A Quantum Analysis of Nested Search Problems with Applications in Cryptanalysis

André Schrottenloher and Marc Stevens

Cryptology Group, CWI, Amsterdam, The Netherlands
`firstname.lastname@cwi.nl`

Abstract. In this paper we study search problems that arise very often in cryptanalysis: nested search problems, where each search layer has known degrees of freedom as well as constraints. Classical nested searches can be transformed into quantum algorithms, using Grover's quantum search or amplitude amplification by Brassard et al., obtaining up to a square-root speedup. However, the nesting introduces technicalities in the quantum complexity analysis that are complex to handle and have been so far analyzed in previous works in a case-by-case manner. In this paper, we aim to simplify the quantum transformation and corresponding analysis.

We introduce a framework to transform classical nested searches into a quantum procedure and to analyze its complexity. The resulting quantum procedure is easier to describe and analyze compared to previous works, both in the asymptotic setting and for concrete instantiations. Its time complexity and success probability can be bounded using a generic formula, or more precisely with numerical optimization.

Along the way to this result, we introduce an algorithm for *variable-time amplitude amplification* of independent interest. It allows to obtain essentially the same asymptotic complexity as a previous algorithm by Ambainis (STACS 2012) using only several layers of amplitude amplification, and without relying on amplitude estimation.

Moreover, we present some direct applications of our results in cryptanalysis.

Keywords: Quantum search · Nested search · Quantum cryptanalysis · Amplitude amplification · Symmetric cryptanalysis.

1 Introduction

In this paper we study quantum algorithms for classical nested search problems with a unique solution, where each of the ℓ search layers has known degrees of freedom as well as constraints as formalized in Algorithm 1. The degrees of freedom for each layer are captured by sets C_i of value choices for $i = 1, \ldots, \ell$. Constraints are captured by functions f_i that output a bit 0 ('invalid partial solution') or 1 ('valid partial solution'). Our formalization also considers minimizing cost and allows an auxiliary state to be passed to avoid redundant computations.

Moreover, we allow the filter functions f_i to compute only a partial auxiliary state to stop as early as possible, whereas a postprocessing function d_i is used to output the final auxiliary state of each layer. Every layer can access (but not modify) the choice and auxiliary state of prior layers. These arise very often

Algorithm 1 Classical nested search problem with backtracking

```
**Input:** NestedSearchProblem of $\ell$ layers defined by
  • Sets $C_i$ for $i \in \{1, \ldots, \ell\}$ representing possible values for $\ell$ variables
  • Filtering functions $f_i : C_1 \times \ldots \times C_i \times S_1 \times \ldots \times S_{i-1} \to \{0,1\} \times S'_i$
  • Postprocessing functions $d_i : C_1 \times \ldots \times C_i \times S_1 \times \ldots \times S_{i-1} \times S'_i \to S_i$
  • Here sets $S'_1, \ldots, S'_\ell$, $S_1, \ldots, S_\ell$ represent internal states of $f_i, d_i$.
  • Having a single solution SOL $= (c_1, \ldots, c_\ell, s_1, \ldots, s_\ell)$ such that:
      • $(1, s'_i) = f_i(c_1, \ldots, c_i, s_1, \ldots, s_{i-1})$ for $i = 1, \ldots, \ell$
      • $s_i = d_i(c_1, \ldots, c_i, s_1, \ldots, s_{i-1}, s'_i)$ for $i = 1, \ldots, \ell$
**Problem: output** SOL
```

in cryptanalytic attacks (e.g., differential [27], impossible differential [8], linear, zero-correlation, Square [16], DS-MITM [14] ...) that exploit structure to solve cryptographic problems significantly faster than a simple exhaustive search. Our constraint for a unique solution might seem arbitrary, but in many cases this constraint occurs naturally: e.g., in key or state recovery attacks. When there are multiple solutions and their expected number is known in advance, e.g., as in hash function collision attacks, it is possible to more strongly filter in the filtering functions f_i such that only a single solution is expected to remain.

For classical computers this nested search problem has a clear and optimal solution consisting of straightforward nested `for`-loops and `if`-statements. However, the optimal solution is not always so clear in the case of quantum computers. Classical nested searches can often be transformed into quantum algorithms using Grover's quantum search [19] or Quantum amplitude amplification (QAA) [9] obtaining up to a square-root speedup. However, contrary to classical exhaustive searches, Grover's algorithm and the QAA run for a fixed number of iterations, which determines their success probability. Finding the number of QAA iterations for each level, and optimizing the exact algorithm, involves subtle technicalities that are complex to handle:

Loss factor per level: Applying Grover or QAA naively results in a multiplicative loss factor of $\pi/2$ per level, totaling $(\pi/2)^\ell$. This can be reduced by a more refined analysis using 'undercooking', i.e., under-amplifying the 'good' amplitudes in intermediate levels and only amplifying to the desired success probability at the last level;

Cascading success probability errors: The success probabilities of each layer, which guide the number of iterations to perform, may be estimations given up to a certain variance. Further errors may arise at each level due to the rounding to an integer number of iterations. These errors cascade through the following levels and need to be properly estimated and controlled.

Uncertainly in success probabilities: When success probabilities for levels are unknown or their variance is too large, they can either be estimated via Quantum amplitude estimation [9], mitigated by performing the same search multiple times in parallel, or handled via "overcooking", i.e. intentionally using more, or a variable number of iterations to lead to a predictable success probability. All three approaches induce further technicalities in the complexity analysis.

Previous works dealing with quantum cryptanalysis often err on the safe side, by performing the analysis layer by layer, counting the $\frac{\pi}{2}$ factors and reducing the intermediate error probabilities to negligible amounts. This leads to an overestimate of these complexities, which may be problematic for attacks close to the bound of exhaustive search (e.g., in one of the attacks of [7], the difference with Grover search is only of a factor 2^2). This also leads to descriptions more focused on the quantum side, even though the classical nested search procedure is often much easier to explain than its quantum version.

1.1 Our Contributions

In this work, we presume the nested search problem functions f_i, d_i are transformed into corresponding quantum algorithms $\mathcal{A}_1, \ldots, \mathcal{A}_\ell$ and $\mathcal{D}_1, \ldots, \mathcal{D}_\ell$ that are deterministic and reversible and operate on a Hilbert space $\mathcal{H} = \mathcal{C}_1 \otimes \ldots \otimes \mathcal{C}_\ell \otimes \mathcal{W}_1 \otimes \ldots \mathcal{W}_\ell \otimes \mathcal{F}$, where

1. $\mathcal{C}_i$ are quantum registers that encode the choice $c_i \in C_i$;
2. $\mathcal{W}_i$ is a workspace register for search level i that includes an encoding of the auxiliary data $s_i \in S_i$;
3. $\mathcal{F}$ is a sequence of flag qubits, where the i-th qubit is initially 0 and modified only by the i-th level quantum algorithm $\mathcal{A}_i$ to 1 if the corresponding partial solution $(c_1, \ldots, c_i)$ is not filtered; there is only one full solution $(c_1, \ldots, c_\ell)$;
4. $\mathcal{A}_i$ implements f_i and operates on the registers $\mathcal{C}_1, \ldots, \mathcal{C}_i, \mathcal{W}_1, \ldots, \mathcal{W}_i, \mathcal{F}$ but only modifies $\mathcal{W}_i$ and the i-th qubit flag in $\mathcal{F}$;
5. $\mathcal{D}_i$ implements d_i and operates on the registers $\mathcal{C}_1, \ldots, \mathcal{C}_i, \mathcal{W}_1, \ldots, \mathcal{W}_i, \mathcal{F}$ but only modifies $\mathcal{W}_i$.

The quantum nested search problem is then informally rephrased as: given $\mathcal{A}_i$, $\mathcal{D}_i$ construct a quantum algorithm optimizing for low cost and high success probability, i.e., the probability to measure the unique solution SOL from the output quantum superposition.

We present two simple solutions and a corresponding quantum complexity analysis for two slightly different settings.

In Section 3, we first cover the special simpler case of a *search with early aborts*, which corresponds to a single choice c_1 (the remaining choices are trivial $C_2 = \ldots = C_\ell = \{0\}$) followed by a sequence of tests $f_1, \ldots, f_\ell$ of increasing complexities. We use the name *variable-time amplitude amplification* for this case, because it also corresponds to a quantum amplitude amplification of a *variable-time algorithm*. This problem was first studied by Ambainis [2], who showed that when doing a search for a good element among N of them, when being "good" is evaluated in time t_i for element number i, the solution can be found in time $\widetilde{\mathcal{O}}\left(\sqrt{t_1^2 + \ldots + t_N^2}\right)$. However, his solution relied on Quantum amplitude estimation. We show that amplitude amplification is enough to tackle this problem, and obtain the same asymptotic complexity with a simpler algorithm.

In Section 4 we cover the general case and present our main framework combining early-abort and backtracking, assuming a certain knowledge on the probabilities of success at each step. Our algorithm also relies on a recursive nesting of QAAs, though its structure is different than for the above case.

In both cases in Sections 3 and 4, we perform detailed computations of the probability of success of our algorithms, parameterized by the number of QAA iterates that we will choose to perform. We obtain generic formulas in Lemma 10 and Lemma 16. These imply that with ℓ layers of QAA, we essentially lose a factor $\mathcal{O}(\sqrt{\ell})$ instead of $\left(\frac{\pi}{2}\right)^\ell$ in the time complexity. Furthermore, we discuss optimizing the complexity numerically for concrete instances in Sections 3.5 and 4.4, which leads not only to a simplified analysis, but also to improvements on quantum attacks from the literature.

In Section 5 we take the key-recovery attacks on AES of [7] as an example to demonstrate our framework. We are able to further optimize quantum nested searches and gain up to a factor 2^4. Further applications are given in-Appendix. We show that quantum key search on *any reasonable block cipher* E_k can be performed by repeating $\frac{\pi}{4}2^{\kappa/2+0.01}$ times a circuit that tests whether $\mathrm{trunc}(E_k(p)) = \mathrm{trunc}(c)$ for a given plaintext-ciphertext pair, where trunc is a truncation to 20 bits. We study the case of a *search with independent tests*, where we are looking for an x that satisfies several independent one-bit testing functions $f_i(x)$, both asymptotically and with exact values. The code of our experiments is available at: `https://github.com/AndreSchrottenloher/quantum-search`.

Related work. Over the time, there has been a few attempts at formalizing the correspondence between classical and quantum nested search algorithms, by seeing them as *sampling* algorithms for their solution set (or *filtering* algorithms as in [7]). However these approaches were only intended to guide the design of such algorithms, and did not provide a generic complexity analysis of the results.

The analysis done by Ambainis in [2,3], for the special case of variable-time amplitude amplification, contains a reduction of the $\left(\frac{\pi}{2}\right)^\ell$ factor to $\mathcal{O}(\sqrt{\ell})$, but it was only used in the context of asymptotic complexities.

Non-asymptotic complexity analyses holding at a rather generic level were also performed, but were limited to a fixed number of nested searches, e.g. $\ell = 2$ for the *search with two oracles* in [13,21].

2 Preliminaries

In this work these upper and lower bounds for sin and arcsin are used:

$$\forall x \geq 0: \quad x\left(1 - x^2/6\right) \leq \sin x \leq x, \text{ and } x^2\left(1 - x^2/3\right) \leq \sin^2 x \leq x^2 \quad (1)$$

$$\forall 0 \leq x \leq 1: \quad x \leq \arcsin x \leq x\left(1 + (\pi/2 - 1)\, x^2\right) \quad (2)$$

$$x^2 \leq \arcsin^2 x \leq x^2\left(1 + \left(\pi^2/4 - 1\right) x^2\right) \leq x^2\pi^2/4 \quad (3)$$

These are standard bounds deduced from the Taylor series approximations, and the study of the function: $x \mapsto \arcsin x - x - \left(\frac{\pi}{2} - 1\right) x^3$.

Quantum Algorithms and Memory Models. We refer to [24] for an introduction to quantum computing. In this paper, we study quantum algorithms in the *quantum circuit model.* In this model, a quantum system consists of a certain number n of *qubits.* A qubit is a 2-dimensional elementary quantum system, with a state described by a vector of norm 1 in $\mathcal{H} = \mathbb{C}^2$, on the canonical basis $|0\rangle, |1\rangle$. The (normalized) state of an n-qubit system lies in the space $\mathcal{H}^{\otimes n} = \mathcal{H}^{2^n}$, with the canonical (*computational*) basis made of all n-bit strings. Measuring the state $|\psi\rangle = \sum_{i \in \{0,1\}^n} \alpha_i |i\rangle$ collapses it to $|i\rangle$ with probability $|\alpha_i|^2$.

A *quantum circuit* is a sequence of basic unitary operators of $\mathcal{H}^{\otimes n}$. These basic operators (*quantum gates*) usually apply to one or two selected qubits. A standard set of such gates is the Clifford+T set, which is often used for counting gates in quantum algorithms (see e.g. [20] or [6]). The *time complexity* is then the number of quantum gates in the circuit. We can either consider an *exact* time complexity, when the circuit is fixed, or an *average* complexity, when the circuit is drawn (before being run) from a family of possible circuits.

We use $G(\mathcal{A})$ to denote the gate count of a quantum circuit $\mathcal{A}$, and $S(\mathcal{A})$ the width of the circuit, i.e., the number of qubits on which it acts. Note that $S(\mathcal{A})$ does not take into account the ancilla qubits.

Any quantum algorithm $\mathcal{A}$ without measurement is reversible, and admits an inverse denoted $\mathcal{A}^\dagger$. We also often say that we "uncompute" $\mathcal{A}$. Uncomputing $\mathcal{A}$ means applying the sequence of gates of $\mathcal{A}$, inverted, and in reverse, so we consider $\mathcal{A}^\dagger$ and $\mathcal{A}$ to have the same time complexity.

In symmetric cryptanalysis, time complexity estimates are often expressed relatively to the cost of a cryptographic function. For example, the exhaustive key search of a 128-bit block cipher is estimated as 2^{128} *evaluations of the cipher.* This principle remains true in quantum cryptanalysis. We can consider the *evaluation of a quantum circuit for the cipher* to be the benchmarking operation, or alternatively, as in [7], we can single out some costly component of the cipher (here an S-Box) and consider only the number of evaluations of this sub-circuit.

Memory Models. Depending on the model considered, a quantum algorithm can access *classical* and *quantum memory.* The quantum memory requirement is the number of qubits used by the circuit, including qubits used for intermediate computations (*ancilla*) which are returned to the initial $|0\rangle$ state afterwards.

Given an array of M memory locations (qubit registers), any *fixed location* can be queried in polynomial time in the circuit model, as it amounts only to apply quantum gates to specific pairs of qubits: $|x\rangle |y_1, \dots, y_M\rangle \xmapsto{\mathsf{Access}_i} |x \oplus y_i\rangle |y_1, \dots, y_M\rangle$. However, accessing a *variable* memory location (*random access*) can a priori be done only by performing a sequence of $\widetilde{\mathcal{O}}(M)$ such queries, which gives a cost $\widetilde{\mathcal{O}}(M)$ for the following operation: $|x\rangle |y_1, \dots, y_M\rangle |i\rangle \xmapsto{\mathsf{Access}} |x \oplus y_i\rangle |y_1, \dots, y_M\rangle |i\rangle$. In the *QRAQM*[1] *model*, we assume that the operation $\xmapsto{\mathsf{Access}}$ can be performed in polynomial time. In practice, we will consider its cost to be comparable to a block cipher or an S-Box evaluation.

