Improving post-quantum cryptography through cryptanalysis

John M. Schanck

Institute for Quantum Computing
Depatment of Combinatorics and Optimization
University of Waterloo

June 24, 2020

Outline

- Context: timeline of my Ph.D. and the NIST post-quantum standardization effort.
- Some results from Chapters 2 and 3.
- Summary of recommendations for quantum cryptanalysis.

Context / 2016

- **Jan** Started Ph.D.
- **Feb** NIST announces post-quantum standards effort.
- **Aug** NIST circulates draft call for proposals.
- **Oct** Visit Peter Schwabe at Radboud — start of work on NTRU-HRSS and Kyber.
- **Dec** NIST circulates official call for proposals.

Context / 2017

- **June** "High speed key encapsulation from NTRU" accepted at CHES 2017. (Joint work with Hülsing, Rijneveld, Schwabe.)
- **Nov** "CRYSTALS–Kyber: a CCA-secure module-lattice-based KEM" accepted at EuroS&P 2018. (Joint work with Bos, Ducas, Kiltz, Lepoint, Lyubashevsky, Schwabe, Seiler, Stehlé.)
- **Nov** Submitted NTRU-HRSS and CRYSTALS–Kyber to NIST.

Context / 2018

- **Jan** Wrote “Multi-power post-quantum RSA” (Chapter 4).
- **Feb** Began collaboration with Samuel Jaques (Chapter 2).
- **Apr** NIST conference and EuroS&P.
- **Apr** Visit Martin Albrecht at Royal Holloway (Chapter 3).
- **Nov** Wrote “A Comparison of NTRU Variants”.
- **Nov** Announced NTRU-HRSS and NTRUEncrypt merger.
- **Dec** Google announces “CECPQ2” experiment, which features NTRU.

Context / 2019

- **Jan** NIST second round candidates announced.
- **Mar** Submitted new versions of NTRU and Kyber.
- **June** Cloudflare and Google announce they will compare NTRU-HRSS and SIKEp434.
- **Aug** Paper w/ Samuel Jaques (Chapter 2) receives "Best Young Researcher Paper" Award at CRYPTO.
- **Aug** NIST conference.

Context / 2020

- Jan Wrote "An upper bound on the decryption failure rate of static-key NewHope".
- Jan "Decryption failure is more likely after success" accepted at PQCrypto 2020. (Joint work with Nina Bindel.)
- Mar Preparations for Round 3 NTRU: faster software for one parameter set; decryption failure analysis for some variants.
- Mar Consumer versions of Google Chrome start to support NTRU.

Driving questions

- How should we evaluate (post-quantum) security?
- How should we compare cryptosytems?

NIST's guidance:

- **Security category 2**
  "Any attack that breaks the relevant security definition must require *computational resources* comparable to or greater than those required for collision search on a 256-bit hash function (e.g. SHA256/ SHA3-256)."
- The criteria must be met with respect to "all metrics that NIST deems to be potentially relevant to practical security."

Driving questions

- How should we evaluate (post-quantum) security?
- How should we compare cryptosytems?

NIST's guidance:

- **Security category 2**
  "Any attack that breaks the relevant security definition must require *computational resources* comparable to or greater than those required for collision search on a 256-bit hash function (e.g. SHA256/ SHA3-256)."
- The criteria must be met with respect to "all metrics that NIST deems to be potentially relevant to practical security."

Driving questions

- How should we evaluate (post-quantum) security?
- How should we compare cryptosytems?

NIST's guidance:

- **Security category 2**
  "Any attack that breaks the relevant security definition must require *computational resources* comparable to or greater than those required for collision search on a 256-bit hash function (e.g. SHA256/ SHA3-256)."
- The criteria must be met with respect to "all metrics that NIST deems to be potentially relevant to practical security."

Algorithms for 2-to-1 collision search

1997 **Brassard–Høyer–Tapp**:
$p = 1$ small quantum processor, $m = O(n^{1/3}) \approx 2^{85}$ bits of qRAM, and time for $t = O(n^{1/3}) \approx 2^{85}$ sequential Grover iterations of the hash function.

1996 **van Oorschot–Wiener**:
$p = n^{1/6} \approx 2^{43}$ small classical processors, $m = O(p)$ bits of memory, and time for $t = O(n^{1/2}/p) \approx 2^{85}$ sequential hash function evaluations.

