Quantum Merging Algorithms

María Naya-Plasencia, André Schrottenloher, André Chailloux, Lorenzo Grassi

Quantum Merging Algorithms

María Naya-Plasencia[2], André Schrottenloher[2]
Joint work with André Chailloux[2] and Lorenzo Grassi[1]

[1] IAIK, Graz University of Technology, Austria
[2] Inria, France

informatiques mathématiques

European Research Council
Established by the European Commission

Outline

1. Quantum (Generalized) Collisions
2. Quantum Merging
3. Extended Quantum Merging

Quantum (Generalized) Collisions

Generalized Birthday Problem(s)

Problem 1: "original"

Given $L_1, \ldots L_\mathbf{k}$ classical lists of random **n**-bit strings, find $x_1, \ldots x_\mathbf{k} \in L_1 \times \ldots L_\mathbf{k}$ such that $x_1 \oplus \ldots \oplus x_\mathbf{k} = 0$.

Problem 2: "oracle"

Given oracle access to a random **n**-bit to **n**-bit function H, find $x_1, \ldots x_\mathbf{k}$ such that $H(x_1) \oplus \ldots \oplus H(x_\mathbf{k}) = 0$.

Problem 3: "unique solution"

Given oracle access to a random **n**/**k**-bit to **n**-bit function H, find the single **k**-tuple $x_1, \ldots x_\mathbf{k}$ such that $H(x_1) \oplus \ldots H(x_\mathbf{k}) = 0$.

Applications

Parity check problem: given $P(X)$ of degree n, find a low-weight multiple of P

Multiple-encryption: given a few plaintext-ciphertext pairs $(x, E_{k_1} \circ \ldots \circ E_{k_r}(x))$, find the independent keys $k_1, \ldots k_r$

Subset-sum: given n integers $a_0, \ldots a_{n-1}$ on poly(n) bits, find a binary $\bar{e}$ such that $\bar{a} \cdot \bar{e} = 0$

LPN: given samples $a, a \cdot s + e$ with n-bit uniform random a and Bernoulli noise e, find s

Except LPN, we have **quantum** oracle access.

Focus on Problem 2 (with oracle)

Problem 2: The "oracle" k-xor

Let $H : \{0,1\}^{\mathbf{n}} \rightarrow \{0,1\}^{\mathbf{n}}$ be a random function, find $x_1, \ldots, x_{\mathbf{k}}$ such that $H(x_1) \oplus \ldots \oplus H(x_{\mathbf{k}}) = 0$.

- We suppose that **quantum** oracle access to H is given
- We focus on the exponent in the time complexity $\widetilde{\mathcal{O}}\left(2^{\alpha_{\mathbf{k}} \mathbf{n}}\right)$
- All the results apply with $+$ instead of $\oplus$

The 1-xor problem: exhaustive search

Classically: look for a preimage of 0. Since H is random, we need to query it $\mathcal{O}(2^{\mathbf{n}})$ times.

Quantumly: use Grover's algorithm. $\mathcal{O}\left(2^{\mathbf{n}/2}\right)$ quantum queries and time.

Grover search / amplitude amplification

Find in S (of size $2^{\mathbf{n}}$) an element x (2^t solutions) such that x satisfies some condition.

$$\underbrace{2^{(\mathbf{n}-t)/2}}_{\substack{2^t \text{ solutions} \\ \text{among } 2^{\mathbf{n}}}} \left(\underbrace{\text{Sampling}}_{\text{Produce } \sum_{s \in S} |s\rangle} + \underbrace{\text{Checking}}_{\text{Test } |x\rangle} \right)$$

Interlude: Quantum Memory

The "quantum memory" landscape

	Sequential access	Quantum random access
Classical write	Classical memory sequential access **SAM**	Classical memory quantum random access **QACM (or qRAM)**
Quantum write	Quantum memory sequential access **Qubits**	Quantum memory quantum random access **QAQM**

The "quantum memory" landscape (ctd.)

In our work, we consider that answering a query "$x \in L$", **for a superposition of x**, costs:

(C)SAM: $\widetilde{\mathcal{O}}(|L|)$
QACM: $\text{poly}(\log |L|)$
Qubits: $\widetilde{\mathcal{O}}(|L|)$
QAQM: $\text{poly}(\log |L|)$

Converting QACM to SAM

We can emulate QACM queries with **classical sequential** memory accesses: perform a sequence of comparisons.

Converting QACM to SAM

On input x, to compute if $x \in L$:

- Read L sequentially;
- Run a sequence of $|L|$ comparison circuits;
- Aggregate the comparison results.

