Quantum Analysis of Nested Search Problems

with Applications in Cryptanalysis

André Schrottenloher and Marc Stevens

Cryptology Group, CWI

European Research Council
Established by the European Commission

Outline

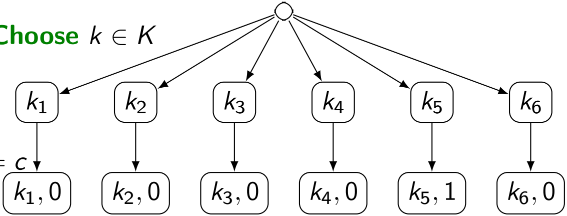
Search problems (and search trees)

- Each edge "$\to$" is a computing step
- Branching $=$ making a choice

Example: search k such that $E_k(p) = c$.

Choose $k \in K$

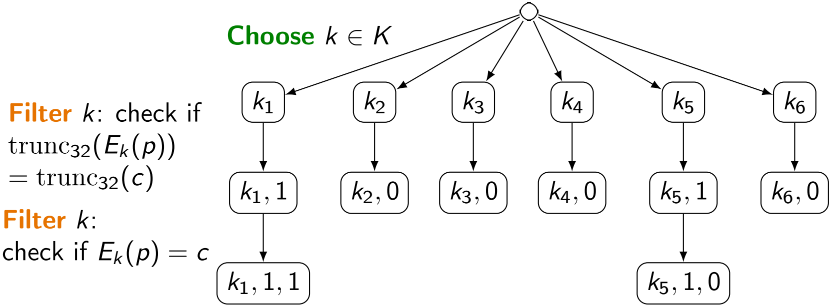
Filter k:
check if $E_k(p) = c$



$$\text{Time} = |K| \times \text{Evaluate } E_k$$

Complex search problems

In cryptanalysis, we consider complex search problems.
Example: search k such that $E_k(p) = c$, in two steps.



$$\text{Time} = |K| \times \text{Evaluate } (\mathrm{trunc}_{32} \circ E_k) + \frac{|K|}{2^{32}} \times \text{Evaluate } E_k$$

Search problems everywhere

In symmetric cryptanalysis:

- differential, linear attacks with dynamic key-guessing
- impossible differential, zero-correlation attacks
- boomerang attacks
- MITM and DS-MITM attacks . . .

are all (at least partially) exploring search trees with **choices**, **filtering** and **backtracking**.

$\Longrightarrow$ We want to turn them into **quantum attacks** using **quantum search**.

From search to quantum search

Amplitude amplification (QAA) starts from a quantum algorithm

$$\mathcal{A}\left|0\right\rangle = \alpha \underbrace{\left|\psi\right\rangle}_{\text{Good}} \left|1\right\rangle + \sqrt{1-\alpha^2} \underbrace{\left|\phi\right\rangle}_{\text{Bad}} \left|0\right\rangle$$

succeeding with probability α^2 and **amplifies** this to $\simeq 1$ with $\mathcal{O}\left(1/\alpha\right)$ iterates of $\mathcal{A}$. What it looks like:

$$\underbrace{(\mathcal{A}O_0\mathcal{A}^\dagger O)}_{\text{QAA iterate}} \cdots (\mathcal{A}O_0\mathcal{A}^\dagger O)\mathcal{A}\left|0\right\rangle$$

After t iterates:

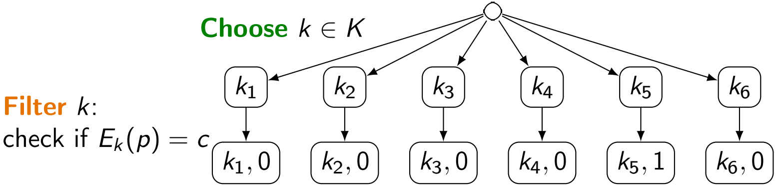
$$\nu\left|\psi\right\rangle\left|1\right\rangle + \sqrt{1-\nu^2}\left|\phi\right\rangle\left|0\right\rangle$$

where

$$\nu = \sin\left[(2t+1)\arcsin\left[\alpha\right]\right]$$

Brassard, Høyer, Mosca, Tapp, "Quantum amplitude amplification and estimation", Contemp. math. 2002

From search to quantum search (ctd.)



By iterating the classical operation "**choose** and **filter**" until it succeeds, classical search succeeds in time:

$$\text{Time} = |K| \times (\textbf{Choose} + \textbf{Filter})$$

By iterating a **quantum algorithm** for "**choose** and **filter**", QAA succeeds in time:

$$\text{Time} = \sqrt{|K|} \times (\textbf{Choose} + \textbf{Filter})$$

QAAs all the way down

QAA is a quantum algorithm, so we can run a QAA inside a QAA.