Criticism of BHT:
2001 Grover–Rudolph
2007 Bernstein
2017 Liu–Perlner

Algorithms for 2-to-1 collision search

1997 **Brassard–Høyer–Tapp**:
$p = 1$ small quantum processor, $m = O(n^{1/3}) \approx 2^{85}$ bits of qRAM, and time for $t = O(n^{1/3}) \approx 2^{85}$ sequential Grover iterations of the hash function.

1996 **van Oorschot–Wiener**:
$p = n^{1/6} \approx 2^{43}$ small classical processors, $m = O(p)$ bits of memory, and time for $t = O(n^{1/2}/p) \approx 2^{85}$ sequential hash function evaluations.

Criticism of BHT:
2001 Grover–Rudolph
2007 Bernstein
2017 Liu–Perlner

Algorithms for 2-to-1 collision search

1997 **Brassard–Høyer–Tapp**:
$p = 1$ small quantum processor, $m = O(n^{1/3}) \approx 2^{85}$ bits of qRAM, and time for $t = O(n^{1/3}) \approx 2^{85}$ sequential Grover iterations of the hash function.

1996 **van Oorschot–Wiener**:
$p = n^{1/6} \approx 2^{43}$ small classical processors, $m = O(p)$ bits of memory, and time for $t = O(n^{1/2}/p) \approx 2^{85}$ sequential hash function evaluations.

Criticism of BHT:
2001 Grover–Rudolph
2007 Bernstein
2017 Liu–Perlner

Algorithms for 2-to-1 collision search

1997 **Brassard–Høyer–Tapp**:
$p = 1$ small quantum processor, $m = O(n^{1/3}) \approx 2^{85}$ bits of qRAM, and time for $t = O(n^{1/3}) \approx 2^{85}$ sequential Grover iterations of the hash function.

1996 **van Oorschot–Wiener**:
$p = n^{1/6} \approx 2^{43}$ small classical processors, $m = O(p)$ bits of memory, and time for $t = O(n^{1/2}/p) \approx 2^{85}$ sequential hash function evaluations.

Criticism of BHT:
2001 Grover–Rudolph
2007 Bernstein
2017 Liu–Perlner

Algorithms for golden collision search

What resources are required for an $n = 2^{128}$ element golden collision search?

1996 **Grover**:
$p = 1$ small quantum processors and time for $t = O(\sqrt{n^2}) \approx 2^{128}$ sequential steps.

2008 **Tani**:
$p = 1$ large quantum processor with $m = O(n^{2/3}) \approx 2^{85}$ qubits, and time for $t = O(n^{2/3}) \approx 2^{85}$ sequential quantum walk steps.

1996 **van Oorschot–Wiener**:
$p = n^{1/3} \approx 2^{43}$ small classical processors, $m = n^{2/3} \approx 2^{85}$ bits of memory, and time for $t = O(\sqrt{n^3/m}/p) \approx 2^{106}$ sequential hash function evaluations.

Algorithms for golden collision search

What resources are required for an $n = 2^{128}$ element golden collision search?

1996 **Grover**:
$p = 1$ small quantum processors and time for $t = O(\sqrt{n^2}) \approx 2^{128}$ sequential steps.

2008 **Tani**:
$p = 1$ large quantum processor with $m = O(n^{2/3}) \approx 2^{85}$ qubits, and time for $t = O(n^{2/3}) \approx 2^{85}$ sequential quantum walk steps.

1996 **van Oorschot–Wiener**:
$p = n^{1/3} \approx 2^{43}$ small classical processors, $m = n^{2/3} \approx 2^{85}$ bits of memory, and time for $t = O(\sqrt{n^3/m}/p) \approx 2^{106}$ sequential hash function evaluations.

Algorithms for golden collision search

What resources are required for an $n = 2^{128}$ element golden collision search?

1996 **Grover**:
$p = 1$ small quantum processors and time for $t = O(\sqrt{n^2}) \approx 2^{128}$ sequential steps.

2008 **Tani**:
$p = 1$ large quantum processor with $m = O(n^{2/3}) \approx 2^{85}$ qubits, and time for $t = O(n^{2/3}) \approx 2^{85}$ sequential quantum walk steps.

