Quantum Algorithms for the k-xor Problem

Lorenzo Grassi[1], María Naya-Plasencia[2], André Schrottenloher[2]

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

December 3, 2018

Outline

Context

The Birthday Problem

Collision search

Let $H : \{0,1\}^n \rightarrow \{0,1\}^n$ be a random function, find a collision of H, i.e a pair $x_1, x_2 \in \{0,1\}^n$ such that $H(x_1) = H(x_2)$.

- Classical queries (to L_1, L_2 or H) $\mathcal{O}\left(2^{n/2}\right)$, time $\mathcal{O}\left(2^{n/2}\right)$ and memory $\widetilde{\mathcal{O}}(1)$ (*Pollard's rho method*).
- $\Omega(2^{n/2})$ is a query lower bound.

The Generalized Birthday problem

k-xor for a random function

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

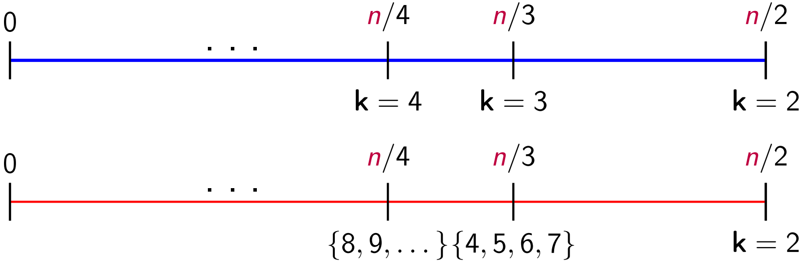
- Many applications in cryptanalysis: (R)FSB, SWIFFT...
- Applications for k-sums: $\oplus$ is replaced by modular $+$

Wagner, *"A Generalized Birthday Problem"*, 2002

Classical Results

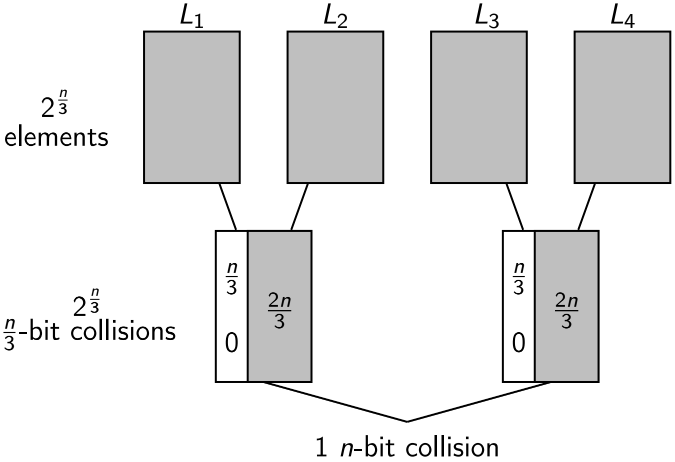
To get a **k**-xor on n bits:

- The query complexity is $\Omega(2^{n/\mathbf{k}})$
- The time complexity is $\mathcal{O}\left(2^{n/(1+\lfloor \log_2(\mathbf{k}) \rfloor)}\right)$
- The **memory** complexity is $\mathcal{O}\left(2^{n/(1+\lfloor \log_2(\mathbf{k}) \rfloor)}\right)$
- ... unless $\mathbf{k} = 2$, in which case memory is $\widetilde{\mathcal{O}}(1)$
- ... when $\mathbf{k} = 3$, logarithmic improvements are available
- ... many time-memory-query tradeoffs.



Wagner's Algorithm

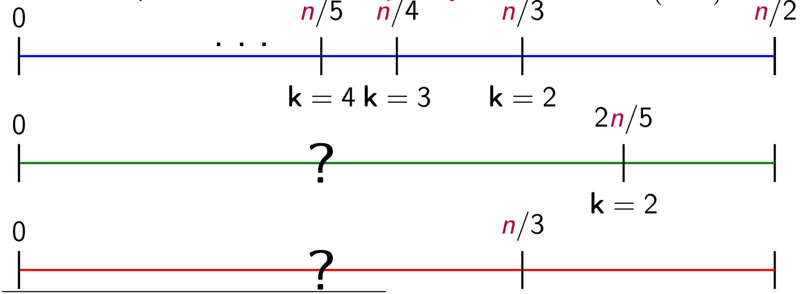
Generic method for the **k**-xor or **k**-sum with a general **k**: works at best when **k** is a power of 2.