- We can make the memory in some quantum algorithms classical (however, no guarantee of a quantum speedup)

Quantum Collisions without qRAM

Joint work with André Chailloux

The 2-xor problem: collision search

Classical (naive): $\mathcal{O}\left(2^{n/2}\right)$ computations and $\mathcal{O}\left(2^{n/2}\right)$ memory.
Classical (Pollard's rho): $\mathcal{O}\left(2^{n/2}\right)$ computations and $\mathcal{O}\left(1\right)$ memory.
Quantum (BHT*): $\widetilde{\mathcal{O}}\left(2^{n/3}\right)$ computations and $\mathcal{O}\left(2^{n/3}\right)$ QACM.

BHT

- Store $2^{n/3}$ arbitrary queries $x, H(x)$ in a list L
- Search $\{0,1\}^{n}$ with the predicate:

$$f(x) = (\exists y \neq x, (y, H(x)) \in L)$$

(needs QACM)

---

* Brassard, Høyer, and Tapp, *"Quantum Cryptanalysis of Hash and Claw-Free Functions"*, LATIN 98

Quantum collisions without qRAM

In BHT, we perform $2^{\mathbf{n}/3}$ membership queries to a list L of size $2^{\mathbf{n}/3}$: the conversion increases the time up to $2^{2\mathbf{n}/3}$!

Let's try again, with:

- A smaller list
- Less membership queries

To do this, we put a constraint on L, and search for a collision in a smaller subspace.

Chailloux, Naya-Plasencia, and S., *"An Efficient Quantum Collision Search Algorithm and Implications on Symmetric Cryptography"*, ASIACRYPT 17

Quantum collisions without qRAM (ctd.)

We search only for a collision among **distinguished points**, *e.g.* x such that $H(x) = 0^{2\mathbf{n}/5} || z$ for $z \in \{0,1\}^{3\mathbf{n}/5}$.

1. Create a list L of **distinguished** $y, H(y)$
2. Grover search **among distinguished points** for a match on L

$$\underbrace{2^{\mathbf{n}/5+\mathbf{n}/5}}_{\text{Build } L} + \underbrace{2^{\mathbf{n}/5}}_{\substack{2^{-2\mathbf{n}/5} \\ \text{probability} \\ \text{of a match}}} \Bigg(\underbrace{2^{\mathbf{n}/5}}_{\substack{\text{Sample} \\ \text{distinguished} \\ \text{points}}} + \underbrace{2^{\mathbf{n}/5}}_{\text{Match } L} \Bigg) = 2^{2\mathbf{n}/5}$$

We do $2^{\mathbf{n}/5}$ accesses to a $2^{\mathbf{n}/5}$-sized memory.

Chailloux, Naya-Plasencia, and S., *"An Efficient Quantum Collision Search Algorithm and Implications on Symmetric Cryptography"*, ASIACRYPT 17

Quantum Algorithms for the (Many-solutions) k-xor Problem

Joint work with Lorenzo Grassi

Classical results for general k

To get a **k**-xor on **n** bits:

- The optimal query complexity is $\Theta(2^{\mathbf{n}/\mathbf{k}})$
- The time complexity is $\mathcal{O}\left(2^{\mathbf{n}/(1+\lfloor \log_2(\mathbf{k}) \rfloor)}\right)$ *
- Logarithmic improvements in time (but we focus on **exponents**)

* Wagner, *"A Generalized Birthday Problem"*, CRYPTO 02

Wagner's algorithm in a single slide

Merging

From two lists L_1, L_2, compute the "join" $L_1 \bowtie_{\mathbf{u}} L_2$: the pairs $x_1, x_2 \in L_1 \times L_2$ with $x_1 \oplus x_2|_{\mathbf{u}} = 0$ (partial collision on **u** bits).

All lists are presumed sorted, the time is:

$$\mathrm{MAX}\left(|L_1 \bowtie_{\mathbf{u}} L_2|, \mathrm{MIN}\left(|L_1|, |L_2|\right)\right)$$

- Wagner's algorithm is a sequence of pairwise joins
- The strategy (optimal **u**) depends on $\lfloor \log_2(\mathbf{k}) \rfloor$; we merge $2^{\lfloor \log_2(\mathbf{k}) \rfloor}$ lists