1996 **van Oorschot–Wiener**:
$p = n^{1/3} \approx 2^{43}$ small classical processors, $m = n^{2/3} \approx 2^{85}$ bits of memory, and time for $t = O(\sqrt{n^3/m}/p) \approx 2^{106}$ sequential hash function evaluations.

Chapter 2: SIKE

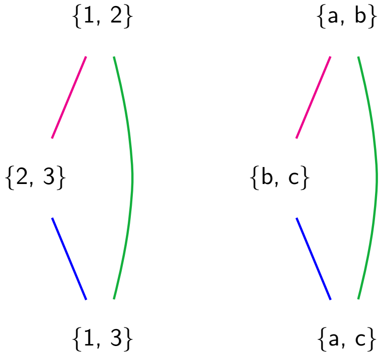
Joint work with Samuel Jaques.

Our contributions:

- Cost analysis of quantum circuits for Tani's algorithm.
- New data structure for Johnson graph vertices.
- Software to cost SIKE parameters.
- Raised issues with the pervasive assumption of zero-cost quantum storage.

Tani's algorithm

Quantum algorithm to find a (unique) claw between $f, g : [n] \to X$.

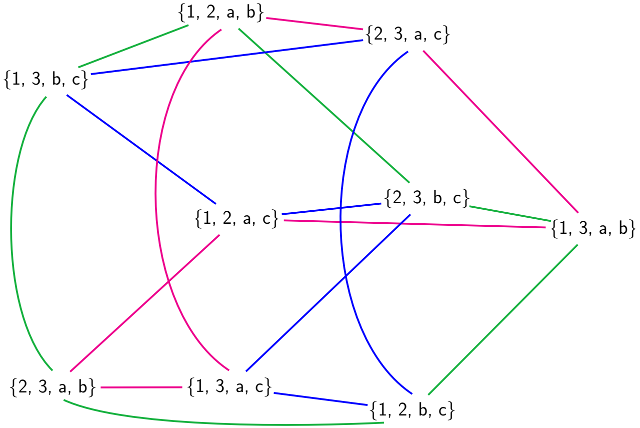


A pair of Johnson graphs

$J(\{1,2,3\},2)$ $\qquad$ $J(\{a,b,c\},2)$

Tani's algorithm

Quantum algorithm to find a (unique) claw between $f, g : [n] \to X$.



The product of Johnson graphs

$$J(\{1,2,3\},2) \times J(\{a,b,c\},2)$$

Tani's algorithm

Quantum algorithm to find a (unique) claw between $f, g : [n] \to X$.

Subroutines:

- Setup: construct Johnson graph vertices $\{(x_1, f(x_1)), ..., (x_r, f(x_r))\}$ and $\{(y_1, g(y_1)), ..., (y_r, g(y_r))\}$
- Update: walk on product of Johnson graphs.
- Check: look for claws, $f(x_i) = g(y_j)$.

Cost (Magniez–Nayak–Roland–Santha):

$$\tilde{O}\left(\text{Setup} + \frac{n}{\sqrt{r}} \cdot \text{Update} + \sqrt{r} \cdot \text{Check}\right).$$

If function evaluations are expensive, then the optimum is $r = n^{2/3}$

... but *data structure operations can be expensive.*

Tani's algorithm

Quantum algorithm to find a (unique) claw between $f, g : [n] \to X$.

Subroutines:

- Setup: construct Johnson graph vertices $\{(x_1, f(x_1)), ..., (x_r, f(x_r))\}$ and $\{(y_1, g(y_1)), ..., (y_r, g(y_r))\}$
- Update: walk on product of Johnson graphs.
- Check: look for claws, $f(x_i) = g(y_j)$.

Cost (Magniez–Nayak–Roland–Santha):

$$\tilde{O}\left(\text{Setup} + \frac{n}{\sqrt{r}} \cdot \text{Update} + \sqrt{r} \cdot \text{Check}\right).$$

If function evaluations are expensive, then the optimum is $r = n^{2/3}$

...but *data structure operations can be expensive.*

Tani's algorithm

Quantum algorithm to find a (unique) claw between $f, g : [n] \to X$.

Subroutines:

- Setup: construct Johnson graph vertices $\{(x_1, f(x_1)), ..., (x_r, f(x_r))\}$ and $\{(y_1, g(y_1)), ..., (y_r, g(y_r))\}$
- Update: walk on product of Johnson graphs.
- Check: look for claws, $f(x_i) = g(y_j)$.