Folklore conversion lemma

- Any classical nested search algorithm can be converted into a quantum search algorithm.
- The quantum complexity is obtained by replacing "Iterations" with "$\sqrt{\text{Iterations}}$"

The problem(s)

1. Rewrite classical attacks in a way that facilitates the "conversion" (partially solved)

2. Compute **exact** complexities for the quantum algorithms.
   - So far handled on a case-by-case basis with lots of technical analysis.
   - Our goal is to externalize this work using a generic framework & analysis.

Results

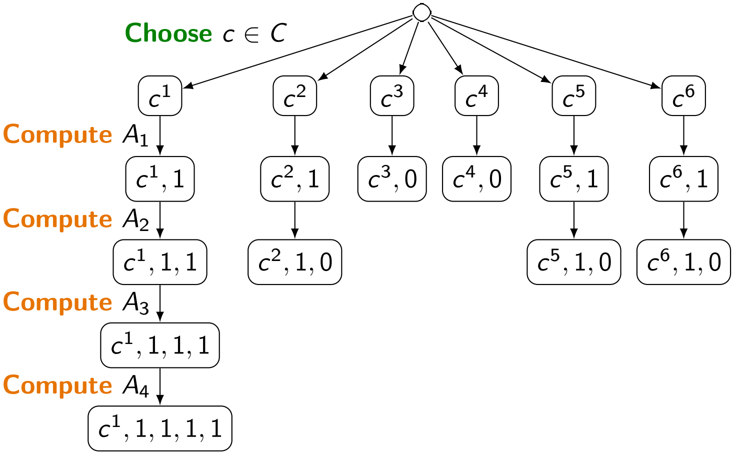
Two quantum algorithms for nested search:

- search with early aborts
- search with backtracking

with exact complexity analysis and optimizations of previous attacks.

Search with Early Aborts

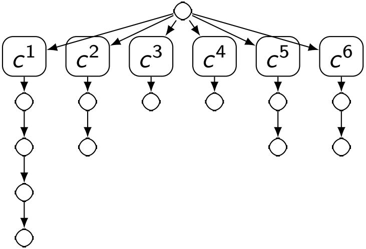
Restricted setting: search with early aborts



- Start from a single choice space C
- Search $c \in C$ that passes all filters: $A_4 \circ A_3 \circ A_2 \circ A_1(c) = 1$
- Many problems where we can "try a few bits first"

Generic idea

- Assume that A_i are implemented by quantum algorithms $\mathcal{A}_i$
- Each A_i creates a new "flag"
- Let β_i^2 = prob. of passing step i if we passed step $i-1$



$\beta_1^2 = 4/6$
$\beta_2^2 = 1/4$
$\beta_3^2 = 1/1$
$\beta_4^2 = 1/1$

There are ℓ levels. We create ℓ quantum algorithms $\mathcal{B}_i$ such that:

$$\mathcal{B}_i |0\rangle = \nu_i^2 |\psi_i\rangle |1\rangle + \sqrt{1-\nu_i^2} |\phi_i\rangle |0\rangle$$

where ν_i^2 is the **probability of success** and $|\psi_i\rangle$ is the unif. superposition of choices passing step i.

Generic idea (ctd.)

$\mathcal{B}_1$ is a QAA with k_1 iterates. The amplified algorithm $\mathcal{A}_1'$ does:

1. choose $c \in C$ unif. at random
2. apply $\mathcal{A}_1$

$$\beta_1^2 \left|\psi_1\right\rangle \left|1\right\rangle + \sqrt{1 - \beta_1^2} \left|\phi_1\right\rangle \left|0\right\rangle$$

So it has a success probability β_1^2 which we amplify with k_1 iterates to ν_1^2:

$$\nu_1 = \sin\left[(2k_1 + 1) \arcsin \beta_1\right]$$

Generic idea (ctd.)

