X_B$;
- $Val(\nu, 0) = 2t-2, Val(\nu', 1) = 2t-1$; and,
- $Val(\nu, 2t-2) = 0, Val(\nu, 2t-1) = 1$.

Lemma 23. *For any sequence $(a_2, \ldots, a_{2t-3})$, where $a_i \in \{0, \ldots, t-1\}$, there are $\sigma \in \Psi$ and $\gamma \in \Gamma$ such that $Val(\nu, z_i) = a_i$ for $z_i \in \{2, \ldots, t-1\}$ and $i \in \{2, \ldots, 2t-3\}$.*

Proof. Let $p_i \in I_A$ such that $p_i = Ind(\nu, z_i)$ for $i \in \{2, \ldots, t-1\}$. Remember that the value of $x_j^{p_i}$ is 0 for any $x_j^{p_i} \in X_B$. Hence the value of $Val(\nu, z_i)$ depends only on the variables from X_A. At least t value variables of p_ith block belong to X_A. Hence we can choose input with a_i 1's in the value variables of p_ith block which belongs to X_A. For set Γ and $i \in \{t, \ldots, 2t-3\}$, we can follow the same proof. □ □

Remember the statement of the theorem: For integer $t = t(n)$, if $\texttt{2-SAF}_t$ satisfies Inequality (1), then

$$N(\texttt{2-SAF}_t) \geq t^{t-2}.$$

Here are the details for the proofs.

Let $\theta \in \Theta(n)$ be an order. Then, we pick the partition $\pi = (X_A, X_B) \in \Pi(\theta)$ such that X_A contains at least t value variables from exactly t blocks.

Let $\sigma, \sigma' \in \Psi$ be two different inputs and τ and τ' be their corresponding mappings, respectively. We show that the subfunctions $\texttt{2-SAF}_\texttt{t}|_\tau$ and $\texttt{2-SAF}_\texttt{t}|_{\tau'}$ are different. Let $z \in \{2, \ldots, t-1\}$ such that $s' = Val(\nu', z) \neq Val(\nu, z) = s$.

If $z > 2$, then we choose $\gamma \in \Gamma$ providing that $Val(\nu, s+t) = 1$, $Val(\nu', s'+t) = 0$, and $Val(\nu, r) = Val(\nu', r) = z$, where $r = Val(\nu, 2)$. That is,

- $Step_1(\nu, 0) = Step_1(\nu', 0) = r$,
- $Step_2(\nu, 0) = s$ and $Step_2(\nu', 0) = s'$,
- $Step_1(\nu, 1) = 2t-1$ and $Step_1(\nu', 1) = 2t-2$, and,
- $Step_2(\nu, 1) = 1$ and $Step_2(\nu', 1) = 0$.

Thus, $\texttt{2-SAF}_t(\nu) = 1$ and $\texttt{2-SAF}_t(\nu') = 0$.

If $z = 2$, then we choose $\gamma \in \Gamma$ providing that $Val(\nu, s+t) = 1$ and $Val(\nu', s'+t) = 0$. That is,

- $Step_1(\nu, 0) = s$ and $Step_1(\nu', 0) = s'$,
- $Step_2(\nu, 1) = 1$ and $Step_2(\nu', 1) = 0$,

- $Step_1(\nu, 1) = 2t - 1$ and $Step_1(\nu', 1) = 2t - 2$, and,
- $Step_2(\nu, 1) = 1$ and $Step_2(\nu', 1) = 0$.

Hence $\texttt{2-SAF}_t(\nu) = 1$ and $\texttt{2-SAF}_t(\nu') = 0$.

Therefore $\texttt{2-SAF}_t|_\tau(\gamma) \neq \texttt{2-SAF}_t|_{\tau'}(\gamma)$ and also $\texttt{2-SAF}_t|_\tau \neq \texttt{2-SAF}_t|_{\tau'}$.

Now, we compute $|\Psi|$. For $\sigma \in \Psi$, we can get each value of $Val(\nu, i)$ for $2 \leq i \leq t - 1$. It means $|\Psi| \geq t^{t-2}$ due to Lemma 23. Therefore, $N^\pi(\texttt{2-SAF}_t) \geq t^{t-2}$, and, by definition of $N(\texttt{2-SAF}_t)$, we have $N(\texttt{2-SAF}_t) \geq t^{t-2}$.

E Proof of Theorem 7

We construct A_n. Let $\nu \in \{0,1\}^n$ be the input. We begin with the first part of automaton A_n which computes $Step_1(X, 0)$. Automaton A_n checks each jth block for predicate $Adr(\nu, j) = 2$. If it is true, then A_n computes $Val(\nu, 2) = r'$ and it checks the next block, otherwise. If A_n checks all blocks and does not find the block, it switches to the rejecting state. If A_n finds r', then it goes to one of the special state $s'_{r'}$. From this state, the automaton goes to the beginning of the input.

We continue with the second part of automaton A_n which computes $Step_2(X, 0)$. From the state $s'_{r'}$, A_n checks each jth block for predicate $Adr(\nu, j) = r'$. If it is true, then A_n computes $Val(\nu, r') = r''$ and it checks the next block, otherwise. If A_n checks all blocks and does not find the block, it switches to the rejecting state. If A_n finds r'', then it goes to one of special state $s''_{r''}$. From this state, the automaton goes to the beginning of the input.

Now, we describe the third part of automaton A_n which computes $Step_1(X, 1)$. From the state $s''_{r''}$, A_n checks each jth block for predicate $Adr(\nu, j) = r''$. If it is true, then A_n computes $Val(\nu, r'') = r'''$ and it checks the next block, otherwise. If A_n checks all blocks and does not find the block, then it switches to the rejecting state. If A_n finds r''', then it goes to one of special state $s'''_{r'''}$. From this state automaton goes to the beginning of the input.

The forth part of automaton A_n computes $Step_2(X, 1)$. From the state $s'''_{r'''}$, A_n checks each jth block for predicate $Adr(\nu, j) = r'''$. If it is true, then A_n computes $Val(\nu, r''') = r^{IV}$, and, it checks next block otherwise. If A_n checks all blocks and does not find the block, it switches to the rejecting state. If A_n finds r^{IV} and $r^{IV} = 1$, the automaton accepts the input and rejects the input, otherwise.

In the first part, the block checking procedure needs only 2 states. Computing $Val(\nu, 2)$ needs w states and there are t $s'_{r'}$ states. So, the size of the first part is $2 + 2t$. In the second part, the block checking procedure needs only 2 states and we have t blocks to check the state pairs for each value of r', that is $2t$ states. Computing $Val(\nu, r')$ needs t states and also A_n has t $s''_{r''}$ states. Therefore, the size of the second part is $4t$. Similarly, we can show that the size of the third part is $4t$. In the fourth part, we need $3t$ states for procedure checking and computing $Val(\nu, r''')$. A_n has also one *accept* and one *reject* states. So, the size of the forth part is $3t + 2$. Thus, the overall size of A_n is $13t + 4$.

F Proof of Theorem 8

We pick the partition $\pi = (X_A, X_B) \in \Pi(id)$ such that X_A contains exactly w blocks.

Let $\sigma, \sigma' \in \Psi$ be two different inputs and τ and τ' be their corresponding mappings, respectively. We show that the subfunctions $\texttt{2-USAF}_\texttt{t}|_\tau$ and $\texttt{2-USAF}_\texttt{t}|_{\tau'}$ are different. Let $z \in \{2, \ldots, t-1\}$ such that $s' = Val(\nu', z) \neq Val(\nu, z) = s$.

If $z > 2$, then we choose $\gamma \in \Gamma$ providing that $Val(\nu, s+t) = 1$, $Val(\nu', s'+t) = 0$, and $Val(\nu, r) = Val(\nu', r) = z$, where $r = Val(\nu, 2)$. That is,

- $Step_1(\nu, 0) = Step_1(\nu', 0) = r$,
- $Step_2(\nu, 0) = s$ and $Step_2(\nu', 0) = s'$,
- $Step_1(\nu, 1) = 2t - 1$ and $Step_1(\nu', 1) = 2t - 2$, and,
- $Step_2(\nu, 1) = 1$ and $Step_2(\nu', 1) = 0$.

Thus, $\texttt{2-USAF}_\texttt{t}(\nu) = 1$ and $\texttt{2-USAF}_\texttt{t}(\nu') = 0$.

If $z = 2$, then we choose $\gamma \in \Gamma$ providing that $Val(\nu, s + t) = 1$ and $Val(\nu', s' + t) = 0$. That is,

- $Step_1(\nu, 0) = s$ and $Step_1(\nu', 0) = s'$,
- $Step_2(\nu, 1) = 1$ and $Step_2(\nu', 1) = 0$,
- $Step_1(\nu, 1) = 2t - 1$ and $Step_1(\nu', 1) = 2t - 2$, and,
- $Step_2(\nu, 1) = 1$ and $Step_2(\nu', 1) = 0$.

Hence $\texttt{2-USAF}_\texttt{t}(\nu) = 1$ and $\texttt{2-USAF}_\texttt{t}(\nu') = 0$.

Therefore $\texttt{2-USAF}_\texttt{t}|_\tau(\gamma) \neq \texttt{2-USAF}_\texttt{t}|_{\tau'}(\gamma)$ and also $\texttt{2-USAF}_\texttt{t}|_\tau \neq \texttt{2-USAF}_\texttt{t}|_{\tau'}$.

Now, we compute $|\Psi|$. For $\sigma \in \Psi$, we can get each value of $Val(\nu, i)$ for $2 \leq i \leq t-1$. It means $|\Psi| \geq t^{t-2}$ due to Lemma 23. Therefore, $N^\pi(\texttt{2-USAF}_\texttt{t}) \geq t^{t-2}$, and, by definition of $N(\texttt{2-USAF}_\texttt{t})$, we have $N(\texttt{2-USAF}_\texttt{t}) \geq t^{t-2}$.