A quantum algorithm can also access *classical memory.* In that case, we assume that a large classical memory $y_1, \dots, y_M$ is stored outside the circuit. The *sequential access* in time $\widetilde{\mathcal{O}}(M)$ is still allowed by the quantum circuit model, since it amounts to control at runtime the sequence of applied gates: $|x\rangle |i\rangle \xmapsto{\mathsf{CAccess}(y_1,\dots,y_M)} |x \oplus y_i\rangle |i\rangle$. In the *QRACM*[2] *model*, we assume that the operation $\mathsf{CAccess}(y_1, \dots, y_M)$ can be performed in polynomial time. In practice, its cost is considered to be comparable to a block cipher or an S-Box evaluation.

In the following, we will consider separately the *classical* and *quantum* memory costs and specify if we use QRAQM / QRACM (otherwise we are in the "plain" quantum circuit model).

Probability of Success. Although there exists exact variants of amplitude amplification (see [9]), we will not use them in this paper. Consequently, all our results will be statements of the form:

> there exists a quantum algorithm of (exact or average) time complexity T, (classical and / or quantum) memory complexity M, succeeding with probability p

with p smaller than one, typically $p \geq 0.5$.

Amplitude Amplification. Quantum amplitude amplification [9], abbreviated QAA in this paper, is a generalization of Grover search which allows to increase the success probability of any measurement-free quantum algorithm by iterating it. Let $\mathcal{A}$ be a quantum circuit such that

$$\mathcal{A}|0\rangle = \left(\textstyle\sum_{x \in G} \alpha_x |x\rangle\right)|0\rangle + \left(\textstyle\sum_{x \in B} \beta_x |x\rangle\right)|1\rangle = \sqrt{p}\,|\psi_G\rangle + \sqrt{1-p}\,|\psi_B\rangle\,, \quad (4)$$

where p is the success probability of $\mathcal{A}$ (real and positive); $|\psi_G\rangle$ is a superposition (not necessarily uniform) of good outcomes (the set G) and $|\psi_B\rangle$ of bad outcomes (the set B), marked by their respective flags 1 and 0. Let O_0 be the *inversion around zero* operator, which flips the phase of the basis vector $|0\rangle$: it does $O_0 |y\rangle = (-1)^{y==0} |y\rangle$; and O be the operator which flips the phase of all basis vectors $|x, b\rangle$ such that $b = 1$. The QAA computes a sequence of states $|\psi_i\rangle, 0 \leq i \leq m$, defined by the following iterative process:

[1] *Quantum random-access quantum memory*, following the terminology from [22].
[2] *Quantum random-access classical memory.*

```
1. Start from $|\psi_0\rangle = \mathcal{A}|0\rangle$
2. For $i = 1$ to $m$:
3.     $|\psi_{i+1}\rangle = \mathcal{A}O_0\mathcal{A}^\dagger O|\psi_i\rangle$
```

Lemma 1 below describes the increase of the amplitude of good outcomes. This implies that after $t = \left\lfloor \frac{\pi}{4\arcsin\sqrt{p}} \right\rfloor$ iterations, the value $(2t+1)\theta$ approaches $\frac{\pi}{2}$, and the success probability is at least $1-p$ [9].

Lemma 1 (From [9]). *Let $\theta = \arcsin(\sqrt{p})$, $|\psi_G\rangle$ and $|\psi_B\rangle$ be such that $|\psi_0\rangle = \sin(\theta)|\psi_G\rangle + \cos(\theta)|\psi_B\rangle$. Then: $|\psi_i\rangle = \sin((2i+1)\theta)|\psi_G\rangle + \cos((2i+1)\theta)|\psi_B\rangle$.*

In quantum search, and in the algorithms presented in this paper, the only computational overhead with respect to the iterations of $\mathcal{A}$ comes from the implementation of O_0 and O. We implement O_0 naively using an n-bit Toffoli gate (see [24], Exercises 4.27–4.30; a better count for O_0 is also given in [18]).

Lemma 2. *The inversion around zero on n qubits can be implemented using $n-1$ ancilla qubits and $44n-39$ Clifford+T gates. The O operator can be implemented using 1 ancilla qubit and 3 Clifford gates.*

Nesting Many QAAs. Since each iteration of QAA contains two calls to $\mathcal{A}$ (one reversed), a QAA needs approximately $\frac{\pi}{2}\frac{1}{\sqrt{p}}$ iterates instead of $\frac{1}{p}$ for a classical search. This multiplicative factor becomes problematic when we want multiple nested QAAs (the algorithm $\mathcal{A}$ contains a QAA, which contains a QAA, and so on), since we will pay a factor $\left(\frac{\pi}{2}\right)^\ell$ with ℓ levels of nesting.

Recall that $\sin x \simeq x$ when x is small, so the probability of success of the QAA initially grows almost as $(2i+1)^2p$: it increases quadratically. The $\frac{\pi}{2}$ factor only steps in if we want to increase it all the way close to 1. Thus, and perhaps counter-intuitively, to avoid piling up these factors we must *under-amplify the QAAs*, and keep their success probability artificially low. This fact is well-known (see Lemma 9 in [1]) and it will appear naturally in our computations.

Unknown Success Probability and "Overcooking". In this paper, we will often encounter the situation where we have only a lower bound $p_{\min}$ on p, and we want to find a solution with constant success probability. This will take us an expected time $\mathcal{O}(1/\sqrt{p_{\min}})$, by Theorem 3 in [9]. We settle for a simple method which consists in running a QAA with a varying number of iterates, either in a *controlled*, or *random* way.

Lemma 3. *Let $\mathcal{A}$ be a quantum circuit defined as in Equation 4. Assume that $p \geq p_{\min}$ and let $M = \lceil 1.21/\sqrt{p_{\min}} \rceil$. Then there exists a quantum procedure that produces $|\psi_G\rangle$ with probability at least $\frac{1}{2}$, and performs $4(M-1)$* controlled *iterations of QAA and a total of $4M$ applications of $\mathcal{A}$.*

Proof. The quantum algorithm does the following (twice):

1. Execute $\mathcal{A}$ and measure its output flag. If it is "1" then return $|\psi_G\rangle$.

2. Execute a controlled QAA with $i \leq M-1$ iterates, where $M = \lceil 1.21/\sqrt{\varepsilon_{\min}} \rceil$ (and each iteration invokes $\mathcal{A}$ and $\mathcal{A}^\dagger$ once). Measure its output flag and the control register. If a "1" flag is measured then return $|\psi_G\rangle$.

Here, each of the two controlled QAA consists in constructing a uniform superposition over a control register i, then applying i QAA iterates (i.e., actually a circuit of $M-1$ iterates controlled on the value of i). In total four attempts are made: 2 controlled QAA executions, and 2 single calls to $\mathcal{A}$ in case the success probability is already very high. For each of these four executions, if a "1" flag is measured, it means that the state has collapsed on $|\psi_G\rangle$, and we return it. Otherwise we only obtain $|\psi_B\rangle$ and we fail. Note that it contains measurements and thus is not reversible.

If we wanted to do this without measurement, then we would have to return a superposition of the four possible outcomes (and controls): $|i_1\rangle |i_2\rangle \otimes_{j=1}^4 |\psi_j\rangle$ where $|\psi_j\rangle \in \{|\psi_G\rangle, |\psi_B\rangle\}$. The same analysis guarantees that the amplitude on $|\psi_B\rangle^{\otimes 4}$ in this superposition is lower than $\frac{1}{\sqrt{2}}$.

Analysis of the controlled QAA. The QAA success probability is given by:

$$\begin{aligned}
&\frac{1}{M}\sum_{i=0}^{M-1} \sin^2\left((2i+1)\theta\right) \\
&= \frac{1}{2} - \frac{1}{4M}\sum_{i=0}^{M-1}\left(e^{\iota(4i+2)\theta} + e^{-\iota(4i+2)\theta}\right) \text{ using } \sin^2 x = \frac{1-\cos(2x)}{2} \\
&= \frac{1}{2} - \frac{1}{4M}\left(e^{\iota 2\theta}\frac{1-e^{\iota 4M\theta}}{1-e^{\iota 4\theta}} + e^{-\iota 2\theta}\frac{1-e^{-\iota 4M\theta}}{1-e^{-\iota 4\theta}}\right) \\
&= \frac{1}{2} - \frac{1}{4M}\left(e^{\iota 2M\theta}\frac{\sin(2M\theta)}{\sin(2\theta)} + e^{-\iota 2M\theta}\frac{\sin(2M\theta)}{\sin(2\theta)}\right) \\
&= \frac{1}{2} - \frac{1}{2M}\frac{\sin(2M\theta)\cos(2M\theta)}{\sin(2\theta)} \geq \frac{1}{2} - \frac{1}{2M\sin(2\theta)} \\
&\geq \frac{1}{2} - \frac{1}{4M\sin\theta\cos\theta} = \frac{1}{2} - \frac{1}{4M\sqrt{p(1-p)}} \quad .
\end{aligned}$$

If we combine this with a single call to $\mathcal{A}$, which has a probability of failure equal to $1-p$, then the probability of failure of the whole operation is bounded by:

$$(1-p)\left(\frac{1}{2} + \frac{1}{4M\sqrt{p(1-p)}}\right) \leq \frac{1}{2} + \frac{1}{4M\sqrt{p_{\min}}} \quad .$$

A single such procedure is not enough to reach a success probability of at least $1/2$. So we combine the results of two independent procedures. By choosing $M = \lceil 1.21/\sqrt{p_{\min}} \rceil$, we have: $M \geq 1.21/\sqrt{p_{\min}}$ which upper bounds the failure probability as: $\left(1/2 + 1/(4 \cdot 1.21)\right)^2 \leq 1/2$, concluding the proof. $\square$

Actually, by running a QAA with a *random* number of iterates, rather than a uniformly superposed number of iterates, we can obtain the same success probability, but now we half the average time complexity.

Lemma 4. *Let $M = \lceil 1.21/\sqrt{p_{\min}} \rceil$ and assumes that $\mathcal{A}$ acts on n qubits with a ancillas. There exists a quantum procedure that makes* on average $2M+2$ *calls to $\mathcal{A}$ and $\mathcal{A}^\dagger$ (uncontrolled), uses* $\max(a, n-1)$ *ancillas and* $(M-1) \times (44n-36)$ *Clifford+T gates, and returns a good result with probability $\frac{1}{2}$.*

Proof. We run a procedure similar to the above (see Lemma 3), but which selects a uniformly random $i \in [0; M-1]$ and runs the QAA with i iterates, rather than doing this for all possible i in superposition. The success probability has the exact same expression. The average number of calls to $\mathcal{A}$ is:

$$2\sum_{i=0}^{M-1}(2+2\cdot i)/M = 4 + 4\cdot({}^{(M-1)M}/{}_2)/M = 4 + 2(M-1) = 2M+2 \ .$$

And the average number of additional Clifford+T gates is: $(M-1) \times (44n-39) + (M-1) \times 3$. This finishes the proof. □

3 Amplitude Amplification with Variable Times

This section first considers a simpler special case of the quantum algorithms specified in Section 1.1 that corresponds to the setting of *variable-time amplitude amplification*, in which we try to amplify the success probability of an algorithm which can stop at different times. Our core idea for this setting is to amplify each level as much as possible, while avoiding any over-amplification. Previous works used Amplitude Estimation for this [3,10]. However, we show that a simple sequence of nested QAAs never over-amplifies, if we know an upper bound on the *cumulative probability of success* at each level. By taking a guess of the average time complexity (in L_2) of the variable-time procedure, such upper bounds follow using Markov's inequality. While this represents essentially the worst case, it is enough to reach the same asymptotic complexity as Ambainis in [3], with a much simpler procedure.

In Section 3.5, we give a tighter way to estimate the complexity and success probability of our method, which is more suited for cryptographic applications where the behavior of the amplified algorithm is completely known.

3.1 Setting and Definitions

This setting corresponds to a special case of the quantum algorithms specified in Section 1.1 where $C_2 = \ldots, C_\ell = \{0\}$ are trivial and only $\mathcal{C}_1$ is non-trivial, and thus all $\mathcal{D}_i$ can be expressed as part of $\mathcal{A}_i$. To simplify notation in this section, we instead use a definition of variable-stopping-time algorithms close to the ones in [3,10]. Let $\ell \geq 2$ be an integer ($\ell = 1$ would be covered by a simple QAA) and $t_1 < \ldots < t_\ell$ be a sequence of time stamps. A *variable-stopping-time* algorithm $\mathcal{A}$ is given by: $\mathcal{A} = \mathcal{A}_\ell \cdots \mathcal{A}_1$ where the gate count of $\mathcal{A}_i \cdots \mathcal{A}_1$ is t_i, i.e., we set $t_0 = 0$ and each $\mathcal{A}_i$ has gate count $G(\mathcal{A}_i) = t_i - t_{i-1}$. All the $\mathcal{A}_i$ act on a space of the form $\mathcal{F} \otimes \mathcal{W}$ where:

- the first register is a sequence of ℓ qubit *flags* which are initialized to the value 0, and the i-th flag is flipped to 1 by $\mathcal{A}_i$ if the corresponding value passes the i-th check;
- the second register is the "work" register of the $\mathcal{A}_i$.

Intuitively, each step $\mathcal{A}_i$ writes the flag number i. A "0" flag indicates that we must stop at this point, whereas a "1" flag means a potential solution and we can continue on this path. This models an *early abort* technique, where the $\mathcal{A}_i$ form a sequence of tests of increasing complexities. We discard early on some chunks of the search space with inexpensive tests, then move on to more expensive ones. Only the last flag, written by $\mathcal{A}_\ell$, determines if we have actually reached the solution. In some applications, we may be able to detect solutions in earlier steps; in that case, to fit in our framework, we still continue the computation until $\mathcal{A}_\ell$ before "officially declaring" that we have the solution. If the algorithm is correctly parameterized, this shouldn't change significantly the average time complexity.

In the following we use 1_i to denote the bit-string $1 \cdots 1$ with i bits, 0_i similarly, and $*_i$ to denote any bit-string of length i. We define amplitudes α_i, β_i and vectors $|\psi_i\rangle, |\psi_i^\perp\rangle$ such that:

$$\begin{aligned} |\psi_0\rangle &= |0\rangle \\ \forall i \geq 1, |1_{i-1}0_{\ell-i+1}\rangle |\psi_{i-1}\rangle &\xmapsto{\mathcal{A}_i} \alpha_i |1_i 0_{\ell-i}\rangle |\psi_i\rangle + \beta_i |1_{i-1}0_{\ell-i+1}\rangle |\phi_i\rangle \\ |1_{\ell-1}0\rangle |\psi_{\ell-1}\rangle &\xmapsto{\mathcal{A}_\ell} \alpha_\ell |1_\ell\rangle |\psi_\ell\rangle + \beta_\ell |1_{\ell-1}0\rangle |\phi_\ell\rangle \end{aligned}$$

Each $\mathcal{A}_i$ acts on the work register, controlled on the $i-1$-th flag being 1, and it updates the i-th flag and the current work register (except $\mathcal{A}_1$, the first step, which does not need to be controlled). Otherwise it does nothing. At the end of the complete computation, i.e., after applying the complete $\mathcal{A}$, the state is a superposition of bad outcomes obtained at each level, and of good outcomes of the final level. Finally, we assume that the α_i, β_i are real, positive numbers.