Quantum results

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

- The query complexity is $\Omega(2^{n/(\mathbf{k}+1)})$
- With $\mathcal{O}(n)$ qubits, the time complexity for $\mathbf{k}=2$ is $\mathcal{O}\left(2^{2n/5}\right)$
- With qRAM, the time complexity for $\mathbf{k}=2$ is $\widetilde{\mathcal{O}}\left(2^{n/3}\right)$.



Brassard, Høyer, and Tapp, *"Quantum Cryptanalysis of Hash and Claw-Free Functions"*, 1998

Belovs and Spalek, *"Adversary lower bound for the k-sum problem"*, 2013

This work

We propose time-efficient quantum algorithms in two scenarios:

1. Using $\mathcal{O}(n)$ qubits;
2. Allowing read-write quantum memory in the qRAM model.

Formalization

- All elements are produced by a random function H and we access the superposition oracle O_H.
- A query to O_H costs $\mathcal{O}(1)$ time.

Results

Low-qubits scenario

- 3-xor is exponentially faster than collision search;
- A quantum time speedup (*or memory improvement*) over Wagner exists for $k \leq 7$.

qRAM scenario

- 3-xor is exponentially faster than collision search;
- k-xor can be solved in time $\widetilde{\mathcal{O}}\left(2^{n/(\mathbf{2}+\lfloor \log_2(\mathbf{k}) \rfloor)}\right)$, using $\widetilde{\mathcal{O}}\left(2^{n/(\mathbf{2}+\lfloor \log_2(\mathbf{k}) \rfloor)}\right)$ qRAM (instead of $\mathcal{O}\left(2^{n/(\mathbf{1}+\lfloor \log_2(\mathbf{k}) \rfloor)}\right)$).

Low-qubits k-xor algorithms

Quantum toolbox

Grover's algorithm

- $f \; : \{0,1\}^{n} \rightarrow \{0,1\}$ is a test function.
- We look for x such that $f(x) = 1$ (there are 2^t solutions).
- We implement f as a quantum circuit.
- With Grover: $\mathcal{O}\left(2^{(n-t)/2}\right)$ calls to f instead of 2^{n-t} classically.

- Grover improves exhaustive search by a quadratic factor **when the oracle f is fast.**

1. Testing membership with few qubits

Assume that L_1 and L_2 of sizes ℓ each are given classically. We search x such that $\exists z_1, z_2 \in L_1 \times L_2, H(z_1) \oplus H(z_2) \oplus H(x) = 0$.

- Grover requires $\sqrt{2^n/\ell^2}$ iterations.
- How to test if x is good?

Grover's test

- The lists are known classically.
- But the oracle question is asked for a **superposition** of x.
- A solution is to compare sequentially: ℓ^2 n-bit comparisons.

Chailloux, Naya-Plasencia, and Schrottenloher, *"An Efficient Quantum Collision Search Algorithm and Implications on Symmetric Cryptography"*, 2017

2. Distinguished solution strategy

We take specific L_1 and L_2: images are prefixed by $\frac{n}{2}$ zeroes.

$2^{n/8}$

$n/2$	$n/2$
0	α_1
$\vdots$	$\vdots$
0	$\alpha_{2^{n/8}}$

$2^{n/8}$

$n/2$	$n/2$
0	β_1
$\vdots$	$\vdots$
0	$\beta_{2^{n/8}}$

- We only need to search for a "distinguished solution" (with the same prefix): we compare pairs less often;
- Producing the lists costs $2^{n/4} \times 2^{n/8} = 2^{3n/8}$ queries and as much for searching x.

Collision: $2^{\frac{1}{2}\cdot\frac{2n}{5}+\frac{n}{5}}+2^{\frac{n}{5}}\left(2^{\frac{1}{2}\cdot\frac{2n}{5}}+2^{\frac{n}{5}}\right)$ and 3-xor: $2^{\frac{1}{2}\cdot\frac{n}{2}+\frac{n}{8}}+2^{\frac{n}{8}}\left(2^{\frac{1}{2}\cdot\frac{n}{2}}+2^{\frac{n}{4}}\right)$

3. Merging technique

We take more specific L_1 and L_2 to reduce the checking cost.

$$
\ell = 2^{n/7} \updownarrow
\begin{array}{|c|c|c|c|}
\multicolumn{1}{c}{2n/7} & \multicolumn{1}{c}{n/7} & \multicolumn{1}{c}{n/7} & \multicolumn{1}{c}{3n/7} \\
\hline
0 & 0 & y_1 & \alpha_1 \\
\hline
\vdots & \vdots & \vdots & \vdots \\
\hline
0 & 0 & y_{2^{n/7}} & \alpha_{2^{n/7}} \\
\hline
\end{array}
\qquad
2^{n/7} \updownarrow
\begin{array}{|c|c|c|c|}
\multicolumn{1}{c}{2n/7} & \multicolumn{1}{c}{n/7} & \multicolumn{1}{c}{n/7} & \multicolumn{1}{c}{3n/7} \\
\hline
0 & z_1 & 0 & \beta_1 \\
\hline
\vdots & \vdots & \vdots & \vdots \\
\hline
0 & z_{2^{n/7}} & 0 & \beta_{2^{n/7}} \\
\hline
\end{array}
$$