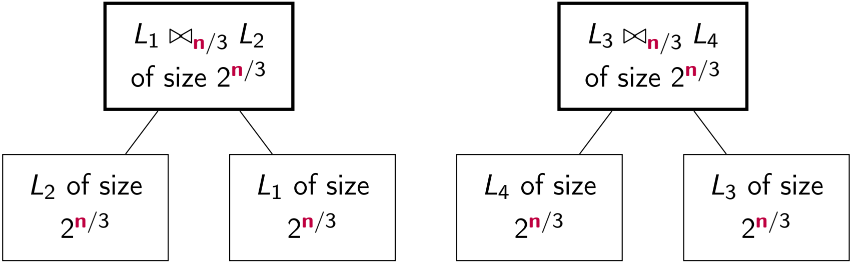
An example with $\mathbf{k} = 4$

1. Query 4 lists of $x, H(x)$: L_1, L_2, L_3, L_4 of size $2^{\mathbf{n}/3}$

L_1 of size $2^{\mathbf{n}/3}$	L_2 of size $2^{\mathbf{n}/3}$	L_3 of size $2^{\mathbf{n}/3}$	L_4 of size $2^{\mathbf{n}/3}$

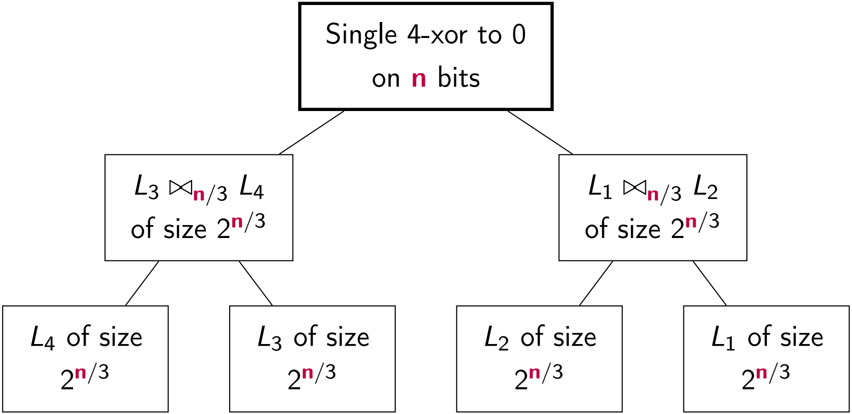
An example with $\mathbf{k} = 4$

1. Query 4 lists of $x, H(x)$: L_1, L_2, L_3, L_4 of size $2^{\mathbf{n}/3}$
2. Compute the joins $L_1 \bowtie_{\mathbf{n}/3} L_2$ and $L_3 \bowtie_{\mathbf{n}/3} L_4$ of size $2^{\mathbf{n}/3}$



An example with $\mathbf{k} = 4$

1. Query 4 lists of $x, H(x)$: L_1, L_2, L_3, L_4 of size $2^{\mathbf{n}/3}$
2. Compute the joins $L_1 \bowtie_{\mathbf{n}/3} L_2$ and $L_3 \bowtie_{\mathbf{n}/3} L_4$ of size $2^{\mathbf{n}/3}$
3. Compute the join $(L_1 \bowtie_{\mathbf{n}/3} L_2) \bowtie_{2\mathbf{n}/3} (L_3 \bowtie_{\mathbf{n}/3} L_4)$ of size 1



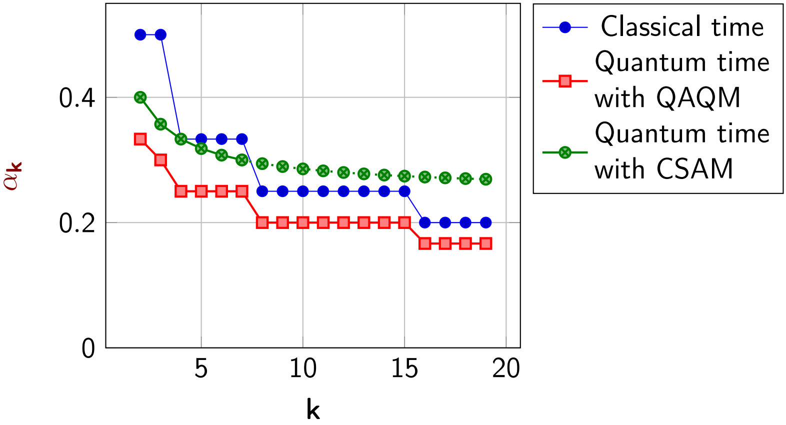
Previous quantum results on k-xor

To get a **k**-xor on **n** bits:

- The optimal query complexity is $\Theta\left(2^{\mathbf{n}/(\mathbf{k}+1)}\right)$ *
- What about the time?
  - We just saw $\mathbf{k} = 2$.
  - And for $\mathbf{k} > 2$?

* Belovs and Spalek, *"Adversary lower bound for the k-sum problem"*, ACM 13

Some new complexities



Grassi, Naya-Plasencia, and S., *"Quantum Algorithms for the k -xor Problem"*, ASIACRYPT 18

Example: 3-xor without qRAM

We use a "parallel matching" technique * with two lists.