Cost (Magniez–Nayak–Roland–Santha):

$$\tilde{O}\left(\text{Setup} + \frac{n}{\sqrt{r}} \cdot \text{Update} + \sqrt{r} \cdot \text{Check}\right).$$

If function evaluations are expensive, then the optimum is $r = n^{2/3}$
. . . but *data structure operations can be expensive*.

Johnson vertex data structure

Data structure requirements:

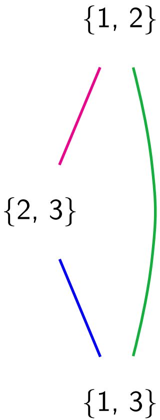
- Store a subset of a fixed n element set.
- Insertion, deletion, membership, relation counting, uniform sampling.
- History independence.

Previous approaches:

- 2004 Ambainis: Hash table + skip list.
- 2013 Bernstein–Jeffery–Lange–Meurer: Radix tree.

Our approach: Flat sorted array.

Previous approaches rely on "random access gates". We achieve a lower gate count in the standard circuit model by not treating memory as a black box.



SIKE Parameters

First round submission

	k	2^{k-1}	$\min(\sqrt{2^{e_2}}, \sqrt{3^{e_3}})$	$\sqrt{2^k}$	$\min(\sqrt[3]{2^{e_2}}, \sqrt[3]{3^{e_3}})$
SIKEp503	128	2^{127}	$1.00 \cdot 2^{125}$	2^{64}	$1.26 \cdot 2^{83}$
SIKEp751	192	2^{191}	$1.00 \cdot 2^{186}$	2^{96}	$1.00 \cdot 2^{124}$
SIKEp964	256	2^{255}	$1.45 \cdot 2^{238}$	2^{128}	$1.02 \cdot 2^{159}$

Recall: resources for golden collision search with $n = 2^{128}$.

- **Tani**: $p = 1$ large quantum processor with $m = O(n^{2/3}) \approx 2^{85}$ qubits, and time for $t = O(n^{2/3}) \approx 2^{85}$ sequential quantum walk steps.

SIKE Parameters

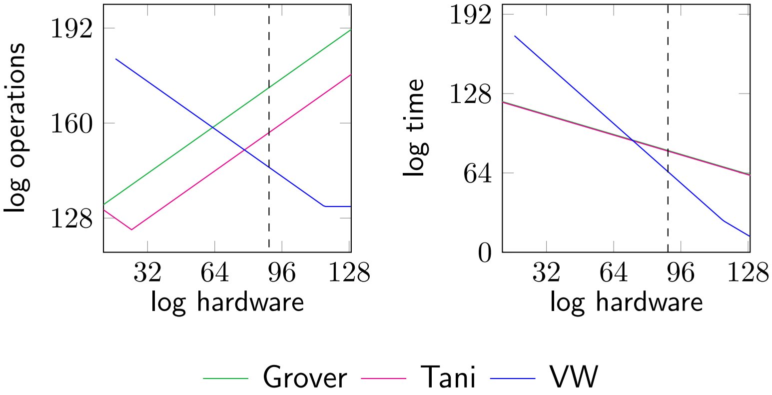
First round submission

	k	2^{k-1}	$\min(\sqrt{2^{e_2}}, \sqrt{3^{e_3}})$	$\sqrt{2^k}$	$\min(\sqrt[3]{2^{e_2}}, \sqrt[3]{3^{e_3}})$
SIKEp503	128	2^{127}	$1.00 \cdot 2^{125}$	2^{64}	$1.26 \cdot 2^{83}$
SIKEp751	192	2^{191}	$1.00 \cdot 2^{186}$	2^{96}	$1.00 \cdot 2^{124}$
SIKEp964	256	2^{255}	$1.45 \cdot 2^{238}$	2^{128}	$1.02 \cdot 2^{159}$

Recall: resources for golden collision search with $n = 2^{128}$.

- **Tani**: $p = 1$ large quantum processor with $m = O(n^{2/3}) \approx 2^{85}$ qubits, and time for $t = O(n^{2/3}) \approx 2^{85}$ sequential quantum walk steps.