G Proof of Theorem 9

We construct A_n. It is consist of four parts similar as automaton from Theorem 7. In the first part, the block checking procedure needs $2\log 2t + 2$ states. Computing $Val(\nu, 2)$ needs $2t$ states and there are w $s'_{r'}$ states. So, the size of the first part is $3t + 2\log 2t + 2$. In the second part, the block checking procedure needs $2\log 2t + 2$ states and we have t blocks to check the state pairs for each value of r', that is $w(2\log 2t + 2)$ states. Computing $Val(\nu, r')$ needs $2t$ states and also A_n has t $s''_{r''}$ states. Therefore, the size of the second part is $5t + 2t\log 2t$. Similarly, we can show that the size of the third part is $5t + 2t\log 2t$. In the fourth part, we need $4t + 2t\log 2t$ states for procedure checking and computing $Val(\nu, r''')$. A_n has also one *accept* and one *reject* states. So, the size of the forth part is $4t + 2t\log 2t + 2$. Thus, the overall size of A_n is $23t + 2(1 + 3t)\log t + 6$.

Symmetric blind information reconciliation and hash-function-based verification for quantum key distribution

E.O. Kiktenko[1,2,3] A.S. Trushechkin[2,4,5] A.K. Fedorovılmaz[1,5,6]

[1]Russian Quantum Center, Skolkovo, Moscow 143025, Russia
[2]Steklov Mathematical Institute of Russian Academy of Sciences, Moscow 119991, Russia
[3]Bauman Moscow State Technical University, Moscow 105005, Russia
[4]National Research Nuclear University MEPhI, Moscow 115409, Russia
[5]Department of Mathematics and Russian Quantum Center, National University of Science and Technology MISiS, Moscow 119049, Russia
[6]LPTMS, CNRS, Univ. Paris-Sud, Université Paris-Saclay, Orsay 91405, France
e.kiktenko@rqc.ru, trushechkin@mi.ras.ru, akf@rqc.ru

Abstract

We consider an information reconciliation protocol for quantum key distribution (QKD). In order to correct down the error rate, we suggest a method, which is based on symmetric blind information reconciliation for the low-density parity-check (LDPC) codes. We develop a subsequent verification protocol with the use of ϵ-universal hash functions, which allows verifying the identity between the keys with a certain probability.

Keywords: quantum key distribution, information reconciliation, LDPC codes, universal hash functions, verification

1 Introduction

A paradigmatic problem of cryptography is the problem of key distribution [1]. Widely used tools for key distribution, based on so-called public key cryptography, use an assumption of the complexity of several mathematical problems such as integer factorization [2] and discrete logarithm [3]. However, a large-scale quantum computer would allow solving such tasks in a more efficient manner in compare with their classical counterparts using Shor's algorithm [4]. Therefore, most of information protection tools are vulnerable to attacks with the use of quantum algorithms. It should be also noted that absence of efficient non-quantum algorithms breaking public key tools remains unproved.

In the view of appearance of large-scale quantum computers, the crucial task is to develop a quantum-safe infrastructure for communications. Crucial components of such an infrastructure are information-theoretically secure schemes, which make no computational assumptions [1]. Examples of these schemes are the one-time-pad encryption [6, 5, 7] and Wegman-Carter authentication [8]. Nevertheless, the

need for establishing shared secret symmetric keys between communicating parties invites the challenge of how to securely distribute keys.

Quantum key distribution (QKD) offers an elegant method for key establishment between distant users (Alice and Bob), without relying on insecure public key algorithms [9]. During last decades, remarkable progress in theory, experimental study, and technology of QKD has been performed [10]. However, realistic error rates in the sifted key using current technologies are of the order of a few percent, which is high for direct applications [9, 10, 11]. Practical QKD then include a post-processing procedure, which is based on the framework of the classical information theory.

In this work, we focus on an information reconciliation task in QKD. We consider a method for error correction based on symmetric blind information reconciliation [12] with low-density parity-check (LDPC) codes [15, 16]. In order to check the identity between the keys after the error correction step, we develop a subsequent verification protocol with the use of ϵ-universal hash functions [20]. We note that the additional procedures (privacy amplification and authentication) are necessary to obtain secure keys shared only by Alice and Bob [11, 12, 13, 14].

The present paper is organized as follows. In Sec. 2, we describe the symmetric blind information reconciliation technique for the low-density parity-check (LDPC) codes. In Sec. 3, we suggest a subsequent verification protocol with the use of ϵ-universal hash functions, which allows one to verify the identity between the keys with a certain probability. In Sec. 4, we give estimations for the information leakage in the suggested information reconciliation protocol. We summarize the main results of our work in Sec. 5

2 Symmetric blind error correction

We suggest the protocol for information reconciliation which has two essential steps: the symmetric blind error correction (SBEC) procedure and the subsequent verification protocol. For the SBEC technique, a block of the sifted key is divided into $N_{\rm sb}$ sub-blocks of length $n_{\rm sb}$, and all sub-blocks are treated in parallel. All the resulting sub-blocks from SBEC are input data for the subsequent verification procedure. If it is necessary during the verification protocol the keys are divided into the same sub-blocks once again for determining the sub-blocks which contain an error (for details, see Fig. 1).

Let us consider the SBEC procedure for the particular sub-blocks of the sifted key $k^A_{\rm sift}$ and $k^B_{\rm sift}$ of length $n_{\rm sb}$ owned by Alice and Bob. For details of the SBEC procedure we refer the reader to Ref. [12], and here we confine ourself with explanation of the main steps.

For the implementation of the SBEC procedure we consider a set ("pool") of nine LDPC codes [15, 16] with the following rates:

$$\mathcal{R} = \{0.9, 0.85, \ldots, 0.5\}, \tag{1}$$

and the frame length $n_{\rm fr}$. The parity-check matrices of these codes can be generated with the use of the improved progressive edge growing algorithm [17] according to generating polynomials from Ref. [18].

The implementation of the SBEC procedure consists of following steps.

0) This initial step of the SBEC procedure realizes the preliminary initialization. The parties initialize zero strings e of length $n_{\rm fr}$.

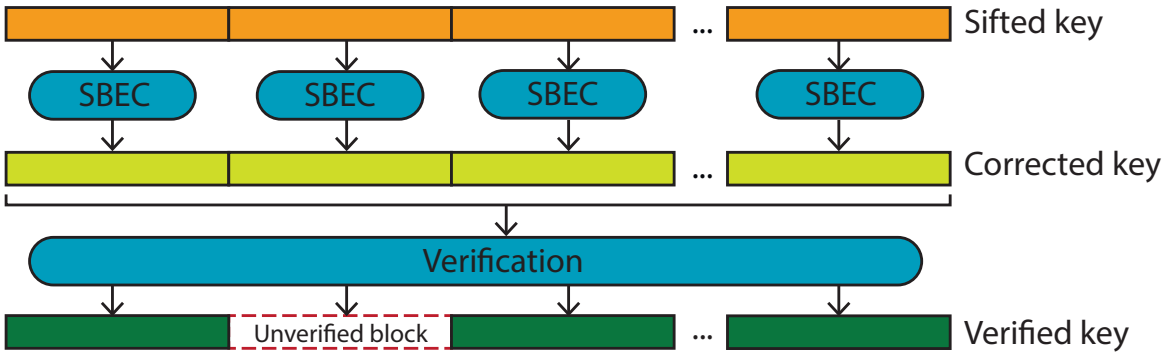


Figure 1: Scheme of the information reconciliation protocol: First, the block of the sifted key is split in sub-blocks which are treated with SBEC in parallel. Second, all the sub-blocks go all together through the verification step.

1) Parties implement the rate adaptation. Alice (Bob) extend their sub-blocks of the sifted key $k_{\text{sift}}^{A(B)}$ of length

$$n_{\text{sb}} := 0.95 n_{\text{fr}} \tag{2}$$

with n_{shrt} shortened and n_{pnct} punctured symbols, where

$$\begin{aligned} n_{\text{shrt}} &:= \lceil h(q_{\text{est}}) n_{\text{fr}} - n_{\text{sb}}(1-R) \rceil, \\ n_{\text{pnct}} &:= \Delta n_{\text{ext}} - n_{\text{shrt}}. \end{aligned} \tag{3}$$

Here, $\lceil \cdot \rceil$ stands for ceiling operation, $h(\cdot)$ is the binary entropy function,

$$\Delta n_{\text{ext}} := 0.05 n_{\text{fr}} \tag{4}$$

is the total number of shortened and punctured symbols, and the code rate R is chosen among $\mathcal{R}$ in such a way that both n_{shrt} and n_{pnct} are non-negative for current estimation of the QBER q_{est}.

The positions for punctured symbols are chosen according untainted puncturing technique [19], while the positions for shortened symbols are chosen pseudo-randomly. We denote the list of positions with sifted key bits as Ω.

2) This step is the realization of the syndromes exchange. Alice and Bob exchange with syndromes

$$s_{A(B)} := k_{\text{ext}}^{A(B)} \mathbf{H}_R^{\text{T}}, \tag{5}$$

where $\mathbf{H}_R$ is the parity-check matrix corresponded to code rate R ($k_{\text{ext}}^{A(B)}$ and $s_{A(B)}$ are treated as row-vectors; T stands for transposition). All the summations in vector matrix multiplication are assumed to be performed by modulo 2.