$\ell = 2^{n/7}$

$2n/7$	$n/7$	$n/7$	$3n/7$
0	0	y_1	α_1
$\vdots$	$\vdots$	$\vdots$	$\vdots$
0	0	$y_{2^{n/7}}$	$\alpha_{2^{n/7}}$

$2n/7$	$n/7$	$n/7$	$3n/7$
0	z_1	0	β_1
$\vdots$	$\vdots$	$\vdots$	$\vdots$
0	$z_{2^{n/7}}$	0	$\beta_{2^{n/7}}$

To check a distinguished point x, match L_1 (find a partially colliding element); then match L_2.

$$2^{n/7+3n/14} + \underbrace{2^{3n/14}}_{\substack{3n/7\\ \text{remaining}\\ \text{bits}}} \left(\underbrace{2^{n/7}}_{\substack{\text{Setup}\\ \text{search}\\ \text{space}}} + \underbrace{(\underbrace{2^{n/7}}_{\text{Match } L_1} + \underbrace{2^{n/7}}_{\text{Match } L_2})}_{\text{Instead of } 2^{n/7} \times 2^{n/7}} \right) = 2^{5n/14}$$

* Naya-Plasencia, *"How to Improve Rebound Attacks"*, CRYPTO 11

Same strategy with QACM

$\ell = 2^{n/5}$

$n/5$	$n/5$	$3n/5$
0	y_1	α_1
$\vdots$	$\vdots$	$\vdots$
0	$y_{2^{n/5}}$	$\alpha_{2^{n/5}}$

$2^{n/5}$

$n/5$	$n/5$	$3n/5$
z_1	0	β_1
$\vdots$	$\vdots$	$\vdots$
$z_{2^{n/5}}$	0	$\beta_{2^{n/5}}$

$$2^{n/5+n/10} + \underbrace{2^{3n/10}}_{\substack{3n/5 \text{ bits} \\ \text{remaining}}} \left(\underbrace{1}_{\substack{\text{Matching} \\ L_1}} + \underbrace{1}_{\substack{\text{Matching} \\ L_2}} \right) = 2^{3n/10} < 2^{n/3}$$

$\Rightarrow$ 3-xor is exponentially faster than collision search (not the case classically).

Previous results

Quantum 3-xor is exponentially faster than quantum collision search.

Quantum speedup without qRAM for $\mathbf{k} \leq 7$.

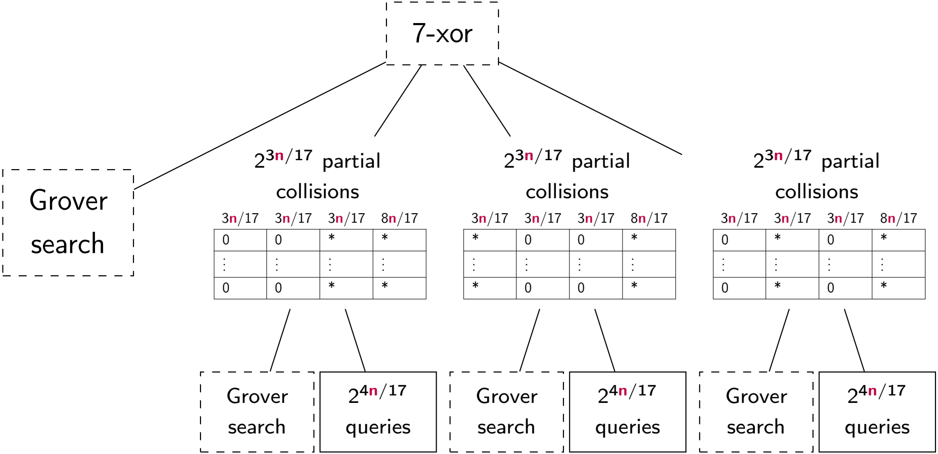
$\mathbf{k}$-xor with QAQM in time $\widetilde{\mathcal{O}}\left(2^{\mathbf{n}/(\mathbf{2}+\lfloor \log_2(\mathbf{k}) \rfloor)}\right)$ using a quantum walk.

Question: are there other improvements on the quantum version of Wagner's algorithm?