Available tradeoffs between time, gates, and hardware



- Tani's algorithm does not achieve cost $n^{2/3}$.
- VW wins under reasonable depth constraints.
- Low memory "dip" relies on zero-cost quantum storage.

Revised parameters

Second round submission

	Target level	Classical gate requirement [38]	Classical security estimates		
			Total time [1] memory 2^{80} units	Gates [21, Fig. 4(d)] memory 2^{96} bits	x64 instructions [9] memory 2^{80} units
SIKEp434	1	143	128	142	143
SIKEp503	2	146	152	169*	169*
SIKEp610	3	207	189	209	210
SIKEp751	5	272	-	263*	262

Also influenced by new cost analysis of VW:

- 2018 Adj–Cervantes-Vázquez–Chi-Domínguez–Menezes–Rodríguez-Henríquez.
- 2019 Costello–Longa–Naehrig–Renes–Virdia

Chapter 3: NTRU / LWE and near neighbor search

Joint work with Martin Albrecht, Vlad Gheorghiu, and Eamonn Postlethwaite.

Contributions

- Software to optimize "near neighbor search" algorithms parameters.
- Leading constants for a special case of "filtered quantum search".
- Analysis of "popcount filter".

Chapter 3: NTRU / LWE and near neighbor search

Joint work with Martin Albrecht, Vlad Gheorghiu, and Eamonn Postlethwaite.

Contributions

- Software to optimize "near neighbor search" algorithms parameters.
- Leading constants for a special case of "filtered quantum search".
- Analysis of "popcount filter".

Near neighbor search

Goal: Given a list of N points on the unit sphere in $\mathbb{R}^d$, find N pairs of points at angular distance $< \pi/3$.

What computational resources are required?

2016 **Becker–Ducas–Gama–Laarhoven**
$\exp_2((0.207\ldots + o(1))d)$ bits of memory and $\exp_2((0.292\ldots + o(1))d)$ mostly parallelizable RAM operations.

or

$\exp_2((0.207\ldots + o(1))d)$ bits of qRAM and $\exp_2((0.265\ldots + o(1))d)$ qubit operations across many different Grover searches.

Near neighbor search

Goal: Given a list of N points on the unit sphere in $\mathbb{R}^d$, find N pairs of points at angular distance $< \pi/3$.

What computational resources are required?

2016 **Becker–Ducas–Gama–Laarhoven**
$\exp_2((0.207\ldots + o(1))d)$ bits of memory and
$\exp_2((0.292\ldots + o(1))d)$ mostly parallelizable RAM operations.

or

$\exp_2((0.207\ldots + o(1))d)$ bits of qRAM and
$\exp_2((0.265\ldots + o(1))d)$ qubit operations across many different Grover searches.

Barriers to a practical quantum speedup

- The asymptotic improvement is "small".
- qRAM might be more expensive than RAM.
- Error correction and other intrinsic overhead for quantum hardware.
- Effectiveness of classical heuristics, e.g. "xor and population count filter".

Barriers to a practical quantum speedup

- The asymptotic improvement is "small".
- qRAM might be more expensive than RAM.
- Error correction and other intrinsic overhead for quantum hardware.
- **Effectiveness of classical heuristics**, e.g. "xor and population count filter".

Filtered classical search

g(1)	g(2)	g(3)	g(4)	g(5)	...	g(57)
f(1)	f(2)	f(3)	f(4)	f(5)	...	f(57)

Filtered classical search

0	0	g(3)	g(4)	g(5)	...	g(57)
f(1)	f(2)	f(3)	f(4)	f(5)	...	f(57)

Filtered classical search

0	0	0	1	g(5)	...	g(57)
f(1)	f(2)	f(3)	f(4)	f(5)	...	f(57)

Filtered classical search

0	0	0	1	g(5)	...	g(57)
f(1)	f(2)	f(3)	0	f(5)	...	f(57)

Filtered classical search

0	0	0	1	0	...	g(57)
f(1)	f(2)	f(3)	0	f(5)	...	f(57)

Filtered classical search

0	0	0	1	0	...	1
f(1)	f(2)	f(3)	0	f(5)	...	f(57)

Filtered classical search

0	0	0	1	0	...	1
f(1)	f(2)	f(3)	0	f(5)	...	1

Filtered quantum search

Lemma

Let f be a predicate on $[N]$.
Let g be a filter for f with $|f \cap g| \approx 1$ and $4 < |g| < N/100$.
We can find a root of f with probability $\geq 1/14$ at a cost of

$$(0.50...) \cdot \sqrt{N} \cdot \mathsf{Cost}(g) + (0.64...) \cdot \sqrt{|g|} \cdot \mathsf{Cost}(f \cap g).$$

Note: There's a missing edge case in copy of thesis I gave you. It has been fixed only the probability of success is affected.