Recursive QAAs. For a sequence of integers $k_1, \ldots, k_\ell$ that we choose afterwards, we define recursively ℓ algorithms $\mathcal{B}_i(k_1, \ldots, k_i)$. Like the $\mathcal{A}_i$, the $\mathcal{B}_i$ work in the space $\mathcal{F} \otimes \mathcal{W}$. Similarly as what we do classically, we use amplitude amplification to rule out failure cases before we move on to the next stage. Our ultimate goal is that, when we measure the output of $\mathcal{B}_\ell$, we project the first register on $|1\rangle$ with high probability, i.e., a good result. We denote by O_i the test oracle that simply flips the phase if the i-th flag is 1 (not finished, or finished and good). Each O_i costs 3 Clifford gates.

$$O_i \; : |*_{i-1} b *_{\ell-i}\rangle |x\rangle \mapsto (-1)^b |*_{i-1} b *_{\ell-i}\rangle |x\rangle \quad .$$

- $\mathcal{B}_1(k_1)$ does k_1 iterates of QAA on $\mathcal{A}_1$, using the test oracle O_1
- for all i, $\mathcal{B}_i(k_1, k_2, \ldots, k_i)$ runs k_i iterates of QAA on $\mathcal{A}_i \mathcal{B}_{i-1}(k_1, k_2, \ldots, k_{i-1})$ (this is indeed an algorithm that takes no input and makes no measurements), using the test oracle O_i

- $\mathcal{B}_\ell(k_1, \dots, k_\ell)$ is our complete algorithm.

For each i, let ν_i^2 be the success probability of $\mathcal{B}_i$, i.e., the probability that it projects the flag number i on $|1\rangle$. We express the ν_i using the framework of QAA, in terms of the α_i, using the lemma below. Now, it remains to choose the parameters k_i appropriately, to ensure a good lower bound on the success probability ν_1^2.

Lemma 5. *For $i = 1$ we have: $\nu_1 = \sin\left[(2k_1 + 1)\arcsin \alpha_1\right]$. For all $i \geq 2$ we have:*

$$\nu_i = \sin\left[(2k_i + 1)\arcsin \alpha_i \nu_{i-1}\right] \quad . \tag{5}$$

Proof. The algorithm $\mathcal{A}_1$ outputs: $\mathcal{A}_1 |0_\ell\rangle = \alpha_1 |10_{\ell-1}\rangle |\psi_1\rangle + \beta_1 |0_\ell\rangle |\phi_1\rangle$, so the output of $\mathcal{B}_1(k_1)$ can then be written as:

$$\mathcal{B}_1(k_1)\, |0_\ell\rangle = \nu_1 |10_{\ell-1}\rangle |\psi_1\rangle + \sqrt{1-\nu_1^2}\, |0_\ell\rangle |\phi_1\rangle \quad ,$$

where ν_1 has the given expression by the properties of QAA (Lemma 1). Now, assume that a given $\mathcal{B}_i$ outputs a superposition of the form:

$$\mathcal{B}_i\, |0_\ell\rangle = \nu_i |1_i 0_{\ell-i}\rangle |\psi_i\rangle + \sqrt{1-\nu_i^2}\, |\chi_i\rangle \tag{6}$$

where $|\chi_i\rangle$ is a superposition of failed states from levels $1, \dots, i$, with different flags of the form $|1_j 0_{\ell-j}\rangle$ depending on the level which failed; and ν_i has the given formula. We show that this is the same for $\mathcal{B}_{i+1}$. Indeed, by definition, $\mathcal{B}_{i+1}(k_{i+1}, \dots, k_1)$ applies k_{i+1} QAA iterates to the algorithm $\mathcal{A}_{i+1}\mathcal{B}_i$. We first look at the output of $\mathcal{A}_{i+1}\mathcal{B}_i$:

$$\begin{aligned}
\mathcal{A}_{i+1}\mathcal{B}_i\, |0_\ell\rangle &= \mathcal{A}_{i+1}\left(\nu_i |1_i 0_{\ell-i}\rangle |\psi_i\rangle + \sqrt{1-\nu_i^2}\, |\chi_i\rangle\right) \\
&= \nu_i \left(\alpha_{i+1} |1_{i+1} 0_{\ell-i-1}\rangle |\psi_{i+1}\rangle + \beta_{i+1} |1_i\rangle |0_{\ell-i}\rangle |\phi_{i+1}\rangle\right) + \sqrt{1-\nu_i^2}\, |\chi_i\rangle \\
&= \nu_i \alpha_{i+1} |1_{i+1} 0_{\ell-i-1}\rangle |\psi_{i+1}\rangle + \sqrt{1-(\nu_i\alpha_{i+1})^2}\, |\chi_{i+1}\rangle
\end{aligned}$$

since $|\chi_i\rangle$ has always 0 in the flag bit i, so $\mathcal{A}_{i+1}$ leaves it unchanged, and we define $|\chi_{i+1}\rangle$ accordingly. By Lemma 1 again, we have that:

$$\mathcal{B}_{i+1}\, |0\rangle = \nu_{i+1} |1_{i+1} 0_{\ell-i-1}\rangle |\psi_{i+1}\rangle + \sqrt{1-\nu_{i+1}^2}\, |\chi_{i+1}\rangle \tag{7}$$

where ν_{i+1} has the given definition. □

3.2 Analysis of the Algorithm

We now analyze the success probability of $\mathcal{B}_\ell$ very precisely in terms of k_i and α_i.

Lemma 6. *Let $\nu_0 = 1$, and $s_i = (2k_i + 1)^2 \alpha_i^2$. Let $d = \left(\frac{\pi^2}{4} - 1\right)$. Then for all $i \geq 1$ we have:*

$$s_i \nu_{i-1}^2 \left(1 - s_i \nu_{i-1}^2\right) \leq \nu_i^2 \leq s_i \nu_{i-1}^2 \left(1 + d s_i \nu_{i-1}^2\right) \quad . \tag{8}$$

Proof. Recall the recursive formula of Lemma 5: $\nu_i = \sin\left[(2k_i+1)\arcsin(\alpha_i\nu_{i-1})\right]$, with the case $i=1$ being the particular case of the first QAA. First we use the bounds on sin, then on arcsin. In the lower bound we have:

$$\begin{aligned}\nu_i^2 &\geq (2k_i+1)^2\arcsin^2(\alpha_i\nu_{i-1})\left[1-\frac{(2k_i+1)^2\arcsin^2(\alpha_i\nu_{i-1})}{3}\right]\\ &\geq (2k_i+1)^2\left(\alpha_i\nu_{i-1}\right)^2\left[1-\frac{1}{3}\frac{\pi^2}{4}(2k_i+1)^2(\alpha_i\nu_{i-1})^2\right]\quad,\end{aligned}$$

and the bound follows from $1/3\cdot\pi^2/4\leq 1$. In the upper bound we have:

$$\nu_i^2\leq(2k_i+1)^2\arcsin^2(\alpha_i\nu_{i-1})\leq(2k_i+1)^2(\alpha_i\nu_{i-1})^2\left(1+\left(\pi^2/4-1\right)(\alpha_i\nu_{i-1})^2\right)$$

and the bound follows from $d:=(\pi^2/4-1)\leq d(2k_i+1)^2$. □

Remark 1. Equation 8 remains valid even if some of the k_i are zero, i.e., those layers do not actually use a QAA. However the quality of the bounds is directly related to ℓ, so it is always better to use these formulas with all $k_i>0$.

From there, we bound the ν_i^2 from above to first ensure that they are always smaller than 1. Our intuition is that when we expand recursively the upper bound $\nu_i^2\leq s_i\nu_{i-1}^2\left(1+ds_i\nu_{i-1}^2\right)$, we obtain a sum of terms that contain products of the s_i. We do not need to upper bound the s_i individually: we only need to upper bound these products.

Lemma 7. *For any $1\leq i\leq\ell-1$, let $S_i=\prod_{j=1}^i s_i=\prod_{j=1}^i(2k_j+1)^2\alpha_j^2$. Let $d=\frac{\pi^2}{4}-1$. Assume that all S_i are bounded above by $\frac{c}{\ell}$ for some constant c. Then for all i, ν_i^2 is upper bounded by:*

$$\nu_i^2\leq S_i\sum_{j=0}^{2^i-1}i^j(dc/\ell)^j \tag{9}$$

In particular, if we assume $c\leq 4/\pi^2$, then for all $1\leq i\leq\ell-1$ we have: $\nu_i^2\leq 1/\ell$.

Proof. We prove the first statement by recurrence over i. For $i=1$ we have immediately by definition: $\nu_1^2\leq s_1(1+ds_1)\leq S_1(1+dc)$. Now, assume that Equation 9 holds for some i. We have:

$$\begin{aligned}\nu_{i+1}^2\leq s_{i+1}\nu_i^2(1+ds_{i+1}\nu_i^2)&\leq S_{i+1}\left(\sum_{j=0}^{2^i-1}i^j(dc/\ell)^j\right)\left(1+\sum_{j=0}^{2^i-1}i^j(dc/\ell)^{j+1}\right)\\ &\leq S_{i+1}\left(\sum_{j=0}^{2^i-1}i^j(dc/\ell)^j+\left(\sum_{j=0}^{2^i-1}i^j(dc/\ell)^j\right)\left(\sum_{j=1}^{2^i}i^{j-1}(dc/\ell)^j\right)\right)\\ &\leq S_{i+1}\left(\sum_{j=0}^{2^i-1}i^j(dc/\ell)^j+\sum_{j=1}^{2^{i+1}-1}\sum_{k=1}^{j}i^{j-k}\times i^{k-1}(dc/\ell)^j\right)\end{aligned}$$

$$\leq S_{i+1} \sum_{j=0}^{2^{i+1}-1} (j i^{j-1} + i^j)(dc/\ell)^j \leq t'_{i+1} \sum_{j=0}^{2^{i+1}-1} (i+1)^j (dc/\ell)^j \ ,$$

which finishes the recurrence. Next, if $c \leq 4/\pi^2$ then for all i (including $i = 1$) we can deduce $dci/\ell \leq dc \leq 1 - 4/\pi^2 := d'$. This simplifies the bound over ν_i^2:

$$\nu_i^2 \leq S_i \left(\sum_{j=0}^{2^i-1} d'^j \right) \leq S_i \frac{1}{1-d'} = S_i \times \frac{\pi^2}{4} \ ,$$

and we use $S_i \leq c/\ell$ (by assumption) to conclude: $\nu_i^2 \leq \pi^2/4 \times c/\ell = 1/\ell$. □

If one can bound the *cumulative* probabilities of success ($\prod_{j=1}^{i} \alpha_j^2$), instead of individual ones, then Lemma 7 implies we can maintain an upper bound on the ν_i^2 which is inverse-linear in ℓ. In turn, this allows to lower bound ν_ℓ^2.

Lemma 8. *Assume that* $\forall i \leq \ell, S_i = \prod_{j=1}^{i}(2k_j+1)\alpha_j^2 \leq \frac{c}{\ell}$ *with* $c \leq \frac{4}{\pi^2}$. *Then we have:*

$$\nu_\ell^2 \geq e^{-1} \cdot S_\ell \ . \tag{10}$$

Proof. From Lemma 7 we can deduce that for all $i \leq \ell$:

$$s_i \nu_{i-1}^2 \leq s_i S_{i-1} \left(\sum_{j=0}^{2^{i-1}-1} i^j (dc/\ell)^j \right) \leq \frac{\pi^2}{4} S_i \ .$$

Thus, for all $\ell \geq i \geq 2$, by $\frac{\pi^2}{4} S_i \leq \frac{1}{\ell}$, we have: $\nu_i^2 \geq s_i \nu_{i-1}^2 (1 - s_i \nu_{i-1}^2) \geq s_i \nu_{i-1}^2 (1 - 1/\ell)$. And $\nu_1^2 \geq s_1$ by definition. Unfolding these inequations gives:

$$\nu_\ell^2 \geq \left(\prod_{i=1}^{\ell} s_i \right) \left(1 - \frac{1}{\ell} \right)^{\ell-1} \geq \frac{1}{e} S_\ell \ .$$ □

Lemma 8 can be interpreted as follows. The value $\prod_i \alpha_i^2$ is the probability of success that we would get if we simply ran the composition of the algorithms $\mathcal{A}_i$, without layers of QAA. When we define the algorithms $\mathcal{B}_i$, under the condition that we do not overestimate the total probability of success of each layer, we can amplify up to an inverse-linear (in ℓ) probability of success.

Time Complexity. The time complexity of $\mathcal{B}_\ell$ is a function of the k_i and the gate counts of $\mathcal{A}_i$. Recall that we note $G(\mathcal{A}_i) = t_i - t_{i-1}$ the gate count of $\mathcal{A}_i$, that includes the controls on the flag qubit number $i-1$. Let w be the number of qubits in the workspace of $\mathcal{A}$ (note that the $\mathcal{A}_i$ could use additional ancilla qubits between two successive steps).

Lemma 9. *The gate complexity of* $\mathcal{B}_\ell(k_1, \ldots, k_\ell)$ *is given by:*

$$G(\mathcal{B}_\ell) = \left(\sum_{i=1}^{\ell} \left(\prod_{j=i}^{\ell} (2k_j+1) \right) G(\mathcal{A}_i) \right) + (44(w+\ell) - 36) \prod_{j=1}^{\ell} (2k_j+1) \ . \tag{11}$$

It uses $w + \ell$ *qubits and at least* $w + \ell - 1$ *ancillas.*

Proof. This is a simple induction on the number of applications of each $\mathcal{A}_i$. For all i, $\mathcal{B}_\ell$ contains $\prod_{j=i}^{\ell}(2k_j+1)$ calls to $\mathcal{A}_i$. Each time O_0 is called, it is applied on $w+\ell$ qubits (the workspace and flags), so it contains less than $44(w+\ell)-39$ Clifford+T gates. Also each O_i contains 3 Clifford gates. Finally, the total number of calls to O_0 (resp. the O_i) is $\prod_{j=1}^{\ell}(2k_j+1)$. □

Example 1. If we take two levels of QAA and one iterate at each level, we obtain at the first level:

$$\mathcal{B}_1 = \mathcal{A}_1 O_0 \mathcal{A}_1^\dagger O_1 \mathcal{A}_1$$

and at the second level:

$$\begin{aligned}\mathcal{B}_2 &= \mathcal{A}_2\mathcal{B}_1 O_0 (\mathcal{A}_2\mathcal{B}_1)^\dagger O_2 \mathcal{A}_2\mathcal{B}_1 \\ &= \mathcal{A}_2\mathcal{A}_1 O_0\mathcal{A}_1^\dagger O_1\mathcal{A}_1 O_0(\mathcal{A}_2\mathcal{A}_1 O_0\mathcal{A}_1^\dagger O_1\mathcal{A}_1)^\dagger O_2\mathcal{A}_2\mathcal{A}_1 O_0\mathcal{A}_1^\dagger O_1\mathcal{A}_1 \\ &= \mathcal{A}_2\mathcal{A}_1 O_0\mathcal{A}_1^\dagger O_1\mathcal{A}_1 O_0\mathcal{A}_1^\dagger O_1\mathcal{A}_1 O_0\mathcal{A}_1^\dagger\mathcal{A}_2^\dagger O_2\mathcal{A}_2\mathcal{A}_1 O_0\mathcal{A}_1^\dagger O_1\mathcal{A}_1 \ .\end{aligned}$$

So the algorithm looks like a QAA in which we stop earlier the amplified algorithm at some iterations.