3) Alice and Bob use belief propagation decoding. The parties apply syndrome decodings based on the belief propagation algorithm with log-likelihood ratio (LLR) [12]. As the input both the parties use the "relative syndrome"

$$\Delta s = s_A \oplus s_B, \tag{6}$$

current information about error pattern e (note, that it was initialized as zero string in the begining if the procedure), parity-check matrix H_R, the estimated QBER q_{est}, and current positions of shortened and punctured symbols ($\oplus$ stands for modulo 2 summation). If the algorithm converges, it returns the new value of the error pattern e, and then Alice calculates the corrected key as follows:

$$k^A_{\text{cor}} := k^A_{\text{ext}}[\Omega] \oplus e[\Omega], \tag{7}$$

where Ω is the list of positions of sifted key symbols in the extended key. Bob assumes his corrected key has the following form:

$$k^B_{\text{cor}} := k^B_{\text{sift}}. \tag{8}$$

The SBEC is finished.

4) If the belief propagation algorithm does not converge, the this step is applied. It is based on disclosing additional information as follows. The parties take

$$d := \lceil 56 - 40R \rceil \tag{9}$$

positions in the extended key, which has the minimal magnitudes of LLR values to the end of belief propagation algorithm, disclose the values of their extended keys in these positions, and mark these positions as shortened. Then the parties update their error patterns e in the newborn shortened positions, and go to the Step 3.

There is still a certain probability that uncorrected errors remain after the SBEC procedure. In order to detect remaining errors, we implement the subsequent verification protocol, which is described below.

3 Verification

Here we suggest the verification protocol based on using the following ϵ-universal family of hash functions [20]:

$$h_k(X) := \mathsf{inttostr}\left[\sum_{i=1}^{n} \mathsf{strtoint}(x_i)k^{i-1} \bmod p\right], \tag{10}$$

where

$$k \in \mathbb{F} \equiv \{0, 1, 2, \ldots, p-1\} \tag{11}$$

is a randomly chosen key for the universal hashing, p is the prime number, X is a binary string of an arbitrary length,

$$(x_1 \| x_2 \| \ldots \| x_n) := X \tag{12}$$

is a partition of the string into substrings x_i of length

$$l_p = \lfloor \log_2 p \rfloor \tag{13}$$

($\lfloor \cdot \rfloor$ stands for floor operation), $\mathsf{inttostr}$ and $\mathsf{strtoint}$ are functions performing conversion between integer values an binary strings.

It can be shown [20] that the collision probability of the hash function (10) (*i.e.* the probability that $h_k(X) = h_k(Y)$ for some $X \neq Y$ and random k) is given by following expression:

$$\epsilon(l) \leq (\lceil l/l_p \rceil - 1)/p, \tag{14}$$

where l is length of X. Below the verification protocol based on the considered family of universal hash functions is considered.

Consider blocks of the corrected keys $K_{\rm cor}^{A(B)}$ owned by Alice (Bob), which consist of

$$n_{\rm b} = N_{\rm sb} \times n_{\rm sb} \tag{15}$$

bits. The implementation of the suggested verification protocol consists of following steps.

1) On the initial step of the verification protocol, the parties use generation of the key for universal hashing. Alice generate a random number $k \in \mathbb{F}$ using a true random generator (TRNG).

2) Further, the calculation of the hash for the whole block on the Alice side is realized. Alice computes $h_k(K_{\rm cor}^A)$ and sends it Bob together with k.

3) This step is comparing the hashes for the whole block Bob computes $h_k(K_{\rm cor}^B)$ and compares it with $h_k(K_{\rm cor}^A)$. If the hashes are identical, then Bob sends the acknowledgement message to Alice. The parties then assume

$$K_{\rm ver}^{A(B)} := K_{\rm cor}^{A(B)}, \tag{16}$$

and the protocol finishes. Here $K_{\rm sift}^{A(B)}$ are blocks owned by Alice (Bob). If the hashes are different, then Bob sends the negative-acknowledgement message, and the protocol continues.

4) Computing the hashes for all the sub-blocks on the Alice side is Step 4. Alice splits the corrected key $K_{\rm cor}^A$ in $N_{\rm sb}$ sub-blocks $\{k_{{\rm cor},i}^A\}$ of length $n_{\rm sb}$, generates $N_{\rm sb}$ random keys $\{k_i\}$ belonging to $\mathbb{F}$, calculates $\{h_{k_i}(k_{{\rm cor},i}^A)\}$, and sends both these sets to Bob.

5) Comparing the hashes for all the sub-block is used as the final step. Bob computes his versions of hashes $\{h_{k_i}(k_{{\rm cor},i}^B)\}$, compares them with corresponding values from Alice, and discard the sub-blocks with mismatched hashes from his corrected key $K_{\rm cor}^B$, to obtain the verified key $K_{\rm ver}^B$. Then he sends Alice the indices of unverified block, and she perform the same operation and the protocol finishes.

To estimate a probability of the remaining error in the verified keys consider a worst case scenario where all of $N_{\rm sb}$ sub-blocks contain errors after SBEC procedure. In this case, the probability of at least hash collision is given by the following expression:

$$\epsilon_{\rm ver} \leq \epsilon(n_{\rm b}) + [1 - \epsilon(n_{\rm b})] \left[1 - (1 - \epsilon(n_{\rm sb}))^{N_{\rm sb}}\right]. \tag{17}$$

Here the first term is a probability of collision for the whole block, and the second term is a probability of at least one collision in the verification of all the sub-blocks in the case of different hashes of the whole blocks . We note that the initial processing of the whole blocks $K_{\rm cor}^A$ and $K_{\rm cor}^B$ is performed to minimize the leakage

of the information via public discussion. Let the frame error rate (FER), that is a probability of remaining error after SBEC in each of the sub-blocks, to be equal to F. Then the information leakage in the verification step, neglecting the hash collision event, is given by the following expression:

$$\mathrm{leak}_{\mathrm{ec}}^{\mathrm{ver}} = (1-F)^{N_{\mathrm{sb}}} l_{\mathrm{ht}} + \\ + \left[1-(1-F)^{N_{\mathrm{sb}}}\right](N_{\mathrm{sb}}+1) l_{\mathrm{ht}}, \quad (18)$$

where

$$l_{\mathrm{ht}} = \lceil \log_2 p \rceil \quad (19)$$

is a hash length. In the Eq. (18) the first term corresponds to the case, where the whole blocks are identical and only one verification hash is transfered. The second term corresponds to the case of at least one error, where additional N_{sb} hashes are transfered.

4 Estimations

Let us consider our protocol based on a set of LDPC codes of frame length $n_{\mathrm{fr}} = 4000$ and code rates given by Eq. (1). According to Eq. (4) and Eq. (2) the number of sifted key bit processed in each sub-block is $n_{\mathrm{sb}} = 3800$. Taking the number of sub-blocks $N_{\mathrm{sb}} = 256$ and the prime number for universal hashing $p = 2^{50} - 27$ with hash length $l_{\mathrm{ht}} = 50$ bit, we obtain the following bound on a probability of the verification fail:

$$\epsilon_{\mathrm{ver}} \leq 2 \times 10^{-11}. \quad (20)$$

Let us then consider a question about information leakage in the verification step. Assuming that the FER of SBEC is $F = 10^{-5}$, we obtain the following result:

$$\mathrm{leak}_{\mathrm{ec}}^{\mathrm{ver}} \approx 1.65 l_{\mathrm{ht}} \approx 83\,\mathrm{bit} \quad (21)$$

In order to compare our approach with currently available post-processing tools, we calculate the information leakage for the setups described in Ref. [21]. We note that in this case the hashes are added to each processed LDPC code block. Therefore, one has the following estimation:

$$\mathrm{leak}_{\mathrm{ec}}^{\mathrm{ver,alt}} \approx n l_{\mathrm{ht}} = 12\,800\,\mathrm{bit}. \quad (22)$$

It is thus clearly seen that the suggested approach has an advantage in

$$\mathrm{leak}_{\mathrm{ec}}^{\mathrm{ver,alt}} / \mathrm{leak}_{\mathrm{ec}}^{\mathrm{ver}} \approx 155 \text{ times}. \quad (23)$$

Thus, the suggested information reconciliation protocol allows one to decrease the information leakage in the verification protocol significantly.

5 Conclusion

We have presented the information reconciliation protocol which combines two approaches: SBEC, based on LDPC codes, and verification, based on ϵ-universal hashing. We have shown that applying SBEC for a number of sub-blocks in parallel and

performing verification for the general block allows significant decreasing the information leakage in the verification stage.

The presented procedure allows one to obtain identical keys from sifted keys with a known bound of an error probability, which depends on the particular parameters of the protocol, such as hash length, block length, and number of sub-blocks.

The open source proof-of-principle realization of the presented algorithms are available [13, 14].

6 Acknowledgments

We thank Y.V. Kurochkin and N.O. Pozhar for useful discussions. The work of A.T. and E.K. was supported by the grant of the President of the Russian Federation (project MK-2815.2017.1). A.K.F. is supported by the RFBR grant (17-08-00742).

References

[1] B. Schneier, *Applied cryptography* (John Wiley & Sons, Inc., New York, 1996).

[2] R.L. Rivest, A. Shamir, and L. Adleman, Commun. ACM **21**, 120 (1978).

[3] W. Diffie and M.E. Hellman, IEEE Trans. Inform. Theor. **22**, 644 (1976).

[4] P.W. Shor, SIAM J. Comput. **26**, 1484 (1997).

[5] G.S. Vernam, J. Amer. Inst. Electr. Engineers **45**, 109 (1926).

[6] C.E. Shannon, Bell Syst. Tech. J. **27**, 379 (1948).

[7] V.A. Kotel'nikov, Classified Report (1941); see S.N. Molotkov, Phys. Usp. **49**, 750 (2006).

[8] M.N. Wegman and J.L. Carter, J. Comp. Syst. Sci. **22**, 265 (1981).

[9] For a review, see N. Gisin, G. Ribordy, W. Tittel, and H. Zbinden, Rev. Mod. Phys. **74**, 145 (2002).