We define $\mathcal{B}_i$ by a QAA with k_i iterates, which amplifies the algorithm:

1. Apply $\mathcal{B}_{i-1}$
2. Apply $\mathcal{A}_i$

By assumption:

$$\mathcal{A}_i \mathcal{B}_{i-1} |0\rangle = \nu_{i-1} \beta_{\mathrm{i}} |\psi_i\rangle |1\rangle + (\ldots) |0\rangle$$

so after amplifying with k_i iterates we get:

$$\nu_i = \sin\left[(2k_i + 1) \arcsin\left[\beta_{\mathrm{i}} \nu_{i-1}\right]\right]$$

Bounding the probability

Example: for 3 levels with k_1, k_2, k_3 iterates:

$$\sin\left[(2k_3+1)\arcsin\left[\beta_3 \sin\left[(2k_2+1)\arcsin\left[\beta_2 \sin\left[(2k_1+1)\arcsin\beta_1\right]\right]\right]\right]\right]$$

$$\mathrm{Time} = (2k_3+1)\Big((2k_2+1)\Big((2k_1+1)\mathrm{Time}(\mathcal{A}_1)+\mathrm{Time}(\mathcal{A}_2)\Big)+\mathrm{Time}(\mathcal{A}_3)\Big)$$

- For small x, $\sin x \simeq x \simeq \arcsin x$
- So if all probabilities remain "small":
  $$\nu_3 \simeq (2k_3+1)\beta_3(2k_2+1)\beta_2(2k_1+1)\beta_1$$
- Of course this shouldn't be "too small": the balance lies at $\mathcal{O}(1/\ell)$.

Bounding the probability (ctd.)

If $\forall i, \prod_{j=1}^{i} \left((2k_j+1)^2 \beta_j^2\right) \leq \frac{4}{\pi^2 \ell}$, then: $\nu_\ell^2 \geq \frac{1}{5} \prod_{j=1}^{\ell} \left((2k_j+1)^2 \beta_j^2\right)$.

To be read as:

If intermediate probabilities of success ($\simeq \prod_{j=1}^{i} \left((2k_j+1)^2 \beta_j^2\right)$) are of order $\frac{1}{\ell}$ then the final probability of success is of order $\frac{1}{\ell}$.

- We can amplify on top of this using $\mathcal{O}\left(\sqrt{\ell}\right)$ iterates, to a constant prob. of success.
- ℓ levels of QAA multiply the complexity by $\sqrt{\ell}$ (w.r.t. the exact square root)

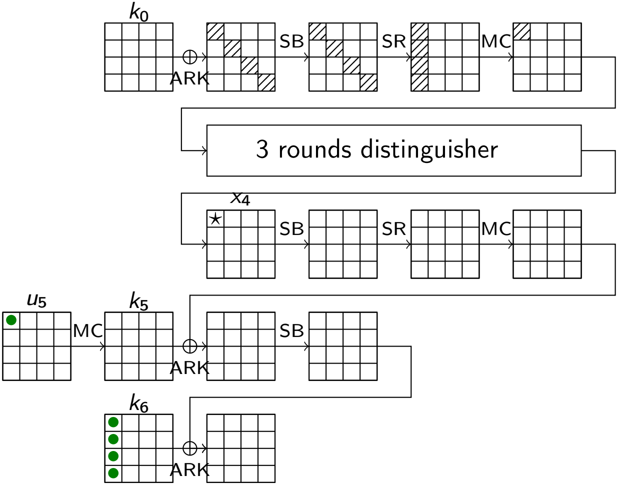
Applications

1. we know (good bounds on) the β_i
$\Longrightarrow$ minimize numerically Complexity$(\mathcal{B}_\ell)/\nu_\ell^2$ in function of k_i.

2. we don't know anything on the β_i
$\Longrightarrow$ variable-time QAA (in the paper)

Search with Backtracking

Example: 6-round AES Square attack



- Encrypt multiple structures of 2^{32} plaintexts (main diagonal varies)
- **Find** $u_5[0], k_6[0, 1, 2, 3]$ s.t. x_4 is balanced for all structures

Example (ctd.)

For each key guess we evaluate a sum over all ciphertexts:

$$\bigoplus_i S^{-1}\left(u_5[0] \oplus a_0 S^{-1}(c_i[0] \oplus k_6[0]) \oplus a_1 S^{-1}(c_i[1] \oplus k_6[1])\right.$$
$$\left. \oplus\, a_2 S^{-1}(t_2 \oplus k_6[2]) \oplus a_3 S^{-1}(s_2 \oplus k_6[3])\right)$$

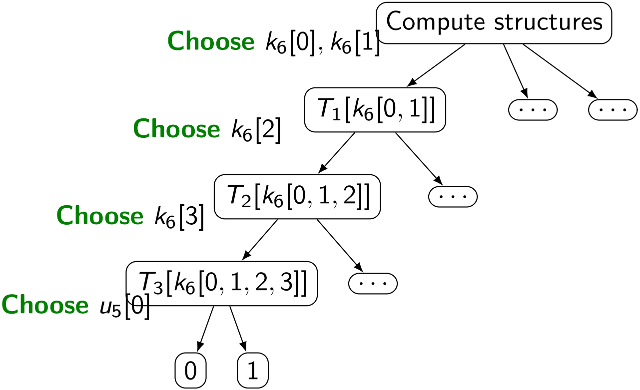
Partial sums method: store intermediate values in tables counting their number of occurrences.