3.3 Choosing the Number of Iterates

Notions of average time. For our applications, we need proper definitions of the *average time complexity* of $\mathcal{A}$. We consider its L_2 average T_2 and its L_1 average T_1. As stated above, we make the simplifying assumption that "good outcomes" are decided at the last time step. In the end we are simply approximating the "good" state $|\psi_\ell\rangle$.

If we let run $\mathcal{A}$ on input $|0\rangle$, without any QAA iterates, we obtain the state:

$$\begin{aligned}\mathcal{A}|0\rangle = \beta_1|0_\ell\rangle|\phi_1\rangle + \beta_2|10_{\ell-1}\rangle|\phi_2\rangle + \dots \\ + \beta_\ell\prod_{i=1}^{\ell-1}\alpha_i|1_{\ell-1}0\rangle|\phi_\ell\rangle + \prod_{i=1}^{\ell}\alpha_i|1_\ell\rangle|\psi_\ell\rangle \ . \end{aligned} \quad (12)$$

We define then the quantities T_2 and T_1 are then defined as the average stopping time, i.e., the average time to decide if a computation path leads to a good result or not. If we define: $\forall i, p_i := \beta_i^2\prod_{j=1}^{i-1}\alpha_j^2$ and $p_\ell := \prod_{i=1}^{\ell-1}\alpha_i^2$, then:

$$T_2 = \sqrt{\sum_i p_i t_i^2} \text{ and } T_1 = \sum_i p_i t_i \ . \quad (13)$$

Here p_i is the probability that the computation takes exactly time t_i. By Jenssen's inequality, we have that: $T_1 \leq T_2$. We can also write T_1 as:

$$T_1 = t_1 + \sum_{i=2}^{\ell}\left(\prod_{j=1}^{i-1}\alpha_j^2\right)(t_i - t_{i-1}) \quad (14)$$

From the formula of T_2, because of the monotonicity of the t_i^2, we have that:

$$\sum_{i=1}^{\ell} p_i t_i^2 = T_2^2 \implies \forall i \geq 1, \left(\sum_{j=i}^{\ell} p_i\right) t_i^2 \leq T_2^2 \implies \prod_{j=1}^{i-1} \alpha_j^2 \leq T_2^2/t_i^2 \ . \qquad (15)$$

Indeed, $\left(\sum_{j=i}^{\ell} p_i\right)$ is the probability that the evaluation time is equal to or larger than t_i, so it is equal to $\prod_{j=1}^{i-1} \alpha_j^2$ (and 1 if $i = 1$, since every element needs at least the first evaluation step). Similarly we have $\prod_{j=1}^{i-1} \alpha_j^2 \leq T_1/t_i$.

Setting the number of iterates. Let us assume that the total success probability $p = \prod_{j=1}^{\ell} \alpha_i^2$ is known. Then using the inequalities on the products of α_j^2, we can set the number of iterates as follows.

Lemma 10. *Assume that $k_1, \ldots, k_\ell$ are such that:*

$$\begin{cases} \forall i \leq \ell - 1, \prod_{j=1}^{i} (2k_j + 1)^2 \leq \max\left(\frac{4}{\pi^2 \ell} \frac{t_{i+1}^2}{T_2^2}, 1\right) \\ k_\ell = 0 \end{cases}$$

then the layered QAA with iterates $k_1, \ldots, k_\ell$ has success probability lower bounded by: $\frac{p}{2e} \prod_{j=1}^{\ell-1} (2k_j + 1)^2$.

Proof. The result comes from Lemma 8. We just need to check that the conditions of the lemma are satisfied. For all $i \leq \ell - 1$ we have by Equation 15:

$$S_i := \prod_{j=1}^{i} (2k_j + 1)^2 \alpha_j^2 \leq \prod_{j=1}^{i} \alpha_j^2 \frac{4}{\pi^2 \ell} \frac{t_{i+1}^2}{T_2^2} \leq \frac{4}{\pi^2 \ell} \ .$$

And $S_\ell := (2k_\ell + 1)^2 \alpha_\ell^2 S_{\ell-1} \leq S_{\ell-1} \leq \frac{4}{\pi^2 \ell}$ for the final case. □

At this point, one can set the number of iterates only depending on our knowledge of T_2, p and the individual times t_i, *regardless* of the exact distribution of evaluation times. Besides, the knowledge of p is required only if one wants to amplify correctly the final step, otherwise T_2 and the t_i are sufficient information.

3.4 Variable-time Amplitude Amplification without Amplitude Estimation

The number of QAA layers that we use should, in general, depend on the structure of the problem. However, in most cases, we can choose to stop the algorithm $\mathcal{A}$ at any point. Let us assume that t_ℓ (the maximal running time of $\mathcal{A}$), T_2 and p are known. Here T_2 is the *actual* average time complexity for $\mathcal{A}$. Also, we introduce u such that $3^{\ell+u-1} T_2 < t_\ell \leq 3^{\ell+u} T_2$.

We set the iteration numbers k_i so that: $k_1 = \ldots = k_{\ell-1} = 1$ and $k_\ell = 0$. To do this, we introduce new time stamps: $t'_i = 3^{i+u} T_2$ for all $i < \ell$. We consider another variable-time algorithm $\mathcal{A}'$ that stops only *at these time stamps*. In particular, $\mathcal{A}'$ has a bigger average time complexity, since the number of stopping steps has been reduced: $T_2' \geq T_2$. However our choice of steps also ensures that $T_2' \leq 3T_2$. We then try to satisfy the conditions of Lemma 10 for $\mathcal{A}'$, by choosing appropriate values for u and ℓ. The first constraint that we put on u is: $u \geq 1 + \left\lceil \log_3 \left(\frac{\pi}{6} \sqrt{\ell} \right) \right\rceil$. Indeed, this allows to satisfy the conditions of the lemma for $i \leq \ell - 1$, as we have:

$$\frac{4}{\pi^2 \ell} \frac{t_{i+1}'^2}{T_2'^2} \geq \frac{4}{9\pi^2 \ell} \frac{t_{i+1}'^2}{T_2^2} \geq \frac{4}{9\pi^2 \ell} 3^{2i+2u+2} = 3^{2i} \times \frac{4}{9\pi^2 \ell} \times 81 \times \left(\frac{\pi}{6} \sqrt{\ell} \right)^2 \geq 3^{2i} \ .$$

We also have the constraint $\ell + u = \left\lceil \log_3 \frac{t_\ell}{T_2} \right\rceil$ by definition. To satisfy these, we can choose:

$$\ell = \left\lfloor \log_3 \frac{t_\ell}{T_2} - \frac{1}{2} \log_3 \log_3 \frac{t_\ell}{T_2} - 3 \right\rfloor \ .$$

Then, we have:

$$\begin{aligned} u = \left\lceil \log_3 \frac{t_\ell}{T_2} \right\rceil - \ell &\geq \log_3 \frac{t_\ell}{T_2} - \left(\log_3 \frac{t_\ell}{T_2} - \frac{1}{2} \log_3 \log_3 \frac{t_\ell}{T_2} - 3 + 1 \right) \\ &\geq \frac{1}{2} \log_3 \log_3 \frac{t_\ell}{T_2} + 2 \geq 2 + \log_3 \sqrt{\left\lfloor \log_3 \frac{t_\ell}{T_2} \right\rfloor} \geq 2 + \log_3 \sqrt{\ell} \ . \end{aligned}$$

This proves that $u \geq 1 + \left\lceil \log_3 \left(\frac{\pi}{6} \sqrt{\ell} \right) \right\rceil$ as required. The running time is given by Equation 11:

$$\begin{aligned} \sum_{i=1}^{\ell} \left(\prod_{j=i}^{\ell} (2k_j + 1) \right) (t_i - t_{i-1}) &= 3^{\ell-1} t_1 + \sum_{i=2}^{\ell} 3^{\ell-i} (t_i - t_{i-1}) \\ &\leq t_\ell + t_\ell + \sum_{i=1}^{\ell-1} 2 \times 3^{\ell-i-1} t_i \\ &\leq 2t_\ell + 2(\ell-1) 3^{\ell+u-1} T_2 \leq 2\ell t_\ell \end{aligned}$$

by definition of u. It is rather strange to obtain here a bound that depends only on t_ℓ, but it makes more sense if we remark that by definition: $p t_\ell^2 \leq T_2^2$ and so $t_\ell \leq \frac{T_2}{\sqrt{p}}$. In other words, our layered QAA averages the running times of the different testing steps with respect to the L_2 norm (like Ambainis' algorithm for the same problem [3]). By Lemma 8 the success probability is higher than:

$$P = \frac{p}{e} 3^{2\ell-2} \geq \frac{p}{3^8 e} \left(\frac{t_\ell}{T_2} \right)^2 \frac{1}{\log_3 \frac{t_\ell}{T_2}} \ . \tag{16}$$

At this point, we can directly use Lemma 4, since we have a lower bound on the success probability: a solution is found, with probability $\frac{1}{2}$, after a procedure that applies $\mathcal{O}\left(1/\sqrt{P}\right)$ QAA iterates. This gives a final time complexity of:

$$\mathcal{O}\left(\frac{1}{\sqrt{p}}\frac{T_2}{t_\ell} \times \sqrt{\log_3 \frac{t_\ell}{T_2}} \times \ell t_\ell\right) = \mathcal{O}\left(\left(\log \frac{t_\ell}{T_2}\right)^{3/2} \frac{T_2}{\sqrt{p}}\right) \quad . \tag{17}$$

We recover a logarithmic factor depending on the maximal running time, which is of the same order as Ambainis' [3]. To define the algorithm we actually only need to know T_2 and an upper bound on t_ℓ. We can use the upper bound $t_\ell \leq T_2/\sqrt{p}$. As for T_2 we can guess it by taking successive larger values until we find a valid upper bound.

3.5 Optimizing the Complexity Numerically

Another analysis is possible when, instead of bounds on their products, we know intervals on the α_i^2 of the form:

$$l_i^2 := \gamma_i^2(1-\varepsilon_i) \leq \alpha_i^2 \leq u_i^2 := \gamma_i^2(1+\varepsilon_i) \enspace .$$

Indeed, we observe that as long as we keep the iteration numbers sufficiently low, we can bound the final success probability using these known upper and lower bounds.

Lemma 11. *Assume that:* $\forall i, k_i \leq \left\lfloor \frac{\pi}{4}\frac{1}{\arcsin u_i} - \frac{1}{2}\right\rfloor$*. Let* ν_i^l *(lower bound) and* ν_i^u *(upper bound) be obtained by replacing the* α_i *by* l_i *and* u_i*, respectively, in the formulas of Lemma 5. Then we have:* $\nu_\ell^u \geq \nu_\ell \geq \nu_\ell^l$*.*

Proof. The upper bounds on k_i ensure that, regardless of the exact value of α_i, all inputs to sin stay in the interval $[0;\pi/2]$ where the function is increasing. The bounds on ν_ℓ then follows by a simple induction. □

Therefore, we can bypass Lemma 8 entirely and directly express the lower bound on the success probability ν_ℓ^2 of $\mathcal{B}_\ell$ as a function of the k_i:

$$\begin{cases} \nu_1^l = \sin\left[(2k_1+1)\arcsin(\gamma_1(1-\varepsilon_1))\right] \\ \forall i, \nu_i^l = \sin\left[(2k_i+1)\arcsin\left[\gamma_i(1-\varepsilon_i)\nu_{i-1}^l\right]\right] \end{cases} \tag{18}$$

which includes the case $\varepsilon_i = 0$ where we know the success probability exactly. Then, given the formula for the gate complexity of $\mathcal{B}_\ell$ as a function of the iteration numbers, our goal is to:

$$\text{minimize } G(\mathcal{B}_\ell)/{\nu_\ell^l}^2 \text{ under the constraints: } \forall i, k_i \leq \left\lfloor \frac{\pi}{4}\frac{1}{\arcsin u_i} - \frac{1}{2}\right\rfloor .$$

We first optimize this numerically, then take the floor of the values k_i obtained (in order to avoid going above the bounds). This last rounding is, in practice, insignificant for the complexity (even if the values are quite small, e.g. of the order of 10).

Note that in case the ε_i are particularly large, it is possible that the success probability of $\mathcal{B}_1$ becomes quite low, in which case we would have to compose it with an overcooked QAA as we did above.

4 Our main framework: Early-abort + Backtracking

Our strategy for variable-time QAA in Section 3 is essentially a recursive combination of QAAs which stop the amplified algorithm at its different steps. However, this does not model many cryptographic applications of QAA. In this section, we develop our main generic framework that combines *early-abort* and *backtracking* strategies.

In general, a *backtracking* algorithm explores a search space by making partial choices for partial values of the solution and being able to check whether a partial solution may lead a full solution, e.g., classically this can be seen as a depth-first tree search where it recognizes whether the current node can lead to a solution, and if it doesn't, returns to the parent node. Montanaro [23] showed a generic quantum speedup of backtracking algorithms. However, the algorithm is rendered inapplicable in our case by the fact that our steps are costly, and have different time complexities. Intuitively, the last step will cost much less than the first one. To gain advantage from this, we use a method based on nested QAAs. This requires knowledge of the success probabilities of each step. They are not necessary in the quantum tree search of [23], but they can usually be computed in cryptanalytic algorithms.

4.1 Setting and Definitions

Consider an input classical nested search problem NestedSearchProblem (see Algorithm 1) of ℓ layers defined by choice sets $C_1, \ldots, C_\ell$, filtering functions $f_1, \ldots, f_\ell$ and post-processing functions $d_1, \ldots, d_\ell$. The nested search problem functions f_i, d_i are transformed into corresponding quantum algorithms $\mathcal{A}_1, \ldots, \mathcal{A}_\ell$ and $\mathcal{D}_1, \ldots, \mathcal{D}_\ell$ that are deterministic and reversible. These quantum algorithms operate on a Hilbert space $\mathcal{H} = \mathcal{C}_1 \otimes \ldots \otimes \mathcal{C}_\ell \otimes \mathcal{W}_1 \otimes \ldots \mathcal{W}_\ell \otimes \mathcal{F}$:

1. $\mathcal{C}_i$ are quantum registers that encode the choice $c_i \in C_i$;
2. $\mathcal{W}_i$ is a workspace register for search level i that includes an encoding of the auxiliary data $s_i \in S_i$;
3. $\mathcal{F}$ is a sequence of flag qubits, where the i-th qubit is initially 0 and modified only by the i-th level quantum algorithm $\mathcal{A}_i$ to 1 if the corresponding partial solution $(c_1, \ldots, c_i)$ is not filtered; there is only one full solution $(c_1, \ldots, c_\ell)$;
4. $\mathcal{A}_i$ implements f_i and operates on the registers $\mathcal{C}_1, \ldots, \mathcal{C}_i, \mathcal{W}_1, \ldots, \mathcal{W}_i, \mathcal{F}$ but only modifies $\mathcal{W}_i$ and the i-th qubit flag in $\mathcal{F}$;
5. $\mathcal{D}_i$ implements d_i and operates on the registers $\mathcal{C}_1, \ldots, \mathcal{C}_i, \mathcal{W}_1, \ldots, \mathcal{W}_i, \mathcal{F}$ but only modifies $\mathcal{W}_i$.