Software: python/mpmath package

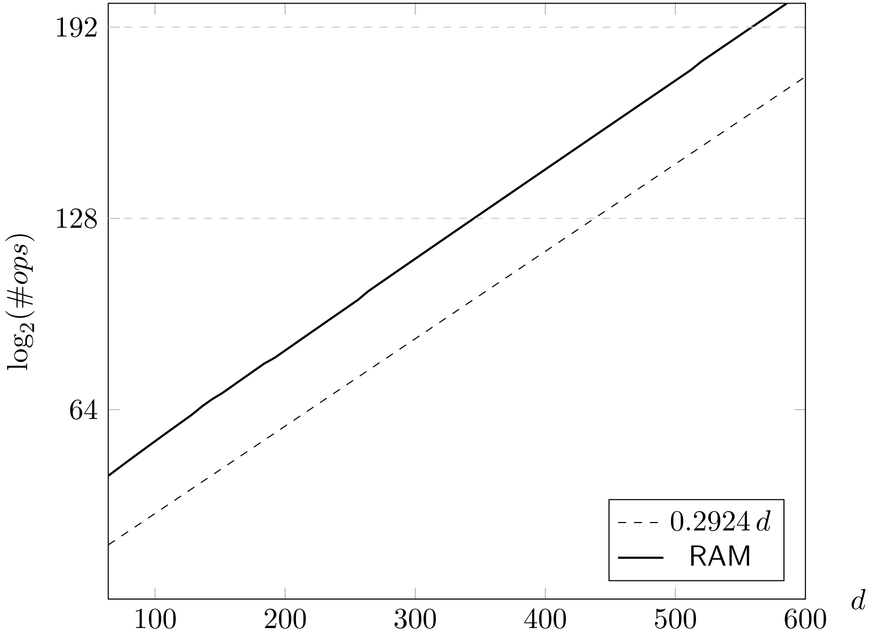
Calculates the circuit depth, width, gate count (etc.) for popcount and filtered quantum search subroutines.

Calculates the accuracy of random popcount filters given

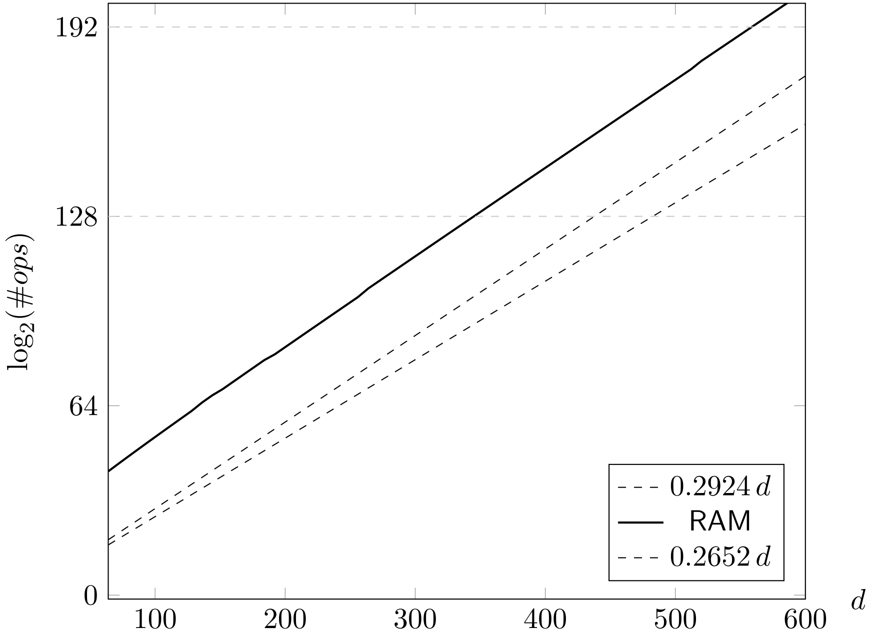
- points uniformly distributed on sphere;
- points uniformly distributed in a cap of angle β.