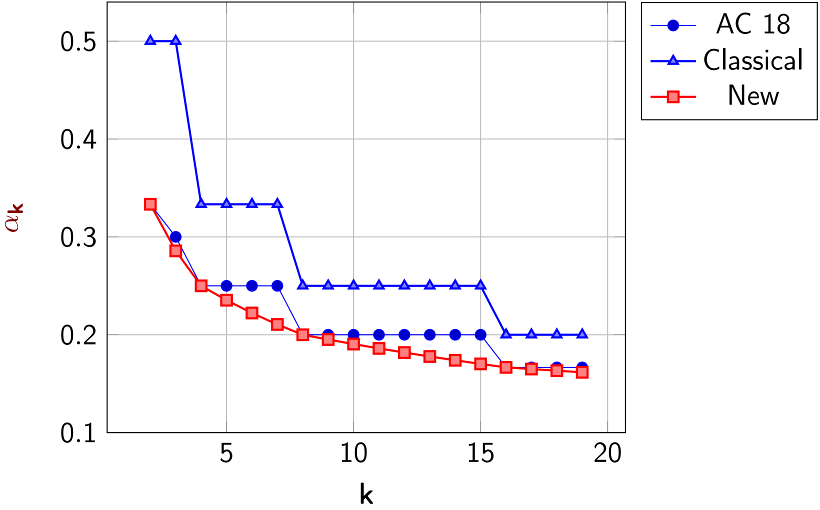
Quantum Merging

A new example: 7-xor with QACM

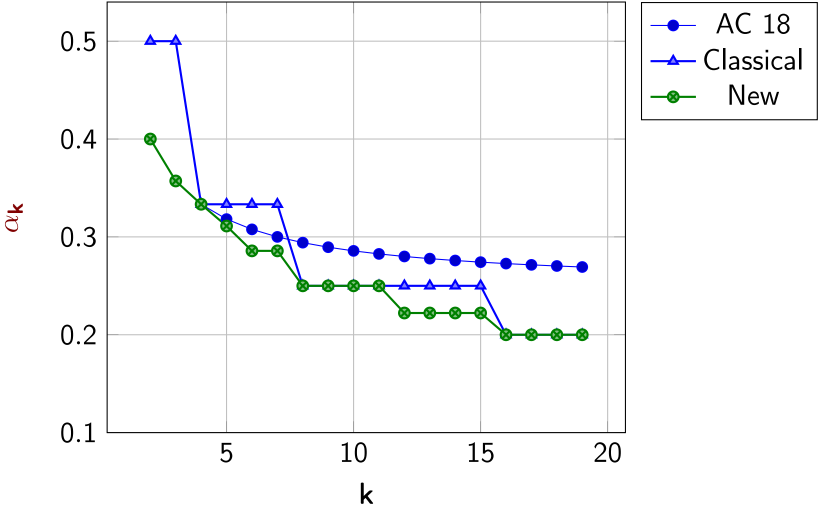
Time: $2^{4n/17} < 2^{n/4}$



Recent results (with QACM)



Recent results (with CSAM)



General strategy

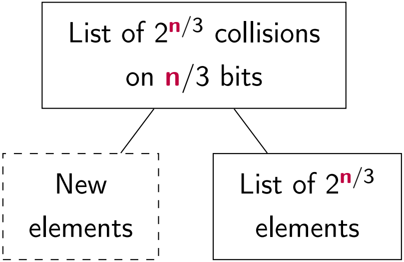
- There are several possible decompositions of the problem into subproblems.
- Each new list is the result of quantum exhaustive searches, using the previous ones for intermediate matchings.
- The optimization is a linear problem: we implemented an automatic search (MILP) for the best merging strategies.

Back to classical merging: 4-xor

Traverse the tree of merges in a depth-first manner (Wagner, CRYPTO 02): store $\lceil \log_2 \mathbf{k} \rceil$ lists instead of $\mathbf{k}$.