Remark 2. Note that to create the quantum algorithms $\mathcal{A}_i$ and $\mathcal{D}_i$, first the classical algorithms f_i and d_i need to be transformed into reversible circuits. To that end, one often has to use a larger workspace than a simple encoding of the auxiliary data. A naive example, if one does not care about memory complexity, is to track all of their intermediate computations into the workspace. Otherwise, in addition to the workspace qubits, each $\mathcal{A}_i$ will also use a certain number of *ancilla* qubits which (by definition) are returned to their initial state $|0\rangle$.

Remark 3. Naturally, our results also apply to more generic deterministic and reversible quantum algorithms $\mathcal{A}_i$ and $\mathcal{D}_i$ that satisfy the above constraints. However, a very important feature of $\mathcal{A}_i$ and $\mathcal{D}_i$ is that they act *reversibly* on the previous workspace registers, i.e., they do not modify them. In the offline-Simon algorithm of [5], there is a single Grover search with a test that uses an external, previously constructed workspace. It is observed that the test does not act (exactly) reversibly on the workspace, and each iteration adds an error term that needs to be globally bounded. Our framework does not support this.

From the above definitions it follows that:

$$\mathcal{D}_i \circ \mathcal{A}_i \left|c_1, \dots, c_i\right\rangle \left|\psi_{c_1,\dots,c_{i-1}}\right\rangle \left|1_{i-1} 0_{\ell-i+1}\right\rangle \\ = \left|c_1, \dots, c_i\right\rangle \left|\psi_{c_1,\dots,c_i}\right\rangle \left|1_{i-1} b_{c_1,\dots,c_i} 0_{\ell-i}\right\rangle \quad . \tag{19}$$

The behavior of $\mathcal{A}_i$ on other states does not need to be specified. At the first and last steps, we have:

$$\mathcal{D}_1 \circ \mathcal{A}_1 \left|c_1\right\rangle \left|0\right\rangle \left|0_\ell\right\rangle = \left|c_1\right\rangle \left|\psi_{c_1}\right\rangle \left|b_{c_1} 0_{\ell-1}\right\rangle$$
$$\mathcal{D}_\ell \circ \mathcal{A}_\ell \left|c_1, \dots, c_\ell\right\rangle \left|\psi_{c_1,\dots,c_{\ell-1}}\right\rangle \left|1_{\ell-1} 0\right\rangle = \left|c_1, \dots, c_\ell\right\rangle \left|\psi_{c_1,\dots,c_\ell}\right\rangle \left|1_{\ell-1} b_{c_1,\dots,c_\ell}\right\rangle \quad .$$

In particular we could set $\mathcal{D}_\ell = I$ and merge $\mathcal{A}_\ell$ and $\mathcal{D}_\ell$ in a single algorithm.

There is only one full solution which is denoted as $(c_1^g, \dots, c_\ell^g)$. At each level, we define two probabilities:

- a *filtering probability*: $\alpha_i'^2$ is the probability that, starting from the good subpath $c_1^g, \dots, c_{i-1}^g$, a uniformly random c_i leads to a path that can still be extended, i.e., there is no early abort at step i on this path.
- a *success probability*: α_i^2 is the probability that, starting from the good subpath $c_1^g, \dots, c_{i-1}^g$, and assuming the filtering, by taking a uniformly random c_i, we extend to the good subpath $c_1^g, \dots, c_i^g$.

Given that there is exactly one solution, we have:

$$\forall i, \alpha_i^2 \alpha_i'^2 = \frac{1}{|C_i|}, \qquad \prod_{i=1}^{\ell} \alpha_i^2 \alpha_i'^2 = \prod_{i=1}^{\ell} \frac{1}{|C_\ell|} \quad . \tag{20}$$

Also, we set $\alpha'_\ell = 1$: there is no filtering at the final step, since we are immediately detecting if the path is good or not. Note that we only need to know the behavior of $\mathcal{A}_i$ on the good path; this is because the other paths do not produce false positives (the good path is the only way to obtain a "1" in the final flag qubit).

Thus, we can define $\left|G_i\right\rangle$ the uniform superposition of paths of the form $c_1^g, \dots, c_{i-1}^g, c_i$, which survive the filtering at step i (and contain, in particular, the good path $c_1^g, \dots, c_{i-1}^g, c_i^g$), and $\left|B_i\right\rangle$ the superposition of paths $c_1^g, \dots, c_{i-1}^g, c_i$ which are identified as bad at step i. Both superpositions include the state of the workspace. We have:

$$\begin{aligned}
\mathcal{D}_1 \circ \mathcal{A}_1 \left(\sum |c_1\rangle\right) |0\rangle |0_\ell\rangle &= \alpha'_1 |G_1\rangle |10_{\ell-1}\rangle + \beta'_1 |B_1\rangle |0_\ell\rangle \\
\mathcal{D}_2 \circ \mathcal{A}_2 |c_1^g\rangle \left(\sum |c_2\rangle\right) \left|\psi_{c_1^g}\right\rangle |10_{\ell-1}\rangle &= \alpha'_2 |G_2\rangle |110_{\ell-2}\rangle + \beta'_2 |B_2\rangle |10_{\ell-1}\rangle \\
\mathcal{D}_i \circ \mathcal{A}_i \left|c_1^g, \dots, c_{i-1}^g\right\rangle \left(\sum |c_i\rangle\right) \left|\psi_{c_1^g,\dots,c_{i-1}^g}\right\rangle |1_{i-1}0_{\ell-i+1}\rangle & \\
&= \alpha'_i |G_i\rangle |1_i 0_{\ell-i}\rangle + \beta_i |B_i\rangle |1_{i-1}0_{\ell-i+1}\rangle \\
\mathcal{A}_\ell \left|c_1^g, \dots, c_{\ell-1}^g\right\rangle \left(\sum |k_\ell\rangle\right) \left|\psi_{c_1^g,\dots,c_{\ell-1}^g}\right\rangle |1_{\ell-1}0\rangle &= \alpha_\ell |G_\ell\rangle |1_\ell\rangle + \beta_\ell |B_\ell\rangle |1_{\ell-1}0\rangle \quad .
\end{aligned}$$

Our goal is to find the good path $(c_1^g, \dots, c_\ell^g)$.

4.2 Description of the Algorithm

Our backtracking algorithm is a layered QAA that is structured differently from the variable-time quantum search. Here, the last step $\mathcal{D}_\ell \circ \mathcal{A}_\ell$ will be called more often, and the first step $\mathcal{D}_1 \circ \mathcal{A}_1$ less often. The classical intuition of backtracking is to go back to the previous computing step and try another path. In the quantum setting, we use levels of QAA to prune the invalid paths.

We take two sequence of integers $(k_1, \dots, k_\ell)$ and $(k'_1, \dots, k'_\ell = 0)$ that we will choose afterwards. We define ℓ algorithms $\mathcal{B}_i$ as follows. Each $\mathcal{B}_i$ acts on $\mathcal{H}$, but it only modifies the registers numbered from i to $\ell - 1$ (including workspace, index and flags). We will see $\mathcal{B}_i$ as an amplified version of the sequence of steps from i to ℓ. The definition of the algorithms are given in Algorithm 2, Algorithm 3 and Algorithm 4. Our complete algorithm is $\mathcal{B}_1$, which starts from input $|0\rangle$.

Analysis. For simplicity, here we often omit the workspace $|\psi\rangle$, since it is always a function of the current index registers: the construction ensures that we only apply $\mathcal{A}_i$ when the workspace contains the outputs of the step $i-1$.

We show that $\mathcal{B}_i$, when it starts *from the good path* $c_1^g, \dots, c_{i-1}^g$, has a probability of success ν_i^2 which is defined by a certain recursion; and otherwise, it outputs zero-flags only. Then $\mathcal{B}_1$ merely starts from input $|0\rangle$, and outputs a state which projects onto $|c_1^g, \dots, c_\ell^g\rangle$ with probability ν_1^2.

Lemma 12. *Using the definitions of Algorithm 2, Algorithm 3 and Algorithm 4:*

$$\begin{aligned}
&\forall i, \mathcal{B}_i \left|c_1^g, \dots, c_{i-1}^g\right\rangle |1_{i-1}0_{\ell-i+1}\rangle = \nu_i \left|c_1^g, \dots, c_\ell^g\right\rangle |1_\ell\rangle + \sqrt{1-\nu_i^2}\, |*\rangle |*0\rangle \\
&\forall i, \forall (c_1, \dots, c_{i-1}) \neq (c_1^g, \dots, c_{i-1}^g), \mathcal{B}_i |c_1, \dots, c_{i-1}\rangle |*\rangle = |*\rangle |0\rangle
\end{aligned}$$

where the $|*\rangle$ are unspecified superpositions since the last flag is always 0. We have: $\nu_\ell = \sin\left[(2k_\ell+1)\arcsin\alpha_\ell\right]$ and for all $i < \ell$:

$$\nu_i = \sin\left[(2k_i+1)\arcsin\left[\nu_{i+1}\alpha_i \sin((2k'_i+1)\arcsin(\alpha'_i))\right]\right] \quad . \tag{21}$$

Algorithm 2 $\mathcal{B}_\ell$: performs k_ℓ iterates of a quantum search on $c_\ell \in C_\ell$. Steps 3 to 5 correspond to the test of j_ℓ.

```
   Input: index registers, work register, flag registers (numbered from 1 to $\ell$)
   Output: acts in place on the registers numbered $\ell$
1: Apply a Hadamard transform on index register $\ell$
2: Repeat $k_\ell$ times
3:     Compute $\mathcal{D}_\ell \circ \mathcal{A}_\ell$ on $c_\ell$ and the current work register
4:     Flip the phase if the $\ell$-th flag qubit is 1
5:     Uncompute $\mathcal{D}_\ell \circ \mathcal{A}_\ell$
6:     Apply a Hadamard transform, an inversion around zero, and another
   Hadamard, all on choice register number $\ell$
7: EndRepeat
8: Compute $\mathcal{D}_\ell \circ \mathcal{A}_\ell$
```

Algorithm 3 $\mathcal{A}'_i$: produces a filtered superposition of indices at the next step. If there is no early abort at this step, then we have simply $\mathcal{A}'_i = \mathcal{A}_i$.

```
   Input: index registers, work register, flag registers
   Output: acts in place on the registers numbered $i$
1: Apply a Hadamard transform on index register $i$
2: Repeat $k'_i$ times
3:     Compute $\mathcal{A}_i$ on $c_i$ and the current work register
4:     Flip the phase if the $i$-th flag qubit is 1
5:     Uncompute $\mathcal{A}_i$
6:     Apply a Hadamard transform, an inversion around zero, and another
   Hadamard, all on choice register number $i$
7: EndRepeat
8: Compute $\mathcal{A}_i$
```

Algorithm 4 $\mathcal{B}_i$: performs a QAA on the algorithm $\mathcal{B}_{i+1} \circ \mathcal{D}_i \circ \mathcal{A}'_i$.

```
   Input: index registers, work registers, flag registers
   Output: acts in place on the registers numbered from $i$ to $\ell$
1: Compute $\mathcal{B}_{i+1} \circ \mathcal{D}_i \circ \mathcal{A}'_i$
2: Repeat $k_i$ times
3:     Flip the phase if the $\ell$-th flag qubit is 1
4:     Uncompute $\mathcal{B}_{i+1} \circ \mathcal{D}_i \circ \mathcal{A}'_i$
5:     Apply an inversion around zero on the registers numbered from $i$ to $\ell$
                                                              ▷ These are all
   the registers on which $\mathcal{A}'_i$, $\mathcal{D}_i$ and $\mathcal{B}_{i+1}$ act. The others are left unmodified, so the
   QAA is performed correctly without having to invert on the whole workspace.
6:     Compute $\mathcal{B}_{i+1} \circ \mathcal{D}_i \circ \mathcal{A}'_i$
7: EndRepeat
```

Proof. We do a descending recursion on the value of i, starting from $i = \ell$.

At $i = \ell$, $\mathcal{B}_\ell$ applies k_ℓ iterates of a quantum search on c_ℓ. Observe that:

$$\mathcal{A}_\ell \left|c_1^g, \ldots, c_{\ell-1}^g\right\rangle \sum \left|c_\ell\right\rangle \left|1_{\ell-1}0\right\rangle = \alpha_\ell \left|c_1^g, \ldots, c_\ell^g\right\rangle \left|1_\ell\right\rangle + \beta_\ell \left|c_1^g, \ldots, c_{\ell-1}^g\right\rangle \sum_{c_\ell \neq c_\ell^g} \left|c_\ell\right\rangle \left|1_{\ell-1}0\right\rangle$$

$$\implies \mathcal{B}_\ell \left|c_1^g, \ldots, c_{\ell-1}^g\right\rangle \left|1\right\rangle = \nu_\ell^2 \left|c_1^g, \ldots, c_\ell^g\right\rangle \left|1_\ell\right\rangle + \sqrt{1 - \nu_\ell^2} \left|*\right\rangle \left|1_{\ell-1}0\right\rangle$$

by the properties of QAA (Lemma 1), where $\nu_\ell = \sin\left[(2k_\ell + 1) \arcsin \alpha_\ell\right]$. If we start from a wrong path, then the flag is always 0, so even after QAA we obtain a superposition with only zero flags.

Next, we assume the lemma true for $i+1$ and prove it for i. We start by analyzing the behavior of $\mathcal{A}'_i$, assuming that we start from the good path $(c_1^g, \ldots, c_{i-1}^g)$. By Lemma 1, $\mathcal{A}'_i$ produces:

$$\begin{aligned} &\mathcal{A}'_i \left|c_1^g, \ldots, c_{i-1}^g\right\rangle \left|0\right\rangle \left|1_{i-1}0_{\ell-i+1}\right\rangle \\ &\qquad = \sin((2k'_i + 1) \arcsin \alpha'_i) \left|G_i\right\rangle \left|1_i 0_{\ell-i}\right\rangle \\ &\qquad + \cos((2k'_i + 1) \arcsin \alpha'_i) \left|B_i\right\rangle \left|1_{i-1}0_{\ell-i+1}\right\rangle \\ &\quad = \alpha_i \sin((2k'_i + 1) \arcsin \alpha'_i) \left|c_1^g, \ldots, c_i^g\right\rangle \left|1_i 0_{\ell-i}\right\rangle + \left|*\right\rangle \left|1_{i-1}0_{\ell-i+1}\right\rangle \quad . \end{aligned}$$

(Recall that $\left|G_i\right\rangle$ is the uniform superposition of paths $\left|c_1^g, \ldots, c_{i-1}^g, c_i\right\rangle$ passing the early abort at step i, and by definition, the probability of such a path to be $(c_1^g, \ldots, c_i^g)$ is α_i^2).