[10] E. Diamanti, H.-K. Lo, and Z. Yuan, npj Quant. Inf. **2**, 16025 (2016).

[11] E.O. Kiktenko, A.S. Trushechkin, Y.V. Kurochkin, and A.K. Fedorov, J. Phys. Conf. Ser. **741**, 012081 (2016).

[12] E.O. Kiktenko, A.S. Trushechkin, C.C.W. Lim, Y.V. Kurochkin, and A.K. Fedorov, arXiv:1612.03673.

[13] E.O. Kiktenko, M.N. Anufriev, N.O. Pozhar, and A.K. Fedorov, Symmetric information reconciliation for the QKD post-processing procedure (2016).

[14] E.O. Kiktenko, A.S. Trushechkin, M.N. Anufriev, N.O. Pozhar, and A.K. Fedorov, Post-processing procedure for quantum key distribution systems (2016).

[15] R. Gallager, IRE Trans. Inf. Theory **8**, 21 (1962).

[16] D.J.C. MacKay, IEEE Trans. Inf. Theory **45**, 399 (1999).

[17] J. Martínez-Mateo, D. Elkouss, and V. Martin, IEEE Comm. Lett. **14**, 1155 (2010).

[18] D. Elkouss, A. Leverrier, R. Alleaume, and J.J. Boutros in *Proceedings of the IEEE International Symposium on Information Theory*, Seoul, South Korea, (2009), p. 1879.

[19] D. Elkouss, J. Martínez-Mateo, and V. Martin, IEEE Wireless Comm. Lett. **1**, 585 (2012).

[20] T. Krovetz and P. Rogaway, Lect. Notes Comp. Sci. **2015**, 73 (2001).

[21] N. Walenta, A. Burg, D. Caselunghe, J. Constantin, N. Gisin, O. Guinnard, R. Houlmann, P. Junod, B. Korzh, N. Kulesza, M. Legré, C.C.W. Lim, T. Lunghi, L. Monat, C. Portmann, M. Soucarros, P. Trinkler, G. Trolliet, F. Vannel, and H. Zbinden, New J. Phys. **16** 013047 (2014).

On the probability of finding marked connected components using quantum walks*

Nikolajs Nahimovs[1], Raqueline A. M. Santos[1] and Kamil Khadiev[1,2]

[1]Center for Quantum Computer Science, University of Latvia
Raina bulv. 19, Riga, LV-1586, Latvia
[2]Kazan Federal University,
Krmlevskaya 18, Kazan, 420008, Russia
nikolajs.nahimovs@lu.lv, rsantos@lu.lv, kamilhadi@gmail.com

Abstract

Finding a marked vertex in a graph can be a complicated task when using quantum walks. Recent results show that for two or more adjacent marked vertices search by quantum walk with Grover's coin may have no speed-up over classical exhaustive search. In this paper, we analyze the probability of finding a marked vertex for a set of connected components of marked vertices. We prove two upper bounds on the probability of finding a marked vertex and sketch further research directions.
Keywords: quantum computing, quantum walks, exceptional configurations, stationary states

1 Introduction

Searching is an important problem in Computer Science. Using Grover's quantum algorithm [3] one can solve the unstructured search problem quadratically faster than classically. A quadratic speed-up is also obtained when searching for a single marked vertex in some classes of graphs by using quantum walks [10, 6, 5]. In case of multiple marked vertices the situation gets more tricky. Krovi *et al.* [5] gave a quantum walk based algorithm that achieves the quadratic speed-up for any reversible and ergodic Markov chain and showed that for multiple marked vertices it can search quadratically faster than a quantity called the "extended hitting time", which is equivalent to the hitting time for one marked vertex and lower-bounded by it. Recently, Hoyer and Komeili [4] described a quantum walk based algorithm for finding multiple marked vertices in the two-dimensional lattice. Their algorithm uses quadratically fewer steps than a random walk on the two-dimensional lattice, ignoring logarithmic factors. On the other hand, for some quantum walk based search algorithms additional marked vertices can make the search easier or harder depending on the placement of marked vertices[7].

*This work was supported by the RAQUEL (Grant Agreement No. 323970) project, the Latvian State Research Programme NeXIT project No. 1, and the ERC Advanced Grant MQC.

In this paper, we consider search by coined discrete-time quantum walk [1] on general graphs with multiple marked vertices. Suppose we have a graph $G = (V, E)$ with a set of vertices V and a set of edges E. Let $n = |V|$ and $m = |E|$. The discrete-time quantum walk on G has associated Hilbert space $\mathcal{H}^{2m}$ with the set of basis states $\{|v, c\rangle : v \in V, 0 \leq c < d_v\}$, where d_v is the degree of vertex v. The evolution operator is the product of the coin operator followed by the shift operator, that is, $U = S \cdot C$. The coin transformation C is the direct sum of coin transformations for individual vertices, i.e. $C = C_{d_1} \bigoplus \cdots \bigoplus C_{d_n}$ with C_{d_i} being the Grover diffusion transformation of dimension d_i. The shift operator S acts as $S|v, c\rangle = |v', c'\rangle$, where v and v' are adjacent, c and c' represent the directions that points from v to v' and from v' to v, respectively.

Searching for a marked vertex is done using the unitary operator $U' = S \cdot C \cdot Q$, where Q is the query transformation, which flips the signs of the amplitudes at the marked vertices, that is,

$$Q = I - 2 \sum_{w \in M} \sum_{c=0}^{d_w - 1} |w, c\rangle\langle w, c|, \tag{1}$$

with M being the set of marked vertices.

The initial state of the algorithm is the equal superposition over all vertex-direction pairs:

$$|\psi(0)\rangle = \frac{1}{\sqrt{2m}} \sum_{v=0}^{n-1} \sum_{c=0}^{d_v - 1} |v, c\rangle. \tag{2}$$

It can be easily verified that the initial state stays unchanged by the evolution operator U, regardless of the number of steps (the same holds for the search operator U' is there are no marked vertices).

In this model, Nahimovs, Rivosh, and Santos [7, 8] were able to define a set of configurations of marked vertices for which quantum walk search does not have any speed-up over the classical exhaustive search. The reason for this is that for such configurations the initial state of the algorithm (2) is close to a 1-eigenvector of the search operator U'. Therefore, the probability of finding a marked vertex stays close to the initial probability and does not grow over time. Instead of analyzing the eigenspectrum of the search operator U' for each configuration of marked vertices the authors of [7, 8] gave the general conditions for a state to be stationary (1-eigenvector of U') in terms of amplitudes of individual vertices and, based on the conditions, constructed the set of "bad" configurations of marked vertices (referred in the papers as *exceptional configurations*).

Another type of exceptional configurations were found by Ambainis and Rivosh for the two-dimensional lattice [2]. In this case, when all vertices on the diagonal are marked, the quantum walk evolves by flipping the signs of the amplitudes of the initial state and the system remains in a uniform probability distribution for all time. Wong and Santos [11] showed that the same happens for the cycle and any higher-dimensional graph that reduces to the 1D line by using Szegedy's quantum walk model.

Recently, Prūsis, Vihrovs and Wong [9] have studied the existence of stationary states on general graphs with multiple marked vertices and found the necessary and sufficient conditions for a set of connected marked vertices to have to a stationary state (i.e. to have an assignment of amplitudes which is a 1-eigenvector of the search operator U').

In this paper, we consider a set of connected components (connected subsets) of marked vertices which has a stationary state. We show that the probability of finding a marked

vertex is upper bounded by a function of the amplitudes of the stationary state as well as properties of the marked components. We give the exact equation of the upper bound function (for both single and multiple marked components) and sketch further research directions.

The paper is structured as follows. In Section 2 we review the results in the literature by describing in which cases a set of marked vertices forms a stationary state. In Section 3 we study the behaviour of the probability of finding a marked vertex when we have a stationary state. We draw our conclusions in Section 4.

2 Stationary states

To start, let us introduce the notation. Let $G = (V, E)$ be a graph with a connected set of marked vertices M. Let $E_M = \{(i, j) \mid i, j \in M\}$ be the set of edges between marked vertices and $E_{\overline{M}} = E \backslash E_M$ be its complement. Let d_i be the degree of vertex i. Let d_i^M be the number of edges from a vertex i to vertices in M and $d_i^{\overline{M}}$ be the number of edges from vertex i to vertices in $V \backslash M$. Trivially, $d_i = d_i^{\overline{M}} + d_i^M$. Let $D^{\overline{M}} = \sum_{i \in M} d_i^{\overline{M}}$ be the total "outgoing" degree of a marked set.

A state $|\psi\rangle$ is stationary if $U'|\psi\rangle = |\psi\rangle$. As an example, consider a step of the walk for cycle of 5 vertices with 2 marked vertices shown on Fig. 1. For simplicity, we will use

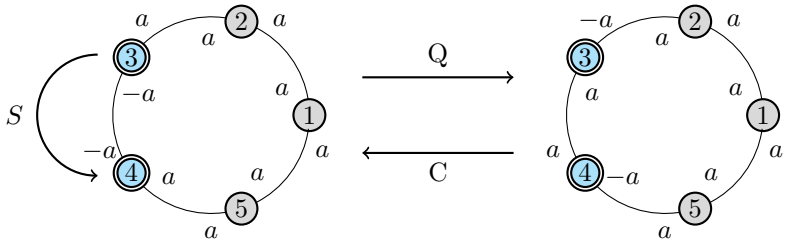


Figure 1: Illustration of the application of the evolution operator $U' = SCQ$ to the cycle of 5 vertices with two marked vertices ($M = \{3, 4\}$). Labels on edges represent directional amplitudes of a vertex. The state on the left side is a stationary state. The amplitudes of marked vertices pointing to each other are equal to $-a$, all other amplitudes are equal to a. In this case, the application of the query operator (Q) and the coin operator (C) will flip the sign of amplitudes in the marked vertices.

