Quantum Merging Algorithms
María Naya-Plasencia, André Schrottenloher, André Chailloux, Lorenzo Grassi

To cite this version:
María Naya-Plasencia, André Schrottenloher, André Chailloux, Lorenzo Grassi. Quantum Merging Algorithms. Dagstuhl Seminar 19421 Quantum Cryptanalysis, Oct 2019, Dagstuhl, Germany. hal-02423380

HAL Id: hal-02423380
https://inria.hal.science/hal-02423380
Submitted on 24 Dec 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Quantum Merging Algorithms

María Naya-Plasencia\textsuperscript{2}, André Schrottenloher\textsuperscript{2}
Joint work with André Chailloux\textsuperscript{2} and Lorenzo Grassi\textsuperscript{1}

\textsuperscript{1} IAIK, Graz University of Technology, Austria
\textsuperscript{2} Inria, France
Outline

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

Quantum Merging

Extended Quantum Merging
Generalized Birthday Problem(s)

Problem 1: “original”
Given $L_1, \ldots, L_k$ classical lists of random $n$-bit strings, find $x_1, \ldots, x_k \in L_1 \times \ldots \times L_k$ such that $x_1 \oplus \ldots \oplus x_k = 0$.

Problem 2: “oracle”
Given oracle access to a random $n$-bit to $n$-bit function $H$, find $x_1, \ldots, x_k$ such that $H(x_1) \oplus \ldots \oplus H(x_k) = 0$.

Problem 3: “unique solution”
Given oracle access to a random $n/k$-bit to $n$-bit function $H$, find the single $k$-tuple $x_1, \ldots, x_k$ such that $H(x_1) \oplus \ldots H(x_k) = 0$. 
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 $H : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a random function, find $x_1, \ldots, x_k$ such that $H(x_1) \oplus \ldots \oplus H(x_k) = 0$.

- We suppose that quantum oracle access to $H$ is given
- We focus on the exponent in the time complexity $\tilde{O} (2^{\alpha_k n})$
- 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 $O(2^n)$ times.

**Quantumly:** use Grover’s algorithm. $O(2^{n/2})$ quantum queries and time.

**Grover search / amplitude amplification**

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

\[
\begin{align*}
\underbrace{2^{(n-t)/2}}_{2^t \text{ solutions among } 2^n} \left( \underbrace{\text{Sampling}}_{\text{Produce } \sum_{s \in S} |s\rangle} + \underbrace{\text{Checking}}_{\text{Test } |x\rangle} \right)
\end{align*}
\]
Interlude: Quantum Memory
<table>
<thead>
<tr>
<th></th>
<th>Sequential access</th>
<th>Quantum random access</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Classical</strong></td>
<td>Classical memory sequential access</td>
<td>Classical memory quantum random access</td>
</tr>
<tr>
<td>write</td>
<td>SAM</td>
<td>QACM (or qRAM)</td>
</tr>
<tr>
<td><strong>Quantum</strong></td>
<td>Quantum memory sequential access</td>
<td>Quantum memory quantum random access</td>
</tr>
<tr>
<td>write</td>
<td>Qubits</td>
<td>QAQM</td>
</tr>
</tbody>
</table>

The “quantum memory” landscape
In our work, we consider that answering a query “$x \in L$”, for a superposition of $x$, costs:

- **(C)SAM:** $\tilde{O}(|L|)$
- **QACM:** $\text{poly}(\log |L|)$
- **Qubits:** $\tilde{O}(|L|)$
- **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): $\mathcal{O}(2^{n/2})$ computations and $\mathcal{O}(2^{n/2})$ memory.
Classical (Pollard’s rho): $\mathcal{O}(2^{n/2})$ computations and $\mathcal{O}(1)$ memory.
Quantum (BHT*): $\tilde{\mathcal{O}}(2^{n/3})$ computations and $\mathcal{O}(2^{n/3})$ 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)

Quantum collisions without qRAM

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

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 $H(x) = 0^{2n/5}||z$ for $z \in \{0, 1\}^{3n/5}$.

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

\[
\begin{align*}
\left(2^{n/5} + 2^{n/5} \cdot 2^{-2n/5}\right) &= 2^{2n/5} \\
\left(2^{n/5} + 2^{n/5} \cdot 2^{-2n/5}\right) &= 2^{2n/5} \\
\text{probability of a match} &+ \text{sample distinguished points} \\
\text{Match } L &+ \text{match } L
\end{align*}
\]

We do $2^{n/5}$ accesses to a $2^{n/5}$-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 $\Theta(2^{n/k})$
- The time complexity is $O\left(2^{n/(1+\lceil\log_2(k)\rceil)}\right)$ *
- 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” $L_1 \bowtie_u L_2$: the pairs $x_1, x_2 \in L_1 \times L_2$ with $x_1 \oplus x_2|_u = 0$ (partial collision on $u$ bits).