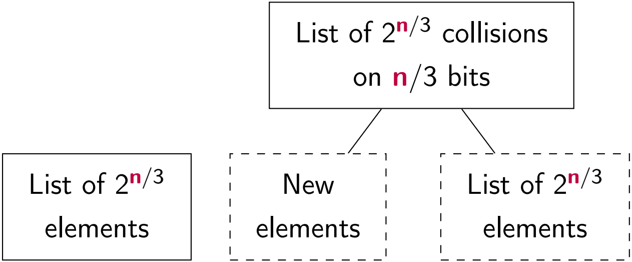
Back to classical merging: 4-xor

1. Store a list of $2^{n/3}$ elements and make new queries to produce a list of $2^{n/3}$ collisions.



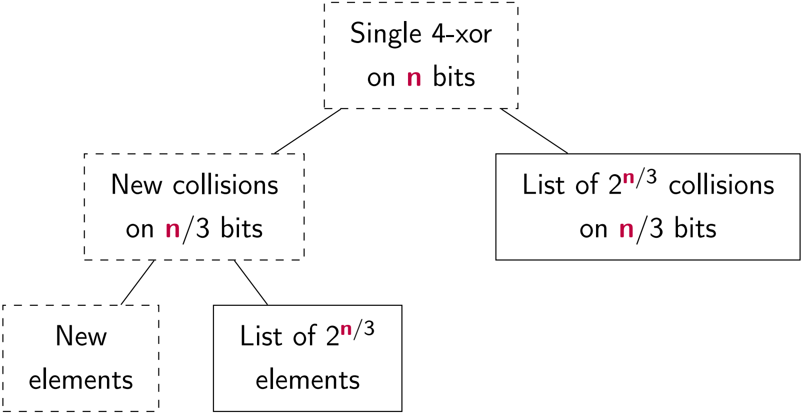
Back to classical merging: 4-xor

2. Store a new list of $2^{n/3}$ elements.

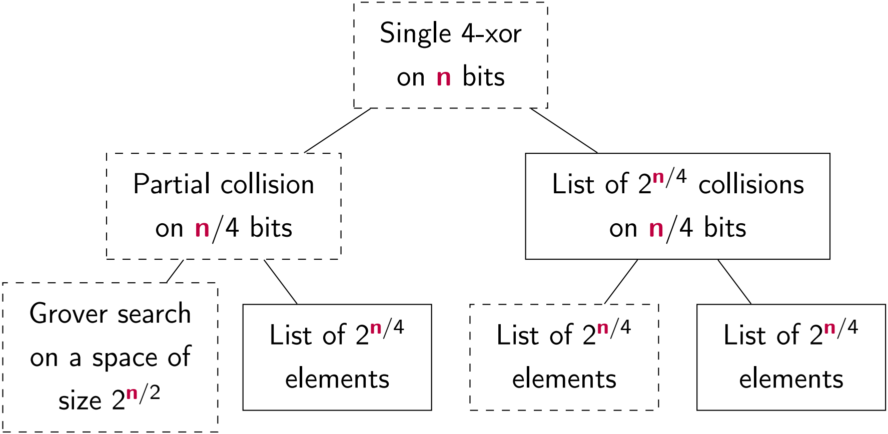


Back to classical merging: 4-xor

3. Make new queries to produce: partial $\mathbf{n}/3$-bit collisions at level 1, then (maybe) a full 4-xor at level 0.

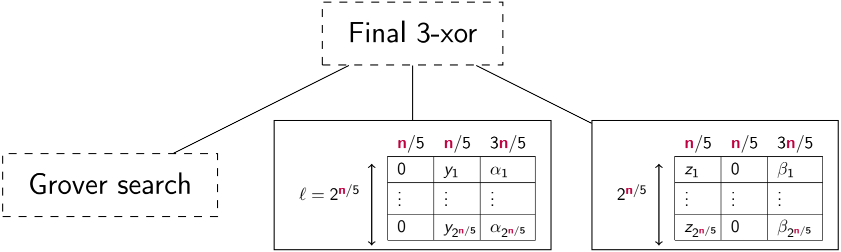


Re-optimizing this example



Rephrasing previous algorithms

- 3-xor algorithms with two intermediate lists: trees of height 2.
- We see on this example that the quantum merging trees can be more than binary.



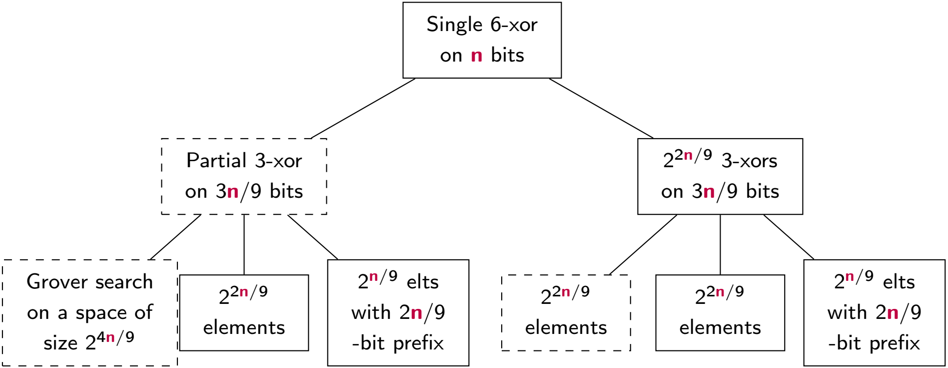
Theorem – with QACM

Theorem

If $\mathbf{k} \geq 2$ and $\kappa = \lfloor \log_2(\mathbf{k}) \rfloor$, the best merging-tree quantum time exponent is

$$\alpha_{\mathbf{k}} = \frac{2^\kappa}{(1+\kappa)2^\kappa + \mathbf{k}} \quad .$$

A complicated example



- Time complexity: $\widetilde{\mathcal{O}}\left(2^{2\mathbf{n}/9}\right) < \widetilde{\mathcal{O}}\left(2^{\mathbf{n}/4}\right)$
- Memory complexity: $\widetilde{\mathcal{O}}\left(2^{2\mathbf{n}/9}\right)$ (QACM)