$|i, j\rangle$ to denote the direction amplitude of vertex i pointing towards vertex j. Using this notation the state on the left is written as

$$|\psi\rangle = a(|1,2\rangle + |1,5\rangle) + a(|2,1\rangle + |2,3\rangle) + a(|3,2\rangle - |3,4\rangle) + a(-|4,3\rangle + |4,5\rangle) + a(|5,4\rangle + |5,1\rangle).$$

and state on the right as

$$|\psi'\rangle = a(|1,2\rangle + |1,5\rangle) + a(|2,1\rangle + |2,3\rangle) - a(|3,2\rangle - |3,4\rangle) - a(-|4,3\rangle + |4,5\rangle) + a(|5,4\rangle + |5,1\rangle).$$

When applying the evolution operator $U' = SCQ$ to the state $|\psi\rangle$, the amplitudes are changed from state $|\psi\rangle$ to state $|\psi'\rangle$ and back. The query operator (Q) will flip the sign of

directional amplitudes of the marked vertices. The coin operator (C) will undo the effect of the query operation by flipping the signs again, as amplitudes in the marked vertices add up to zero. And since the directional amplitudes of adjacent vertices pointing to each other are equal, the shift operator (S) have no effect on the state. Therefore, $|\psi\rangle$ is not changed by a step of the walk, i.e. is stationary.

From this example, it is clear why a state with the following properties is stationary.

Theorem 1 ([8]). *Consider a state $|\psi\rangle$ with the following properties: all amplitudes of the unmarked vertices are equal; the sum of the amplitudes of any marked vertex is 0; the amplitudes of two adjacent vertices pointing to each other are equal. Then $|\psi\rangle$ is a stationary state of a step of the AKR algorithm.*

When a configuration of marked vertices forms a stationary state it is important to know how close is the stationary state to the initial state. Depending on that, it may or may not affect the search. Moreover, as we will see next, a configuration of marked vertices may have multiple stationary states. Therefore, we are interested in the stationary states which have maximal overlap with the initial state $|\psi(0)\rangle$.

According to Prūsis, Vihrovs and Wong [9], the existence of a stationary state depends on whether a marked connected component is bipartite or not, that is,

Theorem 2 ([9]). *A bipartite marked connected component has a stationary state if and only if the sums of $d_i^{\overline{M}}$ for each bipartite set are equal. A non-bipartite marked connected component always has a stationary state.*

In the example on Fig. 1, the connected component is bipartite and $d_3^{\overline{M}} = d_4^{\overline{M}} = 1$. Therefore, is forms a stationary state.

Now, suppose we have a graph with a marked connected component satisfying the Theorem 2. It has a stationary state (depicted on Fig. 2)

$$|\psi_{ST}^a\rangle = \sum_{\substack{i,j\in V\\ j\sim i}} a|i,j\rangle + \sum_{\substack{i,j\in M\\ j\sim i}} (c_{ij}-1)a|i,j\rangle, \tag{3}$$

where $j \sim i$ means there is an edge connecting vertex j to vertex i. All amplitudes of unmarked vertices (represented by single circles) are equal to a, so the coin transformation have no effect on these vertices. The amplitudes of marked vertices (represented by double circles) pointing to unmarked vertices are also equal to a. Let the amplitude of the marked vertex i pointing towards marked vertex j be $c_{ij}a$. Note, that according to Theorem 1 $c_{ij} = c_{ji}$, so the shift operator have no effect on the marked component. Moreover, the sum of the directional amplitudes of a marked vertex must be equal to zero. In this way, the coin operator will flip the sign of the amplitudes, undoing the effect of the query operator. Therefore, the amplitudes c_{ij} should satisfy

$$\sum_{\substack{j\in M\\ j\sim i}} c_{ij} = d_i^{\overline{M}}, \quad \forall i \in M. \tag{4}$$

A marked connected component may have multiple (infinitely many) stationary states satisfying the properties given in Theorem 1. For example, Fig. 3 shows two different

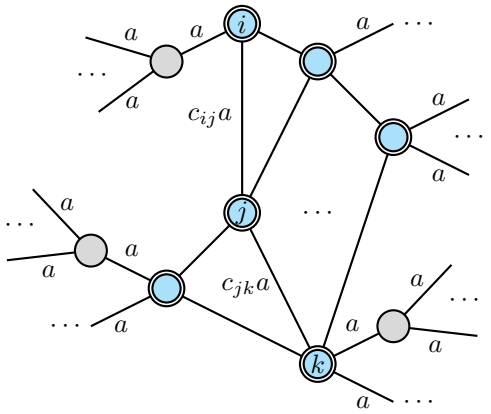
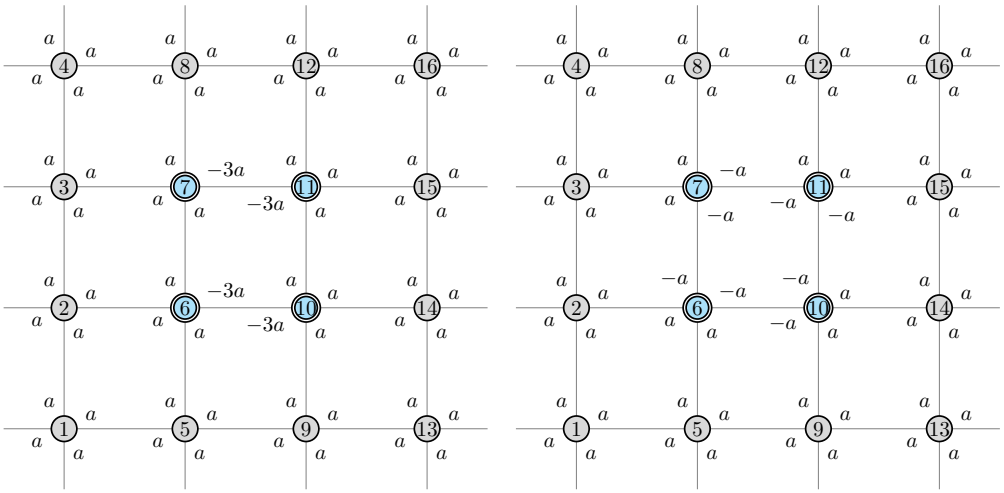


Figure 2: Sketch of amplitudes of a stationary state of a marked connected component. The marked vertices are represented by double circles. All amplitudes of unmarked vertices are equal to a.



(a) Stationary state with amplitudes $-3a$ at vertices 6 and 10, and 7 and 11 pointing to each other.

(b) Stationary state with amplitudes $-a$ at all marked vertices $\{6, 7, 10, 11\}$ pointing to each other.

Figure 3: Stationary states in the two-dimensional lattice.

stationary states for the same marked connected component in a two-dimensional lattice with N vertices.

It might seem that both states – $|\psi_1\rangle$ (on the left) and $|\psi_2\rangle$ (on the right) – have the same overlap with the initial state, however, this is not true. Although, for the both states the overlap is $4a^2(N-4)$, the value of a itself is different. As the states are unit vectors, the sum of squares of all amplitudes needs to be 1. Therefore, value of a for $|\psi_1\rangle$ is $\frac{1}{\sqrt{4(N+8)}}$ and the value of a for $|\psi_2\rangle$ is $\frac{1}{\sqrt{4N}}$. The probability of finding a marked vertex is $48a^2 = \frac{12}{N+8}$ for $|\psi_1\rangle$ and $16a^2 = \frac{4}{N}$ for $|\psi_2\rangle$.

In the following section, we consider a connected set of marked vertices having a stationary state, analyze the evolution of a state of the algorithm and prove an upper bound for the probability of finding a marked vertex.

3 Bounds on the probability

3.1 Upper bound on the probability for a connected component of marked vertices

Theorem 3. *Consider a graph $G = (V, E)$ with a connected component of marked vertices M. Let M be such that there exists a stationary state. Then, the probability p_M of finding a marked vertex, for any number of steps t, is*

$$p_M \leq \frac{2}{m}\left(\sum_{\substack{i,j\in M\\ j\sim i}} c_{ij}^2 + 2D^{\overline{M}} + 2|E_M|\right). \tag{5}$$

Proof. Consider the amplitudes of the stationary state: for all $|i,j\rangle$, where $i \notin M$ or $j \notin M$, the amplitudes are equal to a. For $|i,j\rangle$, where $i,j \in M$, let the amplitudes be $c_{ij} \cdot a$. The stationary state, then, can be written as

$$|\psi_{ST}^a\rangle = \sum_{\substack{i,j\in V\\ j\sim i}} a|i,j\rangle + \sum_{\substack{i,j\in M\\ j\sim i}} (c_{ij}-1)a|i,j\rangle.$$

Thus, we have

$$|\psi_0\rangle = |\psi_{ST}^a\rangle - \sum_{\substack{i,j\in M\\ j\sim i}} (c_{ij}-1)a|i,j\rangle,$$

for $a = 1/\sqrt{2m}$. We will denote the changing part of the initial state as $|\psi_{NST}\rangle$.

Out task is to upper bound the probability of finding a marked vertex p_M, that is, to find a distribution of $|\psi_{NST}\rangle$ over the graph, which maximizes the probability of finding a marked vertex. Clearly, the probability is maximized when the $|\psi_{NST}\rangle$ is distributed over the marked vertices only. Therefore, our task is to maximize

$$p_M = \sum_{i\in M}\left[\sum_{\substack{j\in V\setminus M\\ j\sim i}} (a+\alpha)^2 + \sum_{\substack{j\in M\\ j\sim i}} (c_{ij}a+\alpha_{ij})^2\right] \tag{6}$$

subject to

$$\sum_{i\in M}\left[\sum_{\substack{j\in V\setminus M\\ j\sim i}} \alpha^2 + \sum_{\substack{j\in M\\ j\sim i}} \alpha_{ij}^2\right] = |||\psi_{NST}\rangle||^2, \tag{7}$$

because the evolution is unitary and the norm will not be changed during the evolution. In Eq. (6), a and $c_{ij}a$ come from the stationary part of the state and α and α_{ij} come from the non-stationary part.