Then, we apply $\mathcal{B}_{i+1} \circ \mathcal{D}_i$ on this result. By linearity, and by our recurrence hypothesis, we obtain a superposition of the form:

$$\nu_{i+1} \alpha_i \sin((2k'_i + 1) \arcsin \alpha'_i) \left|c_1^g, \ldots, c_i^g\right\rangle \left|1_\ell\right\rangle + \left|*\right\rangle \left|0\right\rangle \quad . \tag{22}$$

Finally, $\mathcal{B}_i$ applies a QAA on $\mathcal{B}_{i+1} \circ \mathcal{D}_i \circ \mathcal{A}'_i$; it recognizes the good path using the ℓ-th flag. Using Lemma 1 again, we obtain the wanted formula for ν_i^2.

If we start from a wrong path, $\mathcal{A}'_i$ produces a superposition of wrong paths. Going through $\mathcal{B}_{i+1}$, the last flag written is always zero, so even after amplification this remains the case. □

Remark 4. The assumption of having exactly one solution path, which is quite common in cryptanalytic algorithms, allows us to reason only with respect to this path: in particular, we only need to know, or to bound correctly, the filtering ($\alpha_i'^2$) and success (α_i^2) probabilities *along this path.* Indeed, since we assume that the amplified algorithm $\mathcal{A}$ yields no false positives, the behavior on the other paths turn out to be irrelevant for our analysis.

Remark 5. Very rarely, some applications may need to replace $\mathcal{D}_i$ by a probabilistic procedure, with some success probability p_i, or even a quantum algorithm. As long as $\mathcal{D}_i$ is reversible and does not modify the previous workspaces, the formulas remain the same with α_i^2 replaced by $\alpha_i^2 p_i$. Here p_i, similarly to α_i^2, is the success probability of $\mathcal{D}_i$ *on the good branch.* The result of $\mathcal{D}_i$ is not tested. (If it could be tested, then it should rather become its own early-abort step).

4.3 Analysis of the Algorithm

From now on, we assume that we only know an interval on α_i^2 and $\alpha_i'^2$, of the form:

$$\begin{cases} l_i'^2 := \gamma_i'^2(1-\varepsilon_i) \leq \alpha_i'^2 \leq u_i'^2 := \gamma_i'^2(1+\varepsilon_i) \\ l_i^2 := \frac{1}{1+\varepsilon_i}\gamma_i^2 \leq \alpha_i^2 \leq u_i^2 := \frac{1}{1-\varepsilon_i}\gamma_i^2 \text{ where } \gamma_i^2 = \frac{1}{\gamma_i'^2 |C_i|} \end{cases} \tag{23}$$

We can show that as long as we keep the iteration numbers sufficiently low, we can bound the final success probability using these upper and lower bounds.

Lemma 13. *Assume that:* $\forall i, k_i \leq \frac{\pi}{4}\frac{1}{\arcsin u_i} - \frac{1}{2}$ *and* $\forall i < \ell, k_i' \leq \frac{\pi}{4}\frac{1}{\arcsin u_i'} - \frac{1}{2}$*. Let* ν_i^l *(lower bound) and* ν_i^u *(upper bound) be obtained by replacing the* α_i *and* α_i' *by* l_i *and* l_i'*, and by* u_i *and* u_i'*, respectively, in the formulas of Lemma 12. Then we have:* $\nu_1^u \geq \nu_1 \geq \nu_1^l$*.*

Proof. The upper bounds on the k_i and k_i' ensure that, regardless of the exact value of α_i and α_i', all inputs to sin stay in the interval $[0; \pi/2]$ where the function is increasing. The bound on ν_1 then follows by a simple induction. □

Therefore, the conditions of Lemma 13 allow to simplify the analysis of the algorithm, by focusing only on the bound of ν_1^l.

Lemma 14. *Let* $\nu_{\ell+1}^l = 1$*,* $k_\ell' = 0$ *and* $\alpha_\ell' = 1$*. Let* $s_i = (2k_i+1)^2 l_i^2$ *and* $s_i' = \sin^2\left[(2k_i'+1)\arcsin(l_i')\right]$*. Let* $d = \frac{\pi^2}{4} - 1$*. Then for all* $i < \ell$ *we have:*

$$s_i s_i' {\nu_{i+1}^l}^2 \left(1 - s_i s_i' {\nu_{i+1}^l}^2\right) \leq {\nu_i^l}^2 \leq s_i s_i' {\nu_{i+1}^l}^2 \left(1 + d s_i s_i' {\nu_{i+1}^l}^2\right) \tag{24}$$

and for $i = \ell$ *we have:*

$$s_\ell (1 - s_\ell) \leq {\nu_\ell^l}^2 \leq s_\ell (1 + d s_\ell) \tag{25}$$

The proof is similar to Lemma 6 and uses the same inequalities. Note that there is no need to bound the sin and arcsin in s_i'. In practice, we will expect the s_i' to be as close to 1 as possible. By recursively unfolding the inequalities, we obtain the following:

Lemma 15. *For all* i*, let* $S_i = \prod_{j=i}^{\ell} s_j$*. Let* $S_i' = \prod_{j=i}^{\ell} s_j'$*. Assume that for all* i*,* $S_i S_i' \leq \frac{c}{\ell}$ *for some constant* $c \leq \frac{4}{\pi^2}$*. Then for all* i*:* ${\nu_i^l}^2 \leq \frac{1}{\ell}$*. furthermore,* ν_1^2 *(the success probability of the whole algorithm) is lower bounded by:*

$$ {\nu_1^l}^2 \geq \frac{1}{e} S_\ell S_\ell' \ . \tag{26}$$

Proof. The proof is similar to Lemma 7, where we unfold recursively the upper bound in Equation 24 and replace s_i by $s_i s_i'$. For the lower bound, it goes exactly as before except that we now have the product of all s_i' to take into account. □

Afterwards, we can choose iteration numbers which satisfy the conditions of Lemma 13:

Lemma 16. *Choose:*

$$\begin{cases} k_\ell = \left\lfloor \frac{1}{2}\sqrt{\frac{c}{\ell}}\sqrt{|C_\ell|} - \frac{1}{2} \right\rfloor \\ \forall i < \ell \ s.t. \ \alpha_i'^2 \neq 1, k_i = \left\lfloor \frac{1}{2}\gamma_i'\sqrt{|C_i|(1-\varepsilon_i)} - \frac{1}{2} \right\rfloor \\ \forall i < \ell \ s.t. \ \alpha_i'^2 = 1, k_i = \left\lfloor \frac{1}{2}\sqrt{|C_i|} - \frac{1}{2} \right\rfloor \\ \forall i \ s.t. \ \alpha_i'^2 \neq 1, k_i' = \left\lfloor \frac{\pi}{4\arcsin(\gamma_i'\sqrt{1+\varepsilon_i})} - \frac{1}{2} \right\rfloor \\ \forall i \ s.t. \ \alpha_i'^2 = 1, k_i' = 0 \end{cases} \tag{27}$$

where $c = \frac{4}{\pi^2}$. Then the probability of success of $\mathcal{B}_1$ is lower bounded by:

$$\nu_1^2 \geq \frac{1}{e}\prod_i \frac{1}{1+\varepsilon_i}\prod_i (2k_i+1)^2\gamma_i^2 \prod_{i|\alpha_i'\neq 1} \sin^2\left((2k_i'+1)\arcsin(\gamma_i'\sqrt{1-\varepsilon_i})\right) \ . \tag{28}$$

Proof. First of all, we can easily check that these numbers satisfy the conditions of Lemma 13, especially the choice of k_i. Indeed we have for all x, $\arcsin x \leq \frac{\pi}{2}x$, so:

$$k_i \leq \frac{\pi}{4}\left(\arcsin\left(\frac{1}{\sqrt{1-\varepsilon_i}\gamma_i'\sqrt{|C_i|}}\right)\right)^{-1} - \frac{1}{2} \leq \frac{\pi}{4\arcsin u_i} - \frac{1}{2} \ .$$

Next, we have: $\forall i, (2k_i+1)^2\alpha_i^2 \leq 1$ and $(2k_\ell+1)^2\alpha_\ell^2 \leq \frac{c}{\ell}$ so $S_i \leq \frac{c}{\ell}$ for all i, and by Lemma 15: $\nu_1^2 \geq \frac{1}{e}S_\ell S_\ell'$. □

Finally, we give the gate and space complexities of the algorithm depending on k_i and k_i'. For the cost of an inversion around zero on r qubits, we use the costs of Lemma 2, i.e., $44r - 39$ Clifford+T gates (and 3 gates for the bit-flipping). We write $I(r) = 44r - 36$. The total number of ancilla qubits is the maximum between the ancillas of the $\mathcal{A}_i$, and $\ell + \sum_i (w_i + n_i) - 1$, which is the number required by the largest O_0 that we apply.

Recall that we note $G(\mathcal{A})$ the gate cost of the algorithm $\mathcal{A}$ and $S(\mathcal{A})$ the number of qubits on which it acts. Note that each step $\mathcal{A}$ includes the writing of its output flag qubit. We assume that $|C_i|$ are powers of 2: $|C_i| = 2^{n_i}$. Then the space and gate complexities of the $\mathcal{B}_i$ are given by the recursive formulas:

$$\begin{aligned} S(\mathcal{B}_\ell) &= w_\ell + n_\ell + 1 \\ G(\mathcal{B}_\ell) &= (2k_\ell+1)\left(G(\mathcal{A}_\ell) + n_1\right) + k_\ell I(w_\ell) \\ S(\mathcal{B}_i) &= S(\mathcal{B}_{i+1}) + w_i + n_i + 1 \\ G(\mathcal{B}_i) &= (2k_i+1)\Big(G(\mathcal{B}_{i+1}) + G(\mathcal{D}_i) + (2k_i'+1)\left(G(\mathcal{A}_i) + n_i\right) + k_i' I(n_i)\Big) \\ &\qquad + k_i I(S(\mathcal{B}_i)) \ . \end{aligned}$$

4.4 Optimizing the Complexity Numerically

Even though our final result is quite tight, we still have a multiplicative factor of order $\sqrt{e\ell}$ in the complexity. It can become a bit large for practical applications, especially those that require a small value of ℓ, e.g. less than 4, and do not have any errors. Fortunately, we can do the same as in Section 3.5.

We bypass Lemma 16 and directly focus on the value of ν_1^l, the lower bound of the success probability, which as we recall, is obtained by the following recursion:

$$\begin{cases} \nu_\ell^l = \sin\left[(2k_\ell+1)\arcsin\left(\frac{\gamma_\ell}{1+\varepsilon_\ell}\right)\right] \\ \forall i, \nu_i^l = \sin\left[(2k_i+1)\arcsin\left[\nu_{i+1}^l \frac{\gamma_i}{1+\varepsilon_i}\sin((2k_i'+1)\arcsin\gamma_i'\sqrt{1-\varepsilon_i})\right]\right] \end{cases} \tag{29}$$

Which holds both for $\varepsilon_i \neq 0$ and $\varepsilon_i = 0$. We simplify this problem by setting $k_i' = \left\lfloor \frac{\pi}{4}\frac{1}{\arcsin(\gamma_i'\sqrt{1+\varepsilon_i})} - \frac{1}{2}\right\rfloor$, since we will not gain anything by postponing the early-abort step. Then, by simply running $\mathcal{B}_1$ repeatedly, a solution is found with average time complexity $\frac{1}{\nu_1^2}G(\mathcal{B}_1)$. Thus, given the formula for the gate complexity of $\mathcal{B}_1$, as a function of the iteration numbers, our goal is to:

$$\text{minimize } \left(G(\mathcal{B}_1)/{\nu_1^l}^2\right) \text{ under the constraints: } \forall i, k_i \leq \frac{\pi}{4}\frac{1}{\arcsin\frac{\gamma_i}{\sqrt{1-\varepsilon_i}}} - \frac{1}{2} \ .$$

We can observe, as it was done in [20], that this direct optimization actually leads to lower probabilities of success, e.g. between 70% and 80%. In our code (in Supplementary Material) using Scipy, we also tried to optimize directly $G(\mathcal{B}_1)$ under the constraint ${\nu_1^l}^2 \geq 0.95$, but an easier approach is to increase the exponent on ν_1^l, which will naturally bring it closer to 1.

4.5 Analysis of the Memory

When turning a classical nested search into a quantum one, using our framework, some conversion of the memory model is needed (see Section 2 for an overview of quantum memory models). We can summarize it with the following rules:

- The workspace of the steps $\mathcal{A}_i$ and $\mathcal{D}_i$, but also their intermediate data (ancilla qubits), becomes qubits.
- If one of the $\mathcal{A}_i$ needs fast random access to one of the previous workspaces (e.g., reading from a table in memory), then the QRAQM model is required. Otherwise, performing *sequential* access, or accessing cells of fixed (global constant) position can be done with the standard quantum circuit model.
- If one of the $\mathcal{A}_i$ needs fast random access to a table that was initialized *before the first step*, then the QRACM model is required. The data in this table will remain classical, but the indices accessed will be put in superposition. Otherwise, performing *sequential* access, or accessing cells of fixed (global constant) position can be done with the standard quantum circuit model, and no QRACM would be required (only a classical memory).

5 Applications

To illustrate the versatility of our framework, we translate the quantum attacks on AES given in [7]. Since we are interested merely in the shape of these algorithms and their resulting complexities, we refer to [7] for the details of the attacks and their correctness.

The AES [12] is the standardized version of the block cipher Rijndael [11], which is a substitution-permutation network (SPN) operating on a 4×4 matrix of bytes. An AES round applies the operations **AddRoundKey** (XORs the current round key to the state), **SubBytes** (applies the S-Box S to each byte individually), **ShiftRows** (permutes the bytes), **MixColumns** (applies a linear operation to each column, defined by an MDS matrix).

The states of round i are denoted as k_i (round key), x_i (state after **AddRoundKey**), y_i (state after **SubBytes**), z_i (state after **ShiftRows**), w_i (state after **MixColumns**), and u_i (state obtained by applying the inverse **MixColumns** to k_i). The bytes of these states are numbered from $x_i[0]$ to $x_i[15]$ following a standard convention. A plaintext is denoted p and a ciphertext c.

In these attacks, we are given *classical chosen-plaintext* access to a black-box E_k implementing a reduced-round AES with a secret key k. The goal is to find k. A valid quantum attack must outperform the exhaustive search with Grover's algorithm, which always applies [7]. Since the AES S-Box S is the only nonlinear component, it dominates the cost of quantum circuits, and we estimate the complexity not by counting individual gates, but by counting S-Box circuits. Our numbers are rounded to the second decimal.

5.1 Square Attacks on AES

The Square attack on 6-round AES using the partial sums technique [16] is a good example of a simple backtracking algorithm, without early-abort. Its quantum version is given in Appendix A.2 of [7]. The time complexity given in [7] is $2^{44.73}$ S-Boxes.