Theorem – with CSAM

Theorem

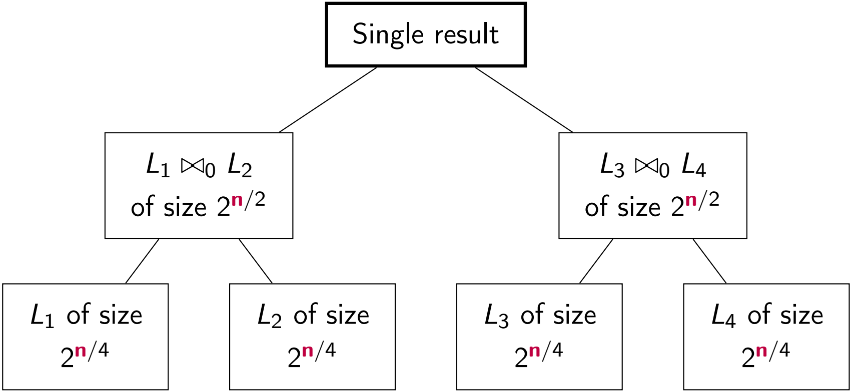
If $\mathbf{k} > 2, \mathbf{k} \neq 3, 5$ and $\kappa = \lfloor \log_2(\mathbf{k}) \rfloor$, the best merging-tree quantum time exponent is:

$$\alpha_{\mathbf{k}} = \frac{1}{\kappa+1} \text{ if } \mathbf{k} < 2^{\kappa} + 2^{\kappa-1} \text{ or } \alpha_{\mathbf{k}} = \frac{2}{2\kappa+3} \text{ if } \mathbf{k} \geq 2^{\kappa} + 2^{\kappa-1}$$

Extended Quantum Merging

Merging with a single solution

All merges become trivial (no prefix): this is a simple collision search in time $\widetilde{\mathcal{O}}\left(2^{n/2}\right)$ and memory $\widetilde{\mathcal{O}}\left(2^{n/2}\right)$. **Merging is not enough!**



Classical "extended" merging

We merge on an arbitrary prefix s (not 0), and we repeat the computation for all values of s.

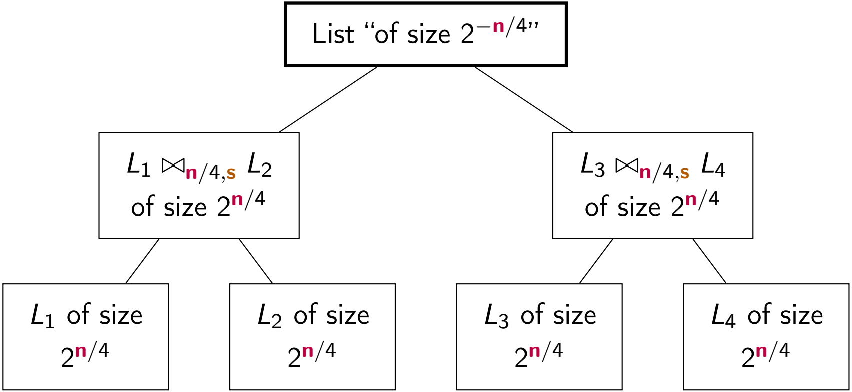
- Subsumes Schroeppel and Shamir's 4-list algorithm (next slide) and the Dissection technique
- Classically, this saves memory
- Quantumly, this reduces in addition **the time complexity**

Schroeppel and Shamir, *"A $\mathrm{T} = O(2^{n/2})$, $\mathrm{S} = O(2^{n/4})$ Algorithm for Certain NP-Complete Problems"*, SIAM 81

Dinur, Dunkelman, Keller, and Shamir, *"Efficient Dissection of Composite Problems, with Applications to Cryptanalysis, Knapsacks, and Combinatorial Search Problems"*, CRYPTO 12

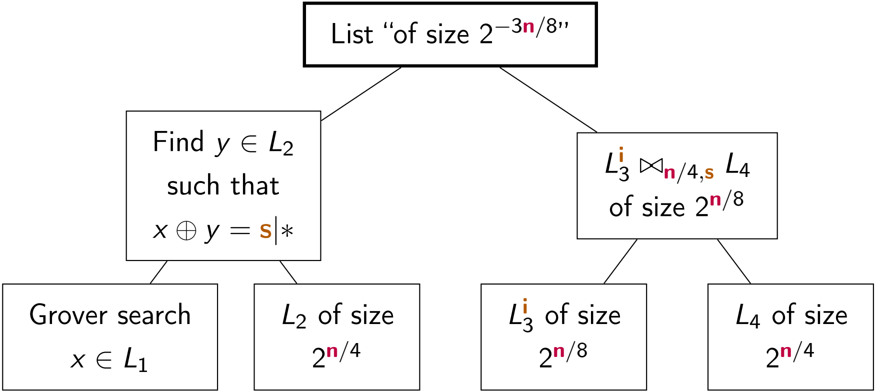
Schroeppel and Shamir's 4-list method

Loop over a chosen prefix s of $n/4$ bits. Time: $\widetilde{\mathcal{O}}\left(2^{n/2}\right)$ and memory: $\widetilde{\mathcal{O}}\left(2^{n/4}\right)$.