Marked vertex i has an "outgoing" degree $d_i^{\overline{M}}$. Therefore, we can rewrite Eqs. (6)-(7) as

$$p_M = \sum_{i\in M}\left[d_i^{\overline{M}}(a+\alpha)^2 + \sum_{\substack{j\in M\\ j\sim i}} (c_{ij}a+\alpha_{ij})^2\right] \tag{8}$$

subject to

$$\sum_{i\in M}\left[d_i^{\overline{M}}\alpha^2+\sum_{\substack{j\in M\\ j\sim i}}\alpha_{ij}^2\right]=|||\psi_{NST}\rangle||^2. \tag{9}$$

Let $D^{\overline{M}}=\sum_{i\in M}d_i^{\overline{M}}$ be the total "outgoing" degree of a marked component. Then, we have

$$p_M=D^{\overline{M}}(a+\alpha)^2+\sum_{\substack{i,j\in M\\ j\sim i}}(c_{ij}a+\alpha_{ij})^2. \tag{10}$$

subject to

$$D^{\overline{M}}\alpha^2+\sum_{\substack{i,j\in M\\ j\sim i}}\alpha_{ij}^2=|||\psi_{NST}\rangle||^2. \tag{11}$$

Next, we will use the following technical Lemma 1(proved in Appendix A).

Lemma 1. *Let* $f(X)=\sum_{i=1}^n(x_i-a_i)^2$ *and* $r^2=\sum_{i=1}^n x_i^2$. *Then*

$$argmax_X f=(-d\cdot a_1,\ldots,-d\cdot a_n),\ for\ d=\frac{r}{\sqrt{\sum_{i=1}^n a_i^2}}.$$

From the lemma we have that the probability reaches its maximum for $\alpha=\frac{r}{d}a$ and $\alpha_{ij}=\frac{r}{d}c_{ij}a$, where

$$r=|||\psi_{NST}\rangle||=\sqrt{\sum_{\substack{i,j\in M\\ j\sim i}}(c_{ij}-1)^2a^2} \tag{12}$$

and

$$d=\sqrt{D^{\overline{M}}a^2+\sum_{\substack{i,j\in M\\ j\sim i}}c_{ij}^2a^2}. \tag{13}$$

Thus, from Eq. (10), the probability p_M reaches its maximum for

$$D^{\overline{M}}\left(a+\frac{r}{d}a\right)^2+\sum_{\substack{i,j\in M\\ j\sim i}}\left(c_{ij}a+\frac{r}{d}c_{ij}a\right)^2=$$

$$\left(1+\frac{r}{d}\right)^2\left(D^{\overline{M}}a^2+\sum_{\substack{i,j\in M\\ j\sim i}}c_{ij}^2a^2\right)=$$

$$\left(1+\frac{r}{d}\right)^2d^2=(d+r)^2.$$

Using values for r and d from Eqs. (12) and (13) we obtain

$$a^2\left(\sqrt{D^{\overline{M}}+\sum_{\substack{i,j\in M\\ j\sim i}}c_{ij}^2}+\sqrt{\sum_{\substack{i,j\in M\\ j\sim i}}(c_{ij}-1)^2}\right)^2.$$

Opening the brackets (under the second square root) and using that

$$D^{\overline{M}} + \sum_{\substack{i,j\in M \\ j\sim i}} c_{ij} = 0,$$

we have

$$a^2 \left(\sqrt{D^{\overline{M}} + \sum_{\substack{i,j\in M \\ j\sim i}} c_{ij}^2} + \sqrt{\sum_{\substack{i,j\in M \\ j\sim i}} c_{ij}^2 + 2D^{\overline{M}} + 2|E_M|} \right)^2 .$$

Therefore,

$$p_M \leq 4a^2 \left(\sum_{\substack{i,j\in M \\ j\sim i}} c_{ij}^2 + 2D^{\overline{M}} + 2|E_M| \right) . \tag{14}$$

□

Note that the first term in Eq. (14) depends on the stationary state, while the two others – on the structure of the graph and the marked component.

For a given graph and a marked component there are infinitely many stationary states. Each stationary state gives a bound on the probability. For a tight bound one needs to consider the stationary state with the minimal $\sum_{\substack{i,j\in M \\ j\sim i}} c_{ij}^2$. This might be a hard task in the general case, and it is still an open question.

3.2 Upper bound on the probability for multiple marked components

Let us consider we have a disjoint set of k marked connected components $\mathrm{M} = \{M_1 \cup M_2 \cup \cdots \cup M_k\}$. Let $E_{M_l} = \{(i,j) \in E \mid i,j \in M_l\}$ be the set of edges inside the marked component M_l and let $d_i^{\overline{M_l}}$ be the number of edges from i to a vertex in $V \backslash M_l$. Then, it easily follows from Theorem 3 that

Corollary 1. *Consider a disjoint set of k connected components of marked vertices $\{M_1, M_2, \ldots, M_k\}$ such that there exists a stationary state with maximal overlap with the initial state $|\psi_0\rangle$, given by Eq. 2. Then, the probability p_M of finding a marked vertex, for any number of steps t, is*

$$p_M \leq \frac{2}{m} \sum_{l=1}^{k} \left(\sum_{\substack{i,j\in M_l \\ j\sim i}} c_{ij}^2 + 2D^{\overline{M_l}} + 2|E_{M_l}| \right) , \tag{15}$$

where $D^{\overline{M_l}} = \sum_{i\in M_l} d_i^{\overline{M_l}}$.

For example, if we consider a d-regular graph with a set of marked vertices which consist of k pairs of adjacent marked vertices (i.e. $|M_1| = |M_2| = \cdots = |M_k| = 2$). Then, the probability of finding a marked vertex, for any number of steps t, is $O\left(\frac{kd^2}{m}\right)$, where m is the number of edges of the graph. Note that $D_{M_l} = d - 1$ and $E_{M_l} = d - 1$ for all $l = 1, \ldots, k$.

4 Conclusions

Due to the interference phenomena, quantum walks behave differently from classical random walks. On the one hand, it can achieve a quadratic speed-up when searching for one marked vertex in a graph. On the other hand, additional marked vertices can make the search harder. We have seen that a placement of marked vertices on a graph can form a stationary state. However, having a stationary state does not automatically mean that the quantum search will not be able to find a marked vertex faster than classically. That is why we need to understand how the probability of finding a marked vertex behaves during the evolution. We proved that the probability is upper bounded by a function on the amplitudes of the stationary state and on the structure of the marked components.

As we have seen, there are infinitely many stationary states for a given set of marked connected components. It is still an open problem to find which stationary state has the maximal overlap with the initial state gives the minimum probability to find a marked vertex. In this way, we can obtain a tighter bound on the probability.

Another interesting question, is whether we can find applications for the exceptional configurations. One idea is to solve the problem of bipartite matching. Given a bipartite graph, our goal is to determine whether a perfect matching exists. The initial idea is to embed the graph into the two-dimensional lattice and make all its vertices marked. We will need to make some restrictions in the graph for that. Then, we claim that a perfect matching will exist if the marked vertices forms a stationary state. We plan to investigate this problem in the near future.

Acknowledgements. The authors thank A. Ambainis, A. Rivosh, K. Prūsis and J. Vihrovs for useful discussions and comments.

References

[1] Y. Aharonov, L. Davidovich, and N. Zagury. Quantum random walks. *Physical Review A*, 48(2):1687–1690, 1993.

[2] A. Ambainis and A. Rivosh. Quantum walks with multiple or moving marked locations. In *Proceedings of SOFSEM*, pages 485–496, 2008.

[3] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th ACM Symposium on the Theory of Computing*, pages 212–219, 1996.

[4] Peter Hoyer and Mojtaba Komeili. Efficient quantum walk on the grid with multiple marked elements. In Heribert Vollmer and Brigitte Valle, editors, *34th Symposium on Theoretical Aspects of Computer Science (STACS 2017)*, volume 66 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 42:1–42:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[5] Hari Krovi, Frédéric Magniez, Maris Ozols, and Jérémie Roland. Quantum walks can find a marked element on any graph. *Algorithmica*, 74(2):851–907, 2016.

[6] Frédéric Magniez, Ashwin Nayak, Peter C. Richter, and Miklos Santha. On the hitting times of quantum versus random walks. *Algorithmica*, 63(1):91–116, 2012.

[7] Nikolajs Nahimovs and Alexander Rivosh. Exceptional configurations of quantum walks with Grover's coin. In *Proceedings of the 10th International Doctoral Workshop on Mathematical and Engineering Methods in Computer Science*, MEMICS 2015, pages 79–92, Telč, Czech Republic, 2016. Springer.

[8] Nikolajs Nahimovs and Raqueline A. M. Santos. Adjacent vertices can be hard to find by quantum walks. In *SOFSEM 2017: Theory and Practice of Computer Science: 43rd International Conference on Current Trends in Theory and Practice of Computer Science, Limerick, Ireland, January 16-20, 2017, Proceedings*, pages 256–267, Cham, 2017. Springer International Publishing.

[9] Krišjānis Prūsis, Jevgēnijs Vihrovs, and Thomas G. Wong. Stationary states in quantum walk search. *Phys. Rev. A*, 94:032334, Sep 2016.

[10] M. Szegedy. Quantum speed-up of Markov chain based algorithms. In *Proceedings of the 45th Symposium on Foundations of Computer Science*, pages 32–41, 2004.

[11] Thomas G. Wong and Raqueline A. M. Santos. Exceptional quantum walk search on the cycle. *Quantum Information Processing*, 16(6):154, 2017.