Calculates the (normalized) spherical measure of

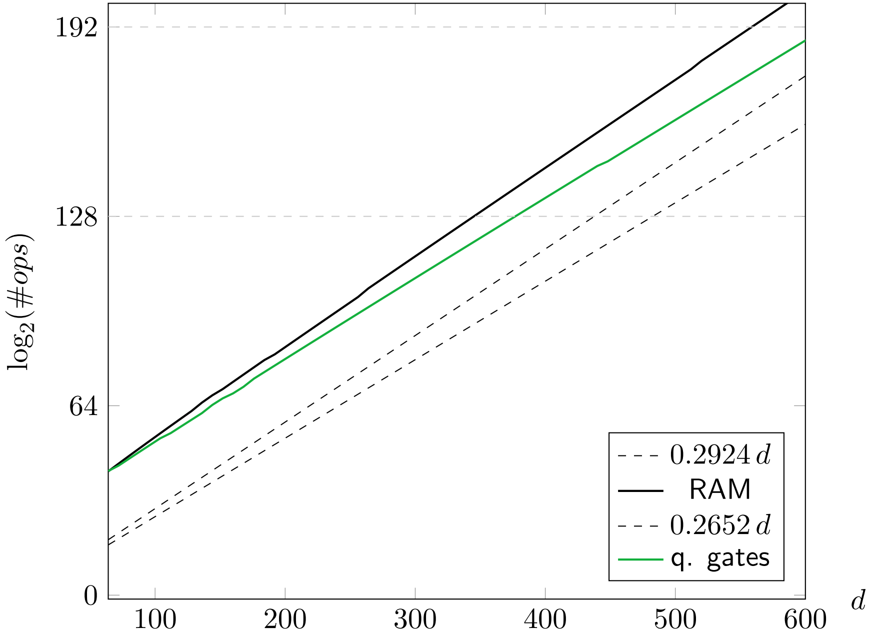
- spherical caps, using ${}_2F_1$ representation of $C_d(\theta)$
- intersections of caps, using an integral representation.



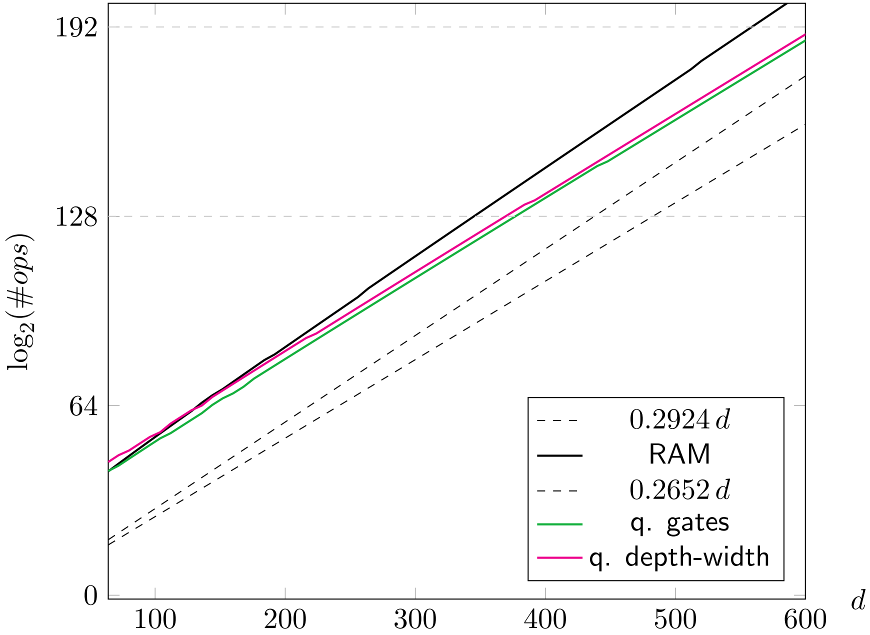
192
128
64
$\log_2(\#ops)$
100
200
300
400
500
600
d
$0.2924\,d$
RAM