Now to test a distinguished point x:

- Find a partially colliding element from L_1;
- Find a partially colliding element from L_2;
- Compute the xor of the three values;
- The test costs $\mathcal{O}(\ell)$ comparisons instead of $\mathcal{O}(\ell^2)$.

Optimization and results

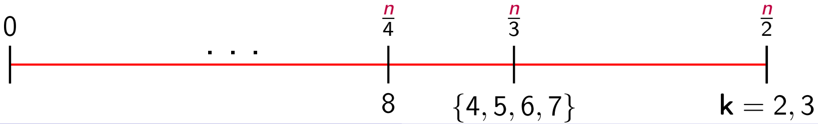
Optimizing the lists / prefix sizes leads to $\mathcal{O}\left(2^{5n/14}\right)$ time for $\mathbf{k} = 3$.

General k

The same merging method can be extended to the k-xor. Time speedup over Wagner for $\mathbf{k} = 3, 5, 6, 7$ and memory improvement for $\mathbf{k} = 4$.

Quantum low-qubits:

Classical:



k-xor algorithms with qRAM

3-xor with qRAM

qRAM is now available.

No need for a **distinguished solution** (testing membership is efficient) but the merging technique still applies.

$\Rightarrow \widetilde{\mathcal{O}}\left(2^{3n/10}\right)$ time with 2 lists of size $2^{n/5}$: better than quantum collision search.

General k

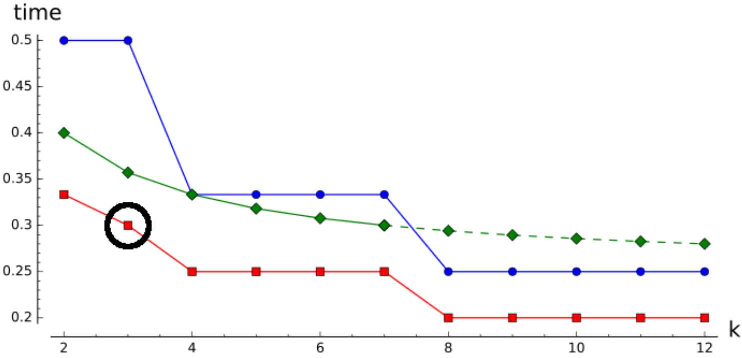
Combining:

- Wagner's method (successive lists of i-collisions with increasing zero prefixes)
- A quantum walk on the Johnson graph

We obtain a general time speedup.

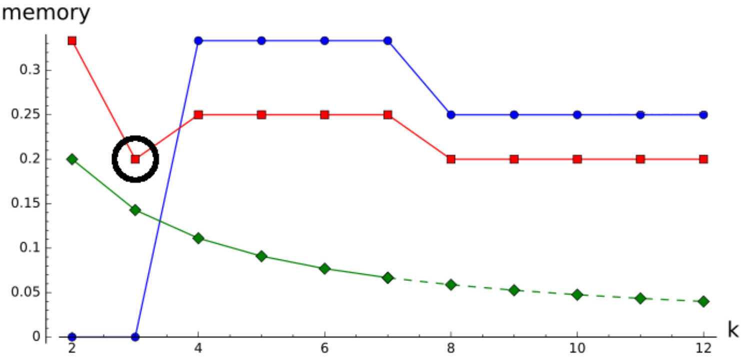
Results

- Classical time (using classical memory)
- Quantum time ($\mathcal{O}(n)$ qubits and classical memory)
- Quantum time (unbounded qRAM)



Memory

- Classical (using classical memory)
- Quantum low-qubits ($\mathcal{O}(n)$ qubits and classical memory)
- Quantum (qRAM)



Conclusion and perspectives

Conclusion

Settled

- An exponential separation between quantum collision and 3-xor (with qRAM, it goes below the quantum collision lower bound)
- With $\mathcal{O}(n)$ qubits, quantum time speedups for some **k**.
- With any **k**, a quantum time speedup using qRAM.
- This applies to **k**-sum modulo 2^n (ePrint version).

Open questions

- Can we improve the time complexity of k-xor with $\mathcal{O}(n)$ qubits, for general **k**?
- Are there other improvements when **k** is not a power of 2?

Thank you.