Here we can take $\mathcal{D}_i = I$ for all i. The successive algorithms $\mathcal{A}_i$ correspond to constructing tables of smaller sizes. We consider a few *structures* containing 2^{32} plaintexts and ciphertexts of a specific shape, denoted c_i ($1 \leq i \leq 2^{32}$). The goal is to compute, for each structure, a sum of the form:

$$\sum_i S^{-1}\big(u_5[0] \oplus a_0 S^{-1}(c_i[0] \oplus k_6[0]) \oplus a_1 S^{-1}(c_i[1] \oplus k_6[1]) \\ \oplus a_2 S^{-1}(t_2 \oplus k_6[2]) \oplus a_3 S^{-1}(s_2 \oplus k_6[3])\big)$$

for known constants a_0, a_1, a_2, a_3 depending on **MixColumns**, and to find the choice of $k_6[0,1,2,3]$ and $u_5[0]$ for which this sum is equal to 0 for all structures. Instead of having to do 2^{32} computations per key guess, Algorithm 5 amortizes this cost thanks to intermediate tables.

Using the generic formula of Lemma 16, we obtain a worse time complexity than the one given in [7]. Indeed, we must take the following number of iterates for the successive steps: 127, 7, 7, 2, with the last one quite small to allow for a reduced probability of success. We obtain a count of $2^{44.24}$ S-Boxes for a probability of success $2^{-5.18}$. But in the attack, the calls to $\mathcal{A}_1$ dominate the time complexity. In [7] there are roughly $\frac{\pi}{2} 2^8$ such calls, but in our case, we need

Algorithm 5 Square attack on 6-round AES.

```
          | Compute 8 structures of $2^{32}$ classical CP queries
16 bits   | **Choose** $k_6[0], k_6[1]$
          |   For each structure, for each ciphertext $c_i$, compute:
          |   $(t_1, t_2, t_3) = a_0 S^{-1}(c_i[0] \oplus k_6[0]) \oplus a_1 S^{-1}(c_i[1] \oplus k_6[1]), c_i[2], c_i[3]$.
$\mathcal{A}_1$ |   Build a table $T_1$ of $2^{24}$ entries that stores, for each three-byte value $(t_1, t_2, t_3)$,
          |   how many times it appears.
          |   This costs $2^{32} \times 8 \times 2 = 2^{36}$ S-Boxes and space: $w_1 = 8 \times 2^{24} \times 32 = 2^{32}$
          |   qubits (we keep large 32-bit counters)
8 bits    | **Choose** $k_6[2]$
          |   For each 3-byte value $(t_1, t_2, t_3)$, compute: $t_1 \oplus a_2 S^{-1}(t_2 \oplus k_6[2]), t_3$
          |   by accessing $T_1$. Build a table $T_2$ of $2^{16}$ entries that stores, for each 2-byte
$\mathcal{A}_2$ |   value $(s_1, s_2)$, how many times it appears.
          |   This costs $8 \times 2^{24} \times 1 = 2^{27}$ S-Boxes and space: $w_2 = 8 \times 2^{16} \times 32 = 2^{24}$
          |   qubits (we keep large counters).
8 bits    | **Choose** $k_6[3]$
          |   For each 2-byte value $(s_1, s_2)$, compute $s_1 \oplus a_3 S^{-1}(s_2 \oplus k_6[3])$.
$\mathcal{A}_3$ |   Build a table $T_3$ that stores how many times each byte appears.
          |   This costs $8 \times 2^{16} \times 1 = 2^{19}$ S-Boxes and space: $w_3 = 8 \times 2^{8} \times 32 = 2^{16}$ qubits.
8 bits    | **Choose** $u_5[0]$
          |   Using the table, compute the sum.
$\mathcal{A}_4$ |   This costs: $8 \times 2^{8} \times 1$ S-Boxes and additional small computations,
          |   and approx. $8 \times 32 = 2^{8}$ qubits.
```

to re-amplify the last layer with 18 calls, so we will call $\mathcal{A}_1$ roughly 18×2^8 times and this is not competitive.

However, the numerical optimization (see Section 4.4) “sees” that $\mathcal{A}_1$ dominates, and amplifies the non-dominating steps to a success probability closer to 1. This results in the number of iterates 171, 11, 11, 11 for a success probability of 0.94 for $\mathcal{B}_1$, and a time complexity of $2^{44.70}$ S-Boxes. The *average* time complexity is only slightly higher at $2^{44.79}$.

5.2 DS-MITM Attack on 8-round AES-256

We consider the DS-MITM key-recovery attack on AES-256 given in [7]. Its complexity analysis relies on the following estimations:

- There are 40 S-Box differential equations of the form $S(x \oplus \Delta) = S(x) \oplus \Delta'$ for known Δ, Δ' in the path. We suppose that for the good path, all of them have 2 solutions exactly, and not 4. This happens for the whole path with probability $2^{-0.45}$ as estimated in [7]
- the success probability of $\mathcal{A}_3$ to $\mathcal{A}_6$ varies less than by 2^{-8} (it depends on the number of solutions for each column when Δx_3 and Δy_4 is fixed). This also comes from an estimate in [7]
- the computation of the equations in $\mathcal{D}_6$ cost less than 2^{10} S-Boxes (most of these computations are only linear)

Algorithm 6 DS-MITM attack on 8-round AES-256.

80 bits	**Choose** $k_0[0,5,10,15]$, $k_1[3]$, $u_7[1]$, $u_8[0,7,10,13]$
	Find a pair which satisfies the differential path (2^{53} S-Boxes)
$\mathcal{D}_1$	Compute the 2^5-sequence of differences $\delta w_5[5]$ ($< 2^{88}$ S-Boxes)
	Compute $\Delta x_2[4-7]$ and $\Delta y_5[3,4,9,14]$
64 bits	**Choose** $\Delta y_2[4-7], \Delta x_5[3,4,9,14]$
$\mathcal{A}_2$	Match $\Delta y_2[4-7], \Delta x_5[3,4,9,14]$ with $\Delta x_2[4-7]$, $\Delta y_5[3,4,9,14]$
Success: 2^{-8}	**Stop** if no match
$\mathcal{D}_2$	Compute the possible states
	Compute Δx_3 and Δy_4
	(Here we have assumed that the S-Box differential equations yield only two solutions)
32 bits	**Choose** $\Delta x_4[0-3]$
$\mathcal{A}_3$	Match $\Delta x_4[0-3]$ with Δx_3 and Δy_4
Succ.: $2^{-8}(1 \pm 2^{-8})$	**Stop** if no match
32 bits	**Choose** $\Delta x_4[4-7]$
$\mathcal{A}_4$	Match $\Delta x_4[4-7]$ with Δx_3 and Δy_4
Succ.: $2^{-8}(1 \pm 2^{-8})$	**Stop** if no match
32 bits	**Choose** $\Delta x_4[8-11]$
$\mathcal{A}_5$	Match $\Delta x_4[8-11]$ with Δx_3 and Δy_4
Succ.: $2^{-8}(1 \pm 2^{-8})$	**Stop** if no match
32 bits	**Choose** $\Delta x_4[12-15]$
$\mathcal{A}_6$	Match $\Delta x_4[12-15]$ with Δx_3 and Δy_4
Succ.: $2^{-8}(1 \pm 2^{-8})$	**Stop** if no match
$\mathcal{D}_6$	Using all the known states, write the equations ($< 2^{10}$ S-Boxes)
	Determine the values of $x_3, x_2[4-7], x_4[0-7,10,11,15]$
	(Here we assume that at most 4 different values are found, which leaves 2^9 choices in total at the next step.)
9 bits	**Choose** one of 2^9 possibilities for $x_4[8,9,12,13,14]$, $x_5[4,9,14]$
$\mathcal{A}_7$	Compute the expected δ-sequence ($2^5 \times 40$ S-Boxes)
Success: 2^{-9}	**Check** if it equals the expected sequence

- at most 4 different values are found after $\mathcal{D}_6$

The algorithm with our framework is different from the one in [7]. In particular, the number of iterations in the last QAA is considerably reduced, and we do not use several instances of Grover search internally to reduce the failure probability. The space complexity remains quite small, and we count only the number of S-Boxes applied. Using our formulas, we obtain an algorithm of complexity: $2^{129.53}$ (S-Boxes) with success probability $\geq 2^{-5.85}$. Most of this uncertainty comes from the factor $\frac{1}{e}$ and the reduction of the success probability that ensures the correctness of our algorithm. Using Lemma 4, we find that with *on average* 22 calls to this procedure and its inverse, we can bring the success probability to 1/2. This gives a complexity of approximately $2^{134.00}$ S-Boxes. It is already a better estimate than in [7].

By running a numerical optimization instead, we obtain a complexity $2^{132.07}$ S-Boxes with a success probability ≥ 0.95, which gives an average complexity $2^{132.15}$ to reach a success. We need to include the factor $2^{0.45}$ for the global success of the attack, so this gives $2^{132.60}$. This is actually quite close to what we would obtain by taking exactly the square root of search spaces in the classical complexity ($2^{131.27}$), suggesting that there is not much more to improve.

Acknowledgments. A.S. would like to thank Xavier Bonnetain and Ronald de Wolf for helpful discussions on quantum search and variable-time amplitude amplification. A.S. is supported by ERC-ADG-ALGSTRONGCRYPTO (project 740972).

References

1. Aaronson, S., Ambainis, A.: Quantum search of spatial regions. Theory Comput. **1**(1), 47–79 (2005)
2. Ambainis, A.: Quantum search with variable times. Theory Comput. Syst. **47**(3), 786–807 (2010)
3. Ambainis, A.: Variable time amplitude amplification and quantum algorithms for linear algebra problems. In: STACS. LIPIcs, vol. 14, pp. 636–647. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2012)
4. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: an ultra-lightweight block cipher. In: CHES. LNCS, vol. 4727, pp. 450–466. Springer (2007)
5. Bonnetain, X., Hosoyamada, A., Naya-Plasencia, M., Sasaki, Y., Schrottenloher, A.: Quantum attacks without superposition queries: The offline simon's algorithm. In: ASIACRYPT (1). LNCS, vol. 11921, pp. 552–583. Springer (2019)
6. Bonnetain, X., Jaques, S.: Quantum period finding against symmetric primitives in practice. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2022**(1), 1–27 (2022)
7. Bonnetain, X., Naya-Plasencia, M., Schrottenloher, A.: Quantum security analysis of AES. IACR Trans. Symmetric Cryptol. **2019**(2), 55–93 (2019)
8. Boura, C., Naya-Plasencia, M., Suder, V.: Scrutinizing and improving impossible differential attacks: Applications to CLEFIA, Camellia, LBlock and Simon. In: ASIACRYPT 2014. Proceedings, Part I. LNCS, vol. 8873, pp. 179–199. Springer (2014)

9. Brassard, G., Hoyer, P., Mosca, M., Tapp, A.: Quantum amplitude amplification and estimation. Contemporary Mathematics **305**, 53–74 (2002)
10. Chakraborty, S., Gilyén, A., Jeffery, S.: The power of block-encoded matrix powers: Improved regression techniques via faster hamiltonian simulation. In: ICALP. LIPIcs, vol. 132, pp. 33:1–33:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019)
11. Daemen, J., Rijmen, V.: AES Proposal: Rijndael. Submission to the NIST AES competition (1999)
12. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Information Security and Cryptography, Springer (2002)
13. Davenport, J.H., Pring, B.: Improvements to quantum search techniques for block-ciphers, with applications to AES. In: SAC. LNCS, vol. 12804, pp. 360–384. Springer (2020)
14. Demirci, H., Selçuk, A.A.: A meet-in-the-middle attack on 8-round AES. In: FSE. LNCS, vol. 5086, pp. 116–126. Springer (2008)
15. Faugère, J., Horan, K., Kahrobaei, D., Kaplan, M., Kashefi, E., Perret, L.: Fast quantum algorithm for solving multivariate quadratic equations. IACR Cryptol. ePrint Arch. p. 1236 (2017)
16. Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D.A., Whiting, D.: Improved cryptanalysis of Rijndael. In: FSE. LNCS, vol. 1978, pp. 213–230. Springer (2000)
17. Frixons, P., Schrottenloher, A.: Quantum security of the legendre PRF. Mathematical Cryptology **1**(2), 52–69 (Mar 2022), https://journals.flvc.org/mathcryptology/article/view/130578
18. Grassl, M., Langenberg, B., Roetteler, M., Steinwandt, R.: Applying grover's algorithm to AES: quantum resource estimates. In: PQCrypto. Lecture Notes in Computer Science, vol. 9606, pp. 29–43. Springer (2016)
19. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: STOC. pp. 212–219. ACM (1996)
20. Jaques, S., Naehrig, M., Roetteler, M., Virdia, F.: Implementing grover oracles for quantum key search on AES and lowmc. In: EUROCRYPT (2). LNCS, vol. 12106, pp. 280–310. Springer (2020)
21. Kimmel, S., Lin, C.Y., Lin, H.: Oracles with costs. In: TQC. LIPIcs, vol. 44, pp. 1–26. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2015)
22. Kuperberg, G.: Another subexponential-time quantum algorithm for the dihedral hidden subgroup problem. In: TQC. LIPIcs, vol. 22, pp. 20–34. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2013)
23. Montanaro, A.: Quantum-walk speedup of backtracking algorithms. Theory Comput. **14**(1), 1–24 (2018)
24. Nielsen, M.A., Chuang, I.: Quantum computation and quantum information (2002)
25. Pring, B.: Exploiting preprocessing for quantum search to break parameters for *MQ* cryptosystems. In: WAIFI. LNCS, vol. 11321, pp. 291–307. Springer (2018)
26. Schwabe, P., Westerbaan, B.: Solving binary *MQ* with grover's algorithm. In: SPACE. LNCS, vol. 10076, pp. 303–322. Springer (2016)
27. Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In: CRYPTO. LNCS, vol. 3621, pp. 17–36. Springer (2005)

Appendix

A Search with Two Oracles and Exhaustive Key Search

A simple, but common, example of variable-time algorithm, as studied in Section 3, is the *search with two oracles* which was considered e.g. in [13,21].