- **Choose** $k_6[0], k_6[1]$. For each ciphertext c_i, compute: $(t_1, t_2, t_3) = a_0 S^{-1}(c_i[0] \oplus k_6[0]) \oplus a_1 S^{-1}(c_i[1] \oplus k_6[1]), c_i[2], c_i[3]$ and store occurrences in $T_1[k_6[0, 1]]$.
- **Choose** $k_6[2]$. For each 3-byte value (t_1, t_2, t_3), compute: $t_1 \oplus a_2 S^{-1}(t_2 \oplus k_6[2]), t_3$ and store occurrences in $T_2[k_6[0, 1, 2]]$.
- **Choose** $k_6[3]$. For each 2-byte value (s_1, s_2), compute $s_1 \oplus a_3 S^{-1}(s_2 \oplus k_6[3])$ and store occurrences in $T_3[k_6[0, 1, 2, 3]]$
- **Choose** $u_5[0]$. Compute final sum.

Ferguson, Kelsey, Lucks, Schneier, Stay, Wagner, Whiting, "Improved cryptanalysis of Rijndael", FSE 2000

Partial sums as a tree search

Example: Square attack on 6-round AES = search with backtracking.



$$\text{Time} = 2^{16} \times \left(\underbrace{\text{Compute } T_1}_{2^{32}} + 2^8 \times \left(\underbrace{T_2}_{2^{16}} + 2^8 \times \left(\underbrace{T_3}_{2^8} + 2^8 \times \underbrace{\text{sum}}_{2^8} \right) \right) \right)$$

General case

- **Tree search with backtracking** = sequence of **choices**, **filtering** and **post-processing** steps
  [Disclaimer: omitting the **filtering** in next slides]
- We must **return to previous states** to amortize the cost of each step
- Contains the attacks on AES of **[BNS19]** (Square and DS-MITM)

Bonnetain, Naya-Plasencia, S., "Quantum security analysis of AES", ToSC 2019

Sketch of the generic algorithm

- Consider a sequence of **choices** and algorithms $\mathcal{A}_i$: $\mathcal{A}_i$ depends on choice c_i and updates workspace
- A **single solution path** $c_1, \ldots, c_\ell$ s.t.: $\mathcal{A}_\ell \circ \cdots \circ \mathcal{A}_1(c_1, \ldots, c_\ell) = 1$
- New parameters: α_i^2 = probability, if $c_1, \ldots, c_{i-1}$ is the good path, that c_i extends it.

Before:

- $\mathcal{B}_i$ produces choices that pass test i
- $\mathcal{B}_i$ calls $\mathcal{B}_{i-1}$ and $\mathcal{A}_i$
- $\mathcal{B}_\ell$ solves the problem

Now:

- $\mathcal{B}_i$, **starting from the right subpath**, produces the entire solution
- $\mathcal{B}_i$ calls $\mathcal{B}_{i+1}$ and $\mathcal{A}_i$
- $\mathcal{B}_1$ solves the problem

Good news

The QAAs are nested in a reverse order, but the recursion formula is the same:

$$\nu_i = \sin\left[(2k_i + 1)\arcsin\left[\alpha_i \nu_{i+1}\right]\right]$$

It suffices to know estimates on α_i to obtain:

- an analytic formula for the success prob. and complexity (not always the best)
- a numerical optimization (always at least improving previous computations)

Results

Example: AES 6-round Square attack

- **[BNS19]**: $2^{44.73}$ AES S-Boxes, success prob. 1
- Analytic: $2^{48.70}$, success prob. 0.5
- Optimized: $2^{44.70}$, success prob. 0.94
- Square root of iterations: $2^{44.05}$

Example: AES-256 8-round DS-MITM

- **[BNS19]**: $2^{136.17}$ AES S-Boxes, success prob. 0.73
- Analytic: $2^{134.23}$, success prob. 0.5×0.73
- Optimized: $2^{132.07}$, success prob. 0.95×0.73
- Square root: $2^{131.07}$

Conclusion

The setting of our algorithms (balanced search trees, known parameters) captures most applications of QAA in cryptanalysis.

- Our formulas may be used to bound the complexities
- Numerical optimisation performs even better
- The final complexity will be **very** close to the “basic square root”

ePrint 2022/761 (new version on the way!)