All lists are presumed sorted, the time is:

$$\text{MAX (}|L_1 \bowtie_u L_2|, \text{MIN (}|L_1|, |L_2|))$$

- Wagner’s algorithm is a sequence of pairwise joins
- The strategy (optimal $u$) depends on $\lfloor \log_2(k) \rfloor$; we merge $2^{\lfloor \log_2(k) \rfloor}$ lists
An example with $k = 4$

1. Query 4 lists of $x, H(x)$: $L_1, L_2, L_3, L_4$ of size $2^{n/3}$
An example with $k = 4$

1. Query 4 lists of $x$, $H(x)$: $L_1, L_2, L_3, L_4$ of size $2^{n/3}$
2. Compute the joins $L_1 \Join_{n/3} L_2$ and $L_3 \Join_{n/3} L_4$ of size $2^{n/3}$
An example with $k = 4$

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

Single 4-xor to 0 on $n$ bits

$L_3 \bowtie_{n/3} L_4$ of size $2^{n/3}$

$L_1 \bowtie_{n/3} L_2$ of size $2^{n/3}$

$L_4$ of size $2^{n/3}$

$L_3$ of size $2^{n/3}$

$L_2$ of size $2^{n/3}$

$L_1$ of size $2^{n/3}$
Previous quantum results on k-xor

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

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

---

* 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 \textbf{two} lists.

\[
\ell = 2^{n/7}
\]

\[
\begin{array}{cccc}
2n/7 & n/7 & n/7 & 3n/7 \\
0 & 0 & y_1 & \alpha_1 \\
\vdots & \vdots & \vdots & \vdots \\
0 & 0 & y_{2n/7} & \alpha_{2n/7} \\
\end{array}
\hspace{1cm}
\begin{array}{cccc}
2n/7 & n/7 & n/7 & 3n/7 \\
0 & z_1 & 0 & \beta_1 \\
\vdots & \vdots & \vdots & \vdots \\
0 & z_{2n/7} & 0 & \beta_{2n/7} \\
\end{array}
\]

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

\[
2^{n/7} + 3n/14 + 2^{3n/14} \left( \underbrace{2^{n/7}}_{\text{Setup search space}} + \left( \underbrace{2^{n/7}}_{\text{Match } L_1} + \underbrace{2^{n/7}}_{\text{Match } L_2} \right) \right) = 2^{5n/14}
\]

* Naya-Plasencia, “How to Improve Rebound Attacks”, CRYPTO 11
Same strategy with QACM

\[
\ell = 2^{n/5}
\]

<table>
<thead>
<tr>
<th>(n/5)</th>
<th>(n/5)</th>
<th>(3n/5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(y_1)</td>
<td>(\alpha_1)</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>0</td>
<td>(y_{2n/5})</td>
<td>(\alpha_{2n/5})</td>
</tr>
</tbody>
</table>

\[
2^{n/5} + \frac{n}{10} + \frac{2^{3n/10}}{3n/5 \text{ bits remaining}} \left( \frac{1}{L_1} + \frac{1}{L_2} \right) = 2^{3n/10} < 2^{n/3}
\]

\[
\Rightarrow \quad \text{3-xor is exponentially faster than collision search (not the case classically).}
\]
Previous results

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

Quantum speedup without qRAM for \( k \leq 7 \).

\( k \)-xor with QAQM in time \( \tilde{O} \left( 2^{n/(2+\lfloor \log_2(k) \rfloor)} \right) \) 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: $2^{4n/17} < 2^{n/4}$

```
<table>
<thead>
<tr>
<th>3n/17</th>
<th>3n/17</th>
<th>3n/17</th>
<th>8n/17</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>3n/17</th>
<th>3n/17</th>
<th>3n/17</th>
<th>8n/17</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>0</td>
<td>0</td>
<td>*</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>*</td>
<td>0</td>
<td>0</td>
<td>*</td>
</tr>
</tbody>
</table>

```

```
<table>
<thead>
<tr>
<th>3n/17</th>
<th>3n/17</th>
<th>3n/17</th>
<th>8n/17</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>*</td>
<td>0</td>
<td>*</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>0</td>
<td>*</td>
<td>0</td>
<td>*</td>
</tr>
</tbody>
</table>

```

Grover search

```
| 2^{4n/17} queries |
```

```
| 2^{4n/17} queries |
```

```
| 2^{4n/17} queries |
```

```
| 2^{4n/17} queries |
```
Recent results (with QACM)

\[ \alpha_k \]

\[ k \]

- AC 18
- Classical
- New
Recent results (with CSAM)

\[ \alpha_r \]

\[ k \]

AC 18
Classical
New

M. Naya-Plasencia, A. Schrottenloher
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 \( \lceil \log_2 k \rceil \) lists instead of \( k \).
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.
2. Store a new list of $2^{n/3}$ elements.
3. Make new queries to produce: partial $n/3$-bit collisions at level 1, then (maybe) a full 4-xor at level 0.