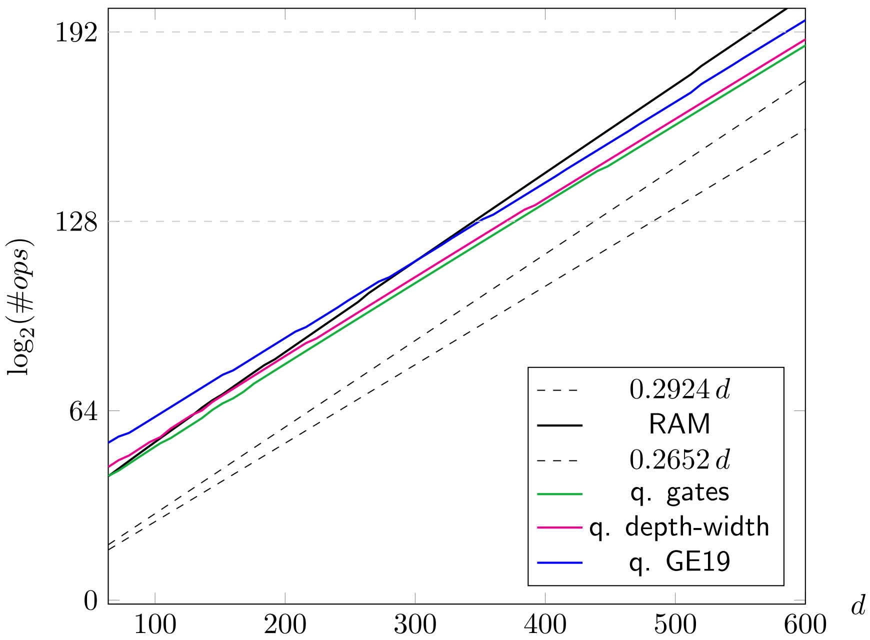
192
128
64
0
$\log_2(\#ops)$
100
200
300
400
500
600
d
$0.2924\,d$
RAM
$0.2652\,d$



192
128
64
0
$\log_2(\#ops)$
100
200
300
400
500
600
d
$0.2924\,d$
RAM
$0.2652\,d$
q. gates



$\log_2(\#ops)$
192
128
64
0
100
200
300
400
500
600
d
$0.2924\,d$
RAM
$0.2652\,d$
q. gates
q. depth-width



192
128
64
0
100
200
300
400
500
600
d
$\log_2(\#ops)$
$0.2924\,d$
RAM
$0.2652\,d$
q. gates
q. depth-width
q. GE19

Questionable assumptions and recommendations

- **Unit-cost random access memory** (Introduction)
  - Technologically motivated memory cap, e.g. 2^{140} bits.
- **Zero-cost quantum storage** (Chapter 2)
  - Replace with unit-cost quantum storage.
- **Unit-cost qRAM** (Chapter 3)
  - Technologically motivated memory cap, e.g. 2^{140} bits.
  - Skepticism. Maybe assume n-bit qRAM has latency $\Omega(n^{1/3})$.
- **Zero-cost classical computation** (Chapter 4)
  - Replace with RAM model classical computation.

Thanks!

Questionable assumptions and recommendations

- **Unit-cost random access memory** (Introduction)
  - Technologically motivated memory cap, e.g. 2^{140} bits.
- **Zero-cost quantum storage** (Chapter 2)
  - Replace with unit-cost quantum storage.
- **Unit-cost qRAM** (Chapter 3)
  - Technologically motivated memory cap, e.g. 2^{140} bits.
  - Skepticism. Maybe assume n-bit qRAM has latency $\Omega(n^{1/3})$.
- **Zero-cost classical computation** (Chapter 4)
  - Replace with RAM model classical computation.

Thanks!

Questionable assumptions and recommendations

- **Unit-cost random access memory** (Introduction)
  - Technologically motivated memory cap, e.g. 2^{140} bits.
- **Zero-cost quantum storage** (Chapter 2)
  - Replace with unit-cost quantum storage.
- **Unit-cost qRAM** (Chapter 3)
  - Technologically motivated memory cap, e.g. 2^{140} bits.
  - Skepticism. Maybe assume n-bit qRAM has latency $\Omega(n^{1/3})$.
- **Zero-cost classical computation** (Chapter 4)
  - Replace with RAM model classical computation.

Thanks!