Consider a quantum search on $\{0,1\}^n$ with two test functions f_1 and f_2, implemented as two oracles O_{f_1} and O_{f_2}, with $X_1 := f_1^{-1}(1)$, $X_2 := f_2^{-1}(1)$, $|X_1| = \alpha_1^2 2^n$, $|X_2 \cap X_1| = \alpha_2^2\alpha_1^2 2^n$ where the combined success probability $p = \alpha_1^2\alpha_2^2$ is known. Here we use a layered QAA with two levels and two differing tests O_{f_1} and O_{f_2} (which write in two different flag results). We compute the success probability as:

$$\sin^2\left[(2k_2+1)\arcsin\left[\alpha_2\sin\left((2k_1+1)\arcsin\alpha_1\right)\right]\right] \quad .$$

As we have seen before, α_1 does not need to be known exactly; we only need to ensure that the internal QAA amplifies only to a small success probability, and does not overamplify. Indeed, if an upper bound $\alpha_1 \leq \alpha$ is given, we have:

$$\begin{aligned}\alpha_2\sin\left((2k_1+1)\arcsin\alpha_1\right) &\geq \alpha_2\alpha_1(2k_1+1)\left(1-\frac{(2k_1+1)^2\alpha_1^2\pi^2}{36}\right)\\ &\geq \sqrt{p}(2k_1+1)\left(1-\frac{(2k_1+1)^2\alpha^2\pi^2}{24}\right) \quad .\end{aligned}$$

We can then choose k_1 and k_2 to maximize the success probability:

$$\begin{aligned}&\sin^2\left[(2k_2+1)\arcsin\left[\sqrt{p}(2k_1+1)\left(1-\frac{(2k_1+1)^2\alpha^2\pi^2}{24}\right)\right]\right]\\ &\qquad\geq \sin^2\left[\sqrt{p}(2k_2+1)(2k_1+1)\left(1-\frac{(2k_1+1)^2\alpha^2\pi^2}{24}\right)\right] \quad .\end{aligned}$$

To do so we choose: $k_2 = \left\lfloor \pi/\left(4\sqrt{p}(2k_1+1)\left(1-\frac{(2k_1+1)^2\alpha^2\pi^2}{24}\right)\right)\right\rfloor$. This ensures that the success probability is bigger than:

$$\begin{aligned}&1-\sin^2\left[\sqrt{p}(2k_1+1)\left(1-\frac{(2k_1+1)^2\alpha^2\pi^2}{24}\right)\right]\\ &\qquad\geq 1-\sqrt{p}(2k_1+1)\left(1-\frac{(2k_1+1)^2\alpha^2\pi^2}{24}\right) \quad .\end{aligned}$$

Let T_1 and T_2 be the gate counts of O_{f_1} and O_{f_2}. The time complexity is:

$$T = k_2(2k_1+1)T_1 + k_2T_2 \leq \frac{\pi}{4\sqrt{p}}T_1\frac{1+\frac{T_2}{T_1(2k_1+1)}}{\left(1-\frac{(2k_1+1)^2\alpha^2\pi^2}{24}\right)} \quad . \tag{30}$$

Block Cipher Exhaustive Key Search. Consider a block cipher E_K and several given plaintext-ciphertext pairs $(P_i, C_i), i \leq \ell$, such that with overwhelming probability there is a single key K such that $E_K(P_i) = C_i$ for all i. We are looking for this K. For example, with AES-256, $\ell = 3$ would be enough. For all intents and purposes, we can consider $\ell = 5$, which works for most practical block ciphers, e.g. AES [12], Present [4], etc. Furthermore, we can consider a key size $\kappa := |K| \geq 64$ and a block size $n \geq 32$.

Let trunc be a truncation of n bits to 32 well-chosen positions (depending on the cipher attacked). Let O_{f_1} and O_{f_2} be two quantum oracles evaluating the test functions:

$$\begin{cases} f_1(K) = 1 \iff \text{trunc}(E_K(P_1)) = \text{trunc}(C_1) \\ f_2(K) = 1 \iff \forall i, E_K(P_i) = C_i \end{cases} \tag{31}$$

Since $\ell = 5$ plaintext-ciphertexts suffice, we can consider a ratio of gate counts of $T_2/T_1 = 10$ (this is assuming that computing the truncation is twice as efficient as computing the whole ciphertext). The success probability of f_1 can be bounded using a Chernoff-Hoeffding bound (for this we have to assume that E is selected at random from all possible block ciphers): $\Pr\left(\alpha_1^2 \geq 2^{-31}\right) \leq e^{-2^{\kappa-32}/3}$. Since $\kappa \geq 64$, this event has negligible probability. So we can assume that $\alpha_1^2 \leq 2^{-31}$. The optimal choice of k_1 is $k_1 = 1480$ for which the multiplying factor is smaller than $1 + 2^{-7.625} \leq 2^{0.0073}$. This means that the cost of f_2 is completely amortized. The corresponding success probability is bigger than:

$$1 - 2^{-64}(2k_1 + 1)\left(1 - \frac{(2k_1+1)^2 2^{-31}\pi^2}{24}\right) \geq 1 - 2^{-52.47} \quad . \tag{32}$$

Fact 1. *For any reasonable block cipher with block size $n \geq 32$ and key size $\kappa \geq 64$, if O_{f_1} is the oracle defined above (testing only 32 bits of a single plaintext), then the complexity of a quantum exhaustive search of the key (to reach an overwhelming success) is smaller than: $\frac{\pi}{4}2^{\kappa/2+0.01}T_1$.*

B Search with Independent Tests

In this section, we tackle the problem of *search with many independent tests*, which arises in several cryptographic applications. It corresponds to a variable-time amplitude amplification, as given in Section 3.

B.1 Search with Independent Tests (Asymptotic)

We consider the following problem, which arises in several cryptanalytic applications.

Problem 1 (Sequence of independent tests). Let $f_1, \ldots, f_m$ be m functions: $f_i : \{0,1\}^n \to \{0,1\}$. Let $X_i = \{x \in \{0,1\}^n, \forall j \leq i, f(x) = 1\}$, with $X_0 = \{0,1\}^n$. Clearly we have $\forall i, X_{i+1} \subseteq X_i$. Assume that the f_i are all independent random boolean functions, which can be evaluated in time t. Sample from X_m.

Typically we will have $m \leq n+3$ since with the assumption of independence, this ensures that $|X_m| = 1$ with a large probability. For intermediate values of i, $|X_i|$ does not deviate too much from its average due to the multiplicative Chernoff-Hoeffding bound:

$$\forall \delta \geq 0, \Pr(|X_i| \geq 2^{n-i}(1+\delta)) \leq e^{-2^{n-i}\delta^2/(2+\delta)} \leq e^{-2^{n-i}\delta^2/2} \quad ,$$

where we take $\delta = 2^{-(n-i)/3}$ to obtain:

$$\forall i, \Pr(|X_i| \geq 2^{n-i} + 2^{2(n-i)/3}) \leq e^{-2^{(n-i)/3-1}} \quad .$$

Using a union bound, we can ensure that most of the X_i are close to their average size:

$$\begin{aligned}\Pr(\exists i \leq n-9, |X_i| \geq 2^{n-i} + 2^{2(n-i)/3}) &\leq \sum_{i=0}^{n-9} e^{-2^{(n-i)/3-1}} \leq \sum_{i=9}^{\infty} \left(e^{-1}\right)^{2^{i/3-1}} \\ &\leq \sum_{i=9}^{\infty} (e^{-1})^{\frac{4\ln 2}{3}(i-9)+4} \\ &\leq e^{-4} \sum_{i=0}^{\infty} \left(2^{-4/3}\right)^i \leq 0.031 \quad .\end{aligned}$$

In particular $|X_i| \leq 2^{n-i}\frac{9}{8}$, which simplifies the asymptotic computations. We will now cut the sequence $f_1, \ldots, f_n$ into a variable-time algorithm $\mathcal{A}_\ell \circ \cdots \circ \mathcal{A}_1$. Each $\mathcal{A}_i$ will run m_i successive functions f_i, so that in total $m_1 + \ldots + m_\ell = n-9$. A final search will be performed for the remaining conditions, but it will only add a constant overhead. Due to the Chernoff bounds, we know that the cumulative probabilities of success of the $\mathcal{A}_i$ are upper bounded by: $\prod_{j=1}^{i} \alpha_j^2 \leq 2^{-\sum_{j=1}^{i} m_j + 1}$. So we can take for the number of iterates k_i:

$$k_1 = \left\lfloor \frac{1}{2}\sqrt{\frac{c}{\ell \times (9/8)}}\sqrt{2^{m_1}} - \frac{1}{2} \right\rfloor, \forall i \geq 1, k_i = \left\lfloor \frac{1}{2}\sqrt{2^{m_i}} - \frac{1}{2} \right\rfloor \quad ,$$

where $c = \frac{\pi^2}{4}$, which are sufficient to ensure the conditions of Lemma 8:

$$\forall i \geq 1, \prod_{j=1}^{i}(2k_j+1)^2 \leq \frac{c}{\ell} 2^{m_1+\ldots+m_i}\frac{8}{9} \implies \prod_{j=1}^{i}(2k_j+1)^2\alpha_j^2 = \prod_{j=1}^{i}(2k_j+1)^2 \frac{|X_i|}{2^n} \leq \frac{c}{\ell} \quad .$$

By Equation 11 the complexity of the full procedure is upper bounded by:

$$\begin{aligned}\sum_{i=1}^{\ell}\left(\prod_{j=i}^{\ell}(2k_j+1)\right) m_i &\leq \left(\sum_{i=2}^{\ell}\sqrt{2^{m_i+\ldots+m_\ell}}m_i\right) + \sqrt{2^{m_1+\ldots+m_\ell}}\sqrt{\frac{8c}{9\ell}}m_1 \\ &\leq \sqrt{2^{n-9}}\left(\sqrt{\frac{8c}{9\ell}}m_1 + \frac{m_2}{2^{m_1/2}} + \frac{m_3}{2^{(m_1+m_2)/2}} + \ldots + \frac{m_\ell}{2^{(m_1+\ldots+m_{\ell-1})/2}}\right) \quad .\end{aligned}$$

It appears clearly that when n is very large, we can minimize the complexity by choosing $m_2, m_3, \ldots, m_\ell$ as follows:

$$m_\ell = n - 9 - \left\lfloor \log_{\sqrt{2}} n \right\rfloor, \forall i, m_{\ell-i} = \left\lfloor \log_{\sqrt{2}}^{(i)} n \right\rfloor - \left\lfloor \log_{\sqrt{2}}^{(i+1)} n \right\rfloor, m_1 = \left\lfloor \log_{\sqrt{2}}^{(\ell-1)} n \right\rfloor$$

where we have used an iterated logarithm in base $\sqrt{2}$. As long as all these numbers are strictly positive, we have for all $i \geq 1$:

$$\frac{m_i}{2^{(m_1+\ldots+m_{i-1})/2}} \leq 1$$

and so the time complexity is upper bounded by $2^{(n-9)/2}\left(\sqrt{\frac{8c}{9\ell}} \log_{\sqrt{2}}^{(\ell-1)} n + \ell\right)$. We are still free to choose ℓ under the condition $k_1 \geq 1$ i.e. $\sqrt{\frac{8c}{9\ell}}\sqrt{2^{m_1}} \geq \frac{3}{2}$. It can be remarked that $\ell = \mathcal{O}(\log^*_{\sqrt{2}} n)$, and the complexity (after the final amplification) to have a success probability $\frac{1}{2}$ is $\mathcal{O}\left((\log^*_{\sqrt{2}} n)^{3/2} 2^{n/2}\right)$.

B.2 Search with Independent Tests (Exact)

For cryptographically relevant parameters, we estimate that cutting the independent tests in three groups should be enough. We select two parameters m_1, m_2; we first perform m_1 tests, then m_2 tests, then the remaining $m - m_1 - m_2$ where $m = n + 3$ to ensure a single solution. We count the complexity in number of tests:

$$m_1(2k_1+1)(2k_2+1)(2k_3+1) + m_2(2k_2+1)(2k_3+1) + (m - m_1 - m_2)(2k_3+1) \ .$$

Since the tests are independent, the probabilities of success of the three steps, $\alpha_1^2 = \frac{|X_{m_1}|}{2^n}$, $\alpha_2^2 = \frac{|X_{m_1} \cap X_{m_1+m_2}|}{|X_{m_1}|}$, $\alpha_3^2 = \frac{1}{2^n \alpha_1^2 \alpha_2^2}$, can be bounded using Chernoff-Hoeffding bounds. There are on average 2^{n-m_1} elements passing the first step and $2^{n-m_2-m_1}$ elements passing the second. We have for all ε_1 and ε_2:

$$\begin{cases} \Pr\left(\left||X_{m_1}| - 2^{n-m_1}\right| \geq \varepsilon_1 2^{n-m_1}\right) \leq 2\exp\left(\frac{-\varepsilon_1^2 2^{n-m_1}}{3}\right) \\ \Pr\left(\left||X_{m_1} \cap X_{m_1+m_2}| - 2^{n-m_1-m_2}\right| \geq \varepsilon_2 2^{n-m_1-m_2}\right) \leq 2\exp\left(\frac{-\varepsilon_2^2 2^{n-m_1-m_2}}{3}\right) \end{cases} \tag{33}$$

Therefore, assuming that $n - m_1 - m_2 \geq 12$, we can take both $\varepsilon_1 = \varepsilon_2 = 2^{-4}$ and these two events occur with overwhelming probability. This gives bounds:

$$\begin{cases} \alpha_1^2 \in \left[2^{-m_1}(1-\varepsilon_1); u_1^2 := 2^{-m_1}(1+\varepsilon_1)\right] \\ \alpha_2^2 \in \left[2^{-m_2}\frac{1-\varepsilon_2}{1+\varepsilon_1}; u_2^2 := 2^{-m_2}\frac{1+\varepsilon_2}{1-\varepsilon_1}\right] \\ \alpha_3^2 \in \left[2^{-n+m_1+m_2}\frac{1}{1+\varepsilon_2}; u_3^2 := 2^{-n+m_1+m_2}\frac{1}{1-\varepsilon_2}\right] \end{cases} \tag{34}$$

Thus, after choosing values for m_1 and m_2, we can follow the approach of Section 3.5: we optimize the complexity as a function of k_1, k_2, k_3 (counting the total number of individual tests performed) under the constraints:

$$k_1 \leq \left\lfloor \frac{\pi}{4}\frac{1}{\arcsin u_1} - \frac{1}{2} \right\rfloor, k_2 \leq \left\lfloor \frac{\pi}{4}\frac{1}{\arcsin u_2} - \frac{1}{2} \right\rfloor, k_3 \leq \left\lfloor \frac{\pi}{4}\frac{1}{\arcsin u_3} - \frac{1}{2} \right\rfloor . \tag{35}$$

We can then try with different m_1 and m_2 and see which ones perform best. For cryptographic parameters, n usually does not exceed 2^{10}, and we expect m_1 to be quite small, so there are not many parameters to try.

B.3 Application: Solving Binary Multi-variate Quadratic Systems

As an example, we can consider the problem of solving binary quadratic equation systems using exhaustive search, which was considered in [26] and improved in [25] (note that an asymptotically better algorithm was given in [15], but its exact complexity has not been analyzed to date). Assume that we have a system of n quadratic equations in n boolean variables, with a single solution: $\forall i, f_i(\mathbf{x}) = 0$. The algorithm in [26] essentially performs a Grover search on the whole search space $\{0,1\}^n$ for $\mathbf{x}$, which tests each equation separately and checks if $f_i(\mathbf{x}) = 0$ for all x. If an equation can be tested in time t (returning a single bit), this gives a time complexity: $\left\lfloor \frac{\pi}{4} 2^{n/2} \right\rfloor nt$.

By applying our framework, for $n = 128$ we obtain an average complexity of $2^{67.93}t$ with $m_1, m_2 = 5, 10$; for $n = 256$ we obtain $2^{132.02}t$ with $m_1, m_2 = 5, 12$. The improvement over the naive exhaustive search is comparable to the *preprocessing* method used in [25]; however both methods are different, since the preprocessing uses the structure of the quadratic equations. It might be possible to combine them both to further reduce the cost, and we leave this as an interesting open question.

Our formula also applies to the problem of recovering the secret in the Legendre PRF via Grover search (it generalizes the search with early abort given in [17]).