From classical to quantum

We loop over $\mathbf{s}$ ($\mathbf{n}/4$ bits) and $1 \leq \mathbf{i} \leq 2^{\mathbf{n}/8}$, where $\mathbf{i}$ defines a choice of sublist: $L_3 = \bigcup_{1 \leq \mathbf{i} \leq 2^{\mathbf{n}/8}} L_3^{\mathbf{i}}$.



Time complexity of this example

$$\underbrace{2^{(3\mathbf{n}/8)/2}}_{\text{Grover: choice of } L_3' \text{ and of } \mathbf{s}} \Big(\underbrace{2^{\mathbf{n}/8}}_{\text{Computation of } L_3' \bowtie_{\mathbf{n}/4,\mathbf{s}} L_4} + \underbrace{2^{(\mathbf{n}/4)/2}}_{\text{Grover: search in } L_1 \text{ for a match}} \Big)$$

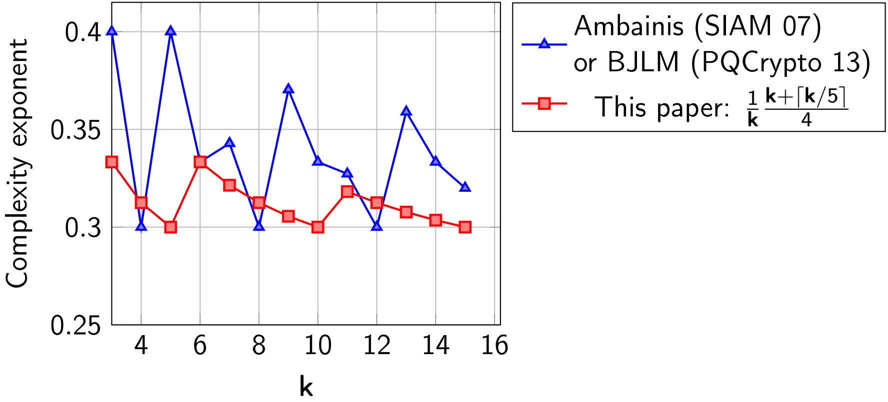
$$= 2^{5\mathbf{n}/16} = 2^{0.3125\mathbf{n}} < 2^{\mathbf{n}/3}$$

The best is $\mathbf{k} = 5$ (or a multiple):

$$\underbrace{2^{(\mathbf{n}/5)/2}}_{\text{Grover: choice of } \mathbf{s}} \Big(\underbrace{2^{\mathbf{n}/5}}_{\text{Computation of } L_4 \bowtie_{\mathbf{n}/5,\mathbf{s}} L_5} + \underbrace{2^{(2\mathbf{n}/5)/2}}_{\text{Grover: search in } L_1 \times L_2 \text{ for a match}} \Big)$$

$$= 2^{3\mathbf{n}/10} = 2^{0.3\mathbf{n}} < 2^{\mathbf{n}/3}$$

General comparison



Ambainis, *"Quantum Walk Algorithm for Element Distinctness"*, SIAM 07

Bernstein, Jeffery, Lange, and Meurer, *"Quantum Algorithms for the Subset-Sum Problem"*, PQCrypto 13

Applications

Parity-check Problem: Improved $\mathbf{k}$-list and approximate $\mathbf{k}$-list algorithms for any target weight (many or few solutions)

k-encryption: Best time complexity for $\mathbf{k} \geq 2$, time $\widetilde{\mathcal{O}}\left(2^{0.3\mathbf{n}}\right)$ for 5-encryption

Subset-sum: Best quantum time-memory product for dense knapsacks: $\widetilde{\mathcal{O}}\left(2^{5\mathbf{n}/12}\right)$ by cutting into 12 lists (prev. $0.452 > 0.412$)

LPN: Building block in the c-sum-BKW algorithm of Esser *et al.* (CRYPTO 18); ex. N_c^3 time for an 8-sum with N_c memory instead of N_c^4

Summary

- (Optimal) quantum merging strategies for **k**-xor with any number of solutions

ePrint report: 2019/501 (some code available to compute the best strategies)

Thank you.