- Single 4-xor on $n$ bits
- New collisions on $n/3$ bits
- New elements
- List of $2^{n/3}$ collisions on $n/3$ bits
- List of $2^{n/3}$ elements
Re-optimizing this example

- Single 4-xor on $n$ bits
- Partial collision on $n/4$ bits
- Grover search on a space of size $2^{n/2}$
- List of $2^{n/4}$ elements
- List of $2^{n/4}$ elements
- List of $2^{n/4}$ collisions on $n/4$ bits
- List of $2^{n/4}$ elements

Grover search on a space of size $2^{n/2}$
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 \( k \geq 2 \) and \( \kappa = \lfloor \log_2(k) \rfloor \), the best merging-tree quantum time exponent is

\[
\alpha_k = \frac{2^\kappa}{(1 + \kappa)2^\kappa + k}.
\]
Quantum (Generalized) Collisions
Quantum Merging
Extended Quantum Merging

A complicated example

Single 6-xor on \( n \) bits

Partial 3-xor on \( 3\frac{n}{9} \) bits

Grover search on a space of size \( 2^{\frac{4n}{9}} \)

\( 2^{\frac{n}{9}} \) elts with \( 2\frac{n}{9} \)-bit prefix

\( 2^{\frac{n}{9}} \) elements

\( 2^{\frac{2n}{9}} \) elts with \( 2\frac{n}{9} \)-bit prefix

\( 2^{\frac{2n}{9}} \) elements

\( 2^{\frac{2n}{9}} \) elements

\( 2^{\frac{n}{9}} \) elements

Time complexity: \( \widetilde{O} \left( 2^{\frac{2n}{9}} \right) < \widetilde{O} \left( 2^{\frac{n}{4}} \right) \)

Memory complexity: \( \widetilde{O} \left( 2^{\frac{2n}{9}} \right) \) (QACM)
Theorem – with CSAM

**Theorem**

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

\[
\alpha_k = \frac{1}{\kappa + 1} \quad \text{if} \quad k < 2^\kappa + 2^{\kappa - 1} \quad \text{or} \quad \alpha_k = \frac{2}{2^\kappa + 3} \quad \text{if} \quad k \geq 2^\kappa + 2^{\kappa - 1}
\]
Extended Quantum Merging
Merging with a single solution

All merges become trivial (no prefix): this is a simple collision search in time $\tilde{O}(2^{n/2})$ and memory $\tilde{O}(2^{n/2})$. **Merging is not enough!**
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 $T = O(2^{n/2})$, $S = O(2^{n/4})$ 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
Schroeppe and Shamir’s 4-list method

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

List "of size $2^{-n/4}$"

$L_1 \Join_{n/4,s} L_2$
of size $2^{n/4}$

$L_3 \Join_{n/4,s} L_4$
of size $2^{n/4}$

$L_1$ of size $2^{n/4}$

$L_2$ of size $2^{n/4}$

$L_3$ of size $2^{n/4}$

$L_4$ of size $2^{n/4}$
From classical to quantum

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

\[ 2^{(3n/8)/2} \]  

Grover: choice of \( L'_3 \) and of \( s \) 

\[ 2^{n/8} \]  

Computation of \( L'_3 \otimes_{n/4, s} L_4 \) 

\[ 2^{(n/4)/2} \]  

Grover: search in \( L_1 \) for a match 

\[ = 2^{5n/16} = 2^{0.3125n} < 2^{n/3} \]

The best is \( k = 5 \) (or a multiple):

\[ 2^{(n/5)/2} \]  

Grover: choice of \( s \) 

\[ 2^{n/5} \]  

Computation of \( L_4 \otimes_{n/5, s} L_5 \) 

\[ 2^{(2n/5)/2} \]  

Grover: search in \( L_1 \times L_2 \) for a match 

\[ = 2^{3n/10} = 2^{0.3n} < 2^{n/3} \]
General comparison

Ambainis (SIAM 07) or BJLM (PQCrypto 13)

This paper: \( \frac{1}{k} \left( \frac{k + \lceil k/5 \rceil}{4} \right) \)

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 k-list and approximate k-list algorithms for any target weight (many or few solutions)

**k-encryption:** Best time complexity for \( k \geq 2 \), time \( \tilde{O}(2^{0.3n}) \) for 5-encryption

**Subset-sum:** Best quantum time-memory product for dense knapsacks: \( \tilde{O}(2^{5n/12}) \) 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 \( N_c^4 \)
(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.