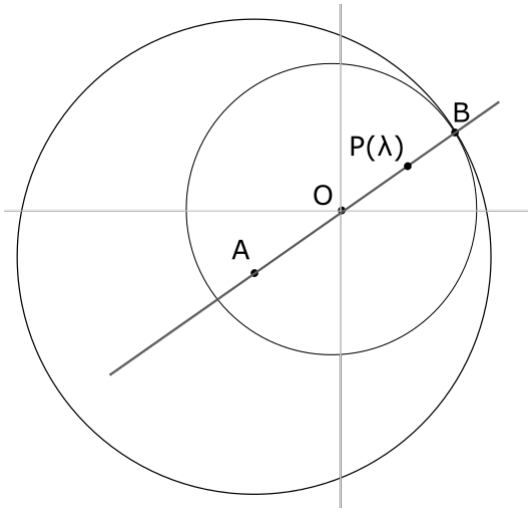
A Proof of Technical Lemma

Lemma 1. *Let* $f(X) = \sum_{i=1}^{n}(x_i - a_i)^2$ *and* $r^2 = \sum_{i=1}^{n} x_i^2$. *Then*

$$argmax_X f = (-d \cdot a_1, \ldots, -d \cdot a_n), \text{ for } d = \frac{r}{\sqrt{\sum_{i=1}^{n} a_i^2}}.$$

Proof. Let $f(X) = \sum_{i=1}^{n}(x_i - a_i)^2$. We want to find the maximal value of $f(X)$, such that $r^2 = \sum_{i=1}^{n} x_i^2$. Let us find $argmax_X f$. Observe that $r^2 = \sum_{i=1}^{n} x_i^2$ is the equation of a $(n-1)$-sphere, denote it S_1. Note that the center of S_1 is $(0, \ldots, 0)$. $f(X)$ is the radius of a $(n-1)$-sphere, denote it S_2. We should find the maximal radius of the sphere such that S_1 and S_2 still have common points.

Let point O be the center of S_1, point A be the center of S_2 and B be the intersection point of the spheres. Then, $|OB| = r$, $|AB| = \sqrt{f}$. Using the Triangle Inequality we can say that $|AB| \leq |OB| + |AO|$. It means that $|AB|$ will achieve its maximum when $|AB| = |OB| + |AO|$, therefore B belongs to the line OA.



Let us consider coordinates of any point on the line OA. This is: $P(\lambda) = (\lambda a_1, \ldots, \lambda a_n)$ for some real λ. Note that $P(1) = A$. Let us compute λ_0, such that $P(\lambda_0) = B$. The length of segment $OP(\lambda)$ is

$$|OP(\lambda)| = \sqrt{\sum_{i=1}^{n} (\lambda a_i)^2} = |\lambda| \sqrt{\sum_{i=1}^{n} a_i^2}.$$

Recall, that $|OP(\lambda_0)| = |OB| = r$, therefore

$$r = |\lambda_0| \sqrt{\sum_{i=1}^{n} a_i^2}, \text{ and } |\lambda_0| = \frac{r}{\sqrt{\sum_{i=1}^{n} a_i^2}}.$$

Note that $\lambda_0 < 0$ because $|P(1)P(|\lambda_0|)| < |P(1)P(-|\lambda_0|)|$. Therefore, $B = (-d \cdot a_1, \ldots, -d \cdot a_n)$ for

$$d = \frac{r}{\sqrt{\sum_{i=1}^{n} a_i^2}}.$$

In other words,

$$argmax_X f = (-d \cdot a_1, \ldots, -d \cdot a_n), \text{ for } d = \frac{r}{\sqrt{\sum_{i=1}^{n} a_i^2}}.$$

□

Quantum random number generators. The state of art, direction of research, tentative results.

Pavel Stremoukhov[1] Farid Ablayev[2] Ansar Safin[2]

[1]National Research University MPEI, Krasnokazarmennaya 14, Moscow, 111250, Russia
[2]Kazan Federal University, Krmlevskaya 18, Kazan, 420008, Russia
1opavel1@mail.ru, fablayev@gmail.com, arsafin@gmail.com

Abstract

A quantum random number generator (QRNG) is a device that utilizes the quantum phenomena to generate true random numbers. It based on fact that the measurement outcome of the microscopic particles is inherently random. In this paper we present brief survey of the art of QRNG constructing. The main goal of the talk is to discuss main possible directions in this hot quantum technology area.
Keywords: quantum random number generator, PRNG, HRNG, random numbers

1 Introduction

True random numbers seem to be of an ever increasing importance. Random numbers are essential in cryptography (classical, stochastic and quantum), Monte Carlo calculations, numerical simulations, statistical research, randomized algorithms, lottery etc. Today, true random numbers are probably most critically required in cryptography and its numerous applications to cyber-security such as: Smart Energy Grid, e-banking, internet trade, prepaid cards etc. A quantum random number generator (QRNG) is a device that utilizes the quantum phenomena to generate true random numbers. It based on fact that the measurement outcome of the microscopic particles is inherently random. Quantum mechanical theory suggests that some physical processes are inherently random (though collecting and using such data presents problems), but deterministic mechanisms, such as computers, cannot be. Any stochastic process (generation of random numbers) simulated on a computer, however, is not truly random, but only pseudorandom.

2 Basic types of random number generators

A random number is a number generated which is unpredictable, not reproducible and unbiased. A random number can be produced by either a software (deterministic algorithm) or hardware (physical process). Random number generators (RNGs) are essential in a wide range of applications, such as secure communications [1], stochastic simulations [2] and gambling [3]. Data encryption uses RNGs to generate cryptographic keys. Stochastic simulations depend on random numbers to

accurately price financial derivatives or model nuclear fusion. Random numbers are essential yet producing them with perfect unpredictability is still very challenging [4].

2.1 Pseudo RNG (PRNG)

PRNGs are computer software programs commonly used to generate substitutes for true random numbers. They are deterministic algorithms that rapidly generate bit sequences with long repetition lengths. In computational complexity theory and cryptography, the existence of pseudorandom generators is related to the existence of one-way functions and hard-core predicates through a number of theorems, collectively referred to as the pseudorandom generator theorem. Formally, pseudorandom generators exist if and only if one-way functions exist, or PRG is equivalent to OWF. Which leads us to a well-known P versus NP problem,which is generally considered unsolved

Truly independent and uniform random bits are either very difficult or impossible to produce. Computer algorithms instead use pseudorandom generators to generate a sequence of bits from some given seed. The generators typically found on our computers usually work well but occasionally give incorrect results both in theory and in practice.While they pass the statistical tests for randomness, they are not random at all. In software generators of random numbers (in most programming languages), the sequence repetition period most often does not exceed 2 in the degree of the operating system's bit depth, or even less. Similar to the coin toss example, if an observer knows the code, current state and inputs, it can reliably predict the output. When using such generators, one must be extremely cautious. It is better to study the recommendations of D. Knuth, and build your own generator, which has a well-known and rather long period in advance. They are designed to eliminate statistical anomalies but not withstand an intelligent observer. Pseudo-random number sampling algorithms are used to transform uniformly distributed pseudo-random numbers into numbers that are distributed according to a given probability distribution. Summarizing, there are two weaknesses: first, PRNG-generated sequences are unpredictable only under limitations of computational power, since PRNGs are inherently based on deterministic algorithms. Second, the random seeds, which are required to define the initial state of a PRNG, limit the amount of entropy in the random-number sequences they generate. This compromises the security of an encryption protocol.

2.2 Hardware RNG (HRNG)

The physical random number generator can be a classical process which is deterministic or a quantum process which is intrinsically random. HRNGs take input data from physical random processes to generate random bits. The famous classical RNG which is coin tossing can be predicted if initial conditions are known. Some examples include electronic and thermal noise [5] and chaotic dynamics in semiconductor lasers [6]. HRNGs derive randomness from classical physical phenomena that pass statistical tests for randomness. As with the coin toss example, however, they have far less entropy under observation. HRNGs are further criticized because they can be damaged or their measurement devices can be tampered with [4]. This numbers are essentially complex but yet deterministic.

2.3 Quantum RNG (QRNG)

Genuine randomness is possible only by a QRNG which is based on the quantumness of microscopic system. QRNGs are a subset of physical random number generators. Randomness is obtained using quantum phenomena, a non-deterministic process where all outcomes are possible and nothing is certain until it is observed. QRNG could fully solve such issues with Monte Carlo methods and are well suited to problems of real-time random number generation for simulations and modeling in high-performance computing. QRNG can have guaranteed indeterminism and entropy, since quantum processes are inherently unpredictable [7–8]. Examples of such processes include quantum phase fluctuations [9–13], spontaneous emission noise [14–16], photon arrival times [17–19], stimulated Raman scattering [20], photon polarization state [21–22], vacuum fluctuations [23–24], and even mobile phone cameras [25]. These QRNGs resolve both shortcomings of the PRNG.

3 Significance of QRNG

3.1 Statistical sampling and Monte-Carlo simulations

Monte Carlo methods are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results. Their essential idea is using randomness to solve problems that might be deterministic in principle. They are often used in physical and mathematical problems and are most useful when it is difficult or impossible to use other approaches. Monte Carlo methods are mainly used in three distinct problem classes: [26] optimization, numerical integration, and generating draws from a probability distribution.

Monte Carlo simulation methods require truly random numbers for applications, such as primality testing and etc. Many of the most useful techniques use deterministic, pseudorandom sequences, making it easy to test and re-run simulations. The only quality usually necessary to make good simulations is for the pseudo-random sequence to appear "random enough" in a certain sense. What this means depends on the application, but typically they should pass a series of statistical tests. Testing that the numbers are uniformly distributed or follow another desired distribution when a large enough number of elements of the sequence are considered is one of the simplest, and most common ones. There is a list of the characteristics of a high quality Monte Carlo simulation:

- the number generator has certain characteristics (e.g., a long "period" before the sequence repeats)
- the number generator produces values that pass tests for randomness
- there are enough samples to ensure accurate results
- the proper sampling technique is used
- the algorithm used is valid for what is being modeled
- it simulates the phenomenon in question.

Often, pseudo-random number sensors are used for Monte Carlo calculations. The main weakness of such generators is the presence of a certain period and as a consequence, predictability The Monte Carlo method can be used for values of N not exceeding (preferably much less than) the period of your pseudo-random number

generator. When making large calculations, you need to make sure that the properties of the random number generator allow you to carry out these calculations.

3.2 Cryptography

Random number generation is an important part of cryptography, because flaws in random number generation can be used by attackers to compromise encryption systems that are algorithmically secure. Random numbers are a fundamental tool in many cryptographic applications like key generation, encryption, masking protocols, or for internet gambling. It's required generators which are able to produce large amounts of secure random numbers for such applications as:

- asymmetric Key Generation
- PIN and Password Generation
- generation of Primes
- random Challenges for Authentication
- key Confirmation
- nonces

4 Future goals

Medium-Term Goals

- realising chip-based QRNG devices
- improved detectors for higher rates, as required for high performance computing and simulation, as well as for high-speed QKD schemes
- new concepts and ideas for the generation of quantum random numbers

Long-term Goals

- practical solutions for self-testing or device independent QRNGs

5 Approaches to the development of quantum random number generators

In the following work [27] demonstrated the generation of practical truly random numbers by using a RNG consisting of eight spin dice (a perpendicularly magnetized magnetic tunnel junction (p-MTJ)) integrated on a single-board circuit. It was shown that the quality of random numbers could be improved by performing an XOR operation on those generated by different p-MTJs. It was evaluated the quality of the random numbers by using the statistical test suite NIST SP-80018) and confirmed that they were practical truly random numbers. To generate high-quality random numbers, equiprobability of each switching event is indispensable. If $P_{switching}$ is exactly equal to 0.5 for all switching events, the statistical distribution of $P_{switching}$ of the generated random numbers should be the same as the binomial distribution. However, the switching probability fluctuates around the nominal value of $P_{switching} = 0.5$ owing to environmental effects such as thermal and current fluctuations. The following features are declared for the spin dice have:

- the binary random numbers are directly obtained from the results of spin-torque switching, and thus the equiprobability of the spin dice is ideal after a nested XOR operation is performed
- temperature compensation for the switching probability can be easily achieved by adjusting the amplitude of the current pulse
- the scalability of spin-torque switching makes it possible to fabricate a large number of arrays of spin dice on a chip and will provide an unlimited supply of practical truly random numbers

One more example is a QRNG prototype is discussed in the following article [28]. Proposed a way to generate the random bit series exploiting inherent randomness of the switching from a superconducting to a non-zero voltage state in Josephson junctions and superconducting nanowires. Authors propose to use the Brownian behavior of a superconducting wave function of a Josephson junction (JJ) or a superconducting nanowire to generate a sequence of random numbers. In superconductivity electrons are strongly correlated what allows to describe them with a single macroscopic wave function. The phase of the wave function, interacting randomly with environment, fluctuates just like a position of a single particle. It was achieved random number generation rates of 10-100 kb/s.

In the following article [29] demonstrated a bias-free true random number generator based on single photon detection using superconducting nanowire single photon detectors (SNSPDs). By comparing the photon detection signals of two consecutive laser pulses and extracting the random bits by the von Neumann correction method, it is achieved a random number generation efficiency of 25% a generation rate of 3.75 Mbit/s at a system clock rate of 15 MHz). In a three-channel SNSPD system, the random number bit generation efficiency was improved to 75%, corresponding to a generation rate of 7.5 Mbit/s with a 10 MHz system clock rate. All of the generated random numbers successfully passed the statistical test suite.

References

[1] Tajima et al., "Practical quantum cryptosystem for metro area applications," *IEEE J. Sel. Topics Quantum Electron.* 13 (2007).

[2] X. Cai and X. Wang, "Stochastic modelling and imulation of gene networks - a review of the state-of-the-rt research on stochastic simulations" *IEEE Signal rocess. Mag.* 24, 27–36 (2007).

[3] C. Hall and B. Schneier, "Remote electronic gambling" p. 232–238 (1997).

[4] Ferguson, Schneier and Kohno, "Cryptographic ngineering" *Wiley* 137–145 (2010).

[5] C. Petrie and J. Connelly "A noise-based ic random umber generator for applications in cryptography" *IEEE trans. Circuits Syst. I, Fundam. Theory Appl* 47, 615–621 (2000).

[6] Kanter et al., "An optical ultrafast random bit enerator" *Nature Photon.* 4, 51–61 (2010).

[7] Cristian S Calude, Michael J Dinneen, Monica Dumitrescu, and Karl Svozil "Experimental evidence of quantum randomness incomputability" *Physical Review A* 82, 022102 (2010).

[8] Karl Svozil "Three criteria for quantum random-number generators based on beam splitters" *Physical Review A* 79, 054306 (2009).

[9] Bing Qi, Yue-Meng Chi, Hoi-Kwong Lo, and Li Qian "High–speed quantum random number generation by measuring phase noise of a single-mode laser" *Optics Letters* 35, 312–314 (2010).

[10] Hong Guo, Wenzhuo Tang, Yu Liu, and Wei Wei "Truly random number generation based on measurement of phase noise of a laser" *Physical Review E* 81, 051137 (2010).

[11] M Jofre, M Curty, F Steinlechner, G Anzolin, JP Torres, MW Mitchell, and V Pruneri, "True random numbers from amplified quantum vacuum" *Optics Express* 19, 20665–20672 (2011).

[12] Feihu Xu, Bing Qi, Xiongfeng Ma, He Xu, Haoxuan Zheng, and Hoi-Kwong Lo "Ultrafast quantum random number generation based on quantum phase fluctuations" *Optics Express 20* 12366–12377 (2012).

[13] C. Abellan, W. Amaya, M. Jofre, M. Curty, A. Acin, J. Capmany, V. Pruneri, and M. W. Mitchell "Ultra-fast quantum randomness generation by accelerated phase diffusion in a pulsed laser diode" *Optics Express* 22, 1645–1654 (2014).

[14] Mario Stipcevic and B Medved Rogina "Quantum random number generator based on photonic emission in semiconductors" *Review of scientific instruments* 78, 045104 (2007).

[15] Caitlin RS Williams, Julia C Salevan, Xiaowen Li, Rajarshi Roy, and Thomas E Murphy "Fast physical random number generator using amplified spontaneous emission" *Optics Express* 18, 23584–23597 (2010).

[16] Y Liu, MY Zhu, B Luo, JW Zhang, and H Guo "Implementation of 1.6 tb/s 1 truly random number generation based on a super-luminescent emitting diode" *Laser Physics Letters* 10, 045001 (2013).

[17] Michael Wahl, Matthias Leifgen, Michael Berlin, Tino Roehlicke, Hans-Juergen Rahn, and Oliver Benson "An ultrafast quantum random number generator with provably bounded output bias based on photon arrival time measurements" *Applied Physics Letters* 98, 171105 (2011).

[18] Michael A Wayne and Paul G Kwiat, "Low-bias highspeed quantum random number generator via shaped optical pulses" *Optics Express* 18, 9351–9357 (2010).

[19] Hai-Qiang Ma, Yuejian Xie, and Ling-An Wu, "Random number generation based on the time of arrival of single photons" *Appl. Opt.* 44, 7760–7763 (2005).

[20] Philip J Bustard, Duncan G England, Josh Nunn, Doug Moatt, Michael Spanner, Rune Lausten, and Benjamin J Sussman "Quantum random bit generation using energy fluctuations in stimulated raman scattering" *Optics Express* 21, 29350–29357 (2013).

[21] M Fiorentino, C Santori, SM Spillane, RG Beausoleil, and WJ Munro "Secure self-calibrating quantum random-bit generator" *Physical Review A* 75, 032334 (2007).

[22] Giuseppe Vallone, Davide G Marangon, Marco Tomasin, and Paolo Villoresi "Quantum randomness certified by the uncertainty principle" *Physical Review A* 90, 052327 (2014).

[23] Thomas Symul, SM Assad, and Ping K Lam "Real time demonstration of high bitrate quantum random number generation with coherent laser light" *Applied Physics Letters* 98, 231103 (2011).

[24] Christian Gabriel, Christoffer Wittmann, Denis Sych, Ruifang Dong, Wolfgang Mauerer, Ulrik L Andersen, Christoph Marquardt, and Gerd Leuchs "A generator for unique quantum random numbers based on vacuum states" *Nature Photonics* 4, 711–715 (2010).

[25] Bruno Sanguinetti, Anthony Martin, Hugo Zbinden, and Nicolas Gisin "Quantum random number generation on a mobile phone" *Phys. Rev.* X 4, 031056 (2014).

[26] Kroese, D. P.; Brereton, T.; Taimre, T.; Botev, Z. I. (2014). "Why the Monte Carlo method is so important today". *WIREs Comput Stat.* 6: 386–392.

[27] Akio Fukushima, Takayuki Seki, Kay Yakushiji, Hitoshi Kubota, Hiroshi Imamura, Shinji Yuasa,Koji Ando *Applied Physics Express* Volume 7, 8 (2014)

[28] M. Foltyn, M. Zgirski, *Phys. Rev. Applied* 4, 024002 (2015)

[29] He Y, Zhang W, Zhou H, You L, Lv C, Zhang L, Liu X, Wu J, Chen S, Ren M, Wang Z and Xie X *Supercond. Sci. Technol.* 29 (085005) (2016)