Quantum Security Analysis of AES

Xavier Bonnetain[1,2], María Naya-Plasencia[2] and André Schrottenloher[2]

[1] Sorbonne Université, Collège Doctoral, F-75005 Paris, France
[2] Inria de Paris, France
{xavier.bonnetain,maria.naya_plasencia,andre.schrottenloher}@inria.fr

Abstract. In this paper we analyze for the first time the post-quantum security of AES. AES is the most popular and widely used block cipher, established as the encryption standard by the NIST in 2001. We consider the secret key setting and, in particular, AES-256, the recommended primitive and one of the few existing ones that aims at providing a post-quantum security of 128 bits. In order to determine the new security margin, i.e., the lowest number of non-attacked rounds in time less than 2^{128} encryptions, we first provide generalized and quantized versions of the best known cryptanalysis on reduced-round AES, as well as a discussion on attacks that don't seem to benefit from a significant quantum speed-up.

We propose a new framework for structured search that encompasses both the classical and quantum attacks we present, and allows to efficiently compute their complexity. We believe this framework will be useful for future analysis.

Our best attack is a quantum Demirci-Selçuk meet-in-the-middle attack. Unexpectedly, using the ideas underlying its design principle also enables us to obtain new, counter-intuitive classical TMD trade-offs. In particular, we can reduce the memory in some attacks against AES-256 and AES-128.

One of the building blocks of our attacks is solving efficiently the AES S-Box differential equation, with respect to the quantum cost of a reversible S-Box. We believe that this generic quantum tool will be useful for future quantum differential attacks.

Judging by the results obtained so far, AES seems a resistant primitive in the post-quantum world as well as in the classical one, with a bigger security margin with respect to quantum generic attacks.

Keywords: AES, symmetric cryptanalysis, quantum cryptanalysis, classical cryptanalysis, quantum algorithms, security margin, amplitude amplification, post-quantum security, DS-meet-in-the-middle, square attack.

1 Introduction

For a few years now, the cryptographic community has been worried about the security of asymmetric primitives against quantum adversaries due to Shor's algorithm [Sho94], while the common knowledge suggested that doubling the key lengths for symmetric primitives would counter any problem. Nowadays, the quantum security of symmetric primitives is not taken for granted anymore. As our confidence is based on cryptanalysis as an empirical measure of the security, in order to determine if a primitive will be secure against quantum adversaries, we have to know first how these adversaries could attack the primitive by *quantum cryptanalysis*.

Many new results [KM10, KM12, Kap14], most in the last three years, like [KLLN16a, KLLN16b, LM17, CNS17, HSX17, Bon18], have shown that a lot is yet to be done to prepare symmetric cryptography for the post quantum world. Some of them [KM12, KLLN16a, LM17] show that constructions proven classically secure can be broken in some quantum adversary models; while others have shown ways for a quantum adversary to speed up classical attacks [Kap14, KLLN16b, CNS17, HSX17]. As a side consequence of these recent results, the NIST, that has just launched a competition for recommending new lightweight primitives, explicitely asked in the report on lightweight cryptography NISTIR8114 [MBT+17] [1], that the algorithms submitted to the project should be quantum-safe when long-term security is needed.

Due to Grover's algorithm [Gro96], that allows to perform an exhaustive search in the square root of the classical time, primitives providing 128-bits of security against quantum adversaries need to have a key length of at least 256 bits. That is the main reason why the actual recommendation for encryption with post-quantum security is the version of AES [DR99] with a 256-bit key. Some results have been published regarding generic attacks, and the cost of applying Grover to AES [GLRS16]. Despite the fact that there is an enormous number of published classical attacks on reduced-round versions of AES, allowing to determine its security margin and endowing AES of the confidence needed for a standard, no quantum cryptanalysis or quantum security analysis is yet known. Without this analysis, there is no way of determining if the security margin (i.e. how far is the primitive from being broken) is better or worse than in the classical scenario.

Let us precise here that the normal definition of broken is that a better attack than the generic ones (like for instance, exhaustive key search) exists. As in the quantum scenario the generic exhaustive search time is in the square root of the classical time, some attacks that work for a certain number of rounds in a classical setting might not work anymore in the quantum one, if they cannot profit from a quadratic speed-up. The converse could also occur with a higher than quadratic speed-up, for example thanks to the use of Simon's algorithm in a particular attacker setting [KM10, KM12, KLLN16a].

On quantum vs classical security margin. In a post-quantum future, we can presume that the expected security of a primitive will be given by its best generic attack (*i.e.* Grover), and that the security margin of this primitive will be determined by the highest number of rounds cryptanalyzed with any attack more efficient than this exhaustive search. Therefore, we believe that the logical evolution is that the *classical* or *quantum* surnames will disappear, and the most efficient attacks, possibly using quantum tools, will be the most important information regarding how far a primitive is from being broken. From this point of view, our results are the first step towards determining this future and unique security margin for AES, and in particular AES-256.

1.1 Motivation

AES security against quantum adversaries. An algorithm of the importance of AES should have a detailed post-quantum security evaluation, that should be continuous and evolve through time, as it is the case for the classical setting. For now and to the best of our knowledge, the only results at hand in this direction are precise Grover-quantum resource estimates [GLRS16], a general discussion on the generic attack [RMYK17], an analysis of Grover combined with side channel attacks [MMOS18], as well as generic algorithms that could target the internal state using multiple preimages [BB18, CNS17]. We do not know anything about its quantum security margin, *i.e.* the number of rounds broken by a quantum adversary: is it the same than for the classical scenario or does it differ?

The aim of this paper is to propose a starting point in the secret-key setting (the most meaningful one, see for instance [DR12] for a discussion on the AES related-key attacks): we provide an extensive quantum evaluation of AES. For this we first perform the tedious task of generalizing, rewriting, optimizing and quantizing the families of attacks

[1] https://nvlpubs.nist.gov/nistpubs/ir/2017/NIST.IR.8114.pdf

that provide the best known classical cryptanalysis on AES, as it was done for instance in [KLLN16b] with respect to differential and linear attacks. From these previous results we also learned that "quantizing" the best classical attack does not always provide the best quantum attack. Considering all the classically efficient ones and comparing them seems to us to be the most reasonable approach. We point out that, from the beginning, it seemed much more likely for classical attacks to stop working in a quantum setting than the other way around (*i.e.* finding attacks with an exponential speedup).

1.2 Main Results

First, we propose a *new framework* for quantum and classical structured search, which allows to concisely present our algorithms, and compute their complexities. Second, we use this framework to present new quantum attacks that are quantum versions of the most efficient cryptanalysis families on reduced-round AES. While some of these families do not benefit from a competitive/significant speed up (and therefore won't be developed here), we managed to accelerate two. Though we consider several quantum models for the attacker, our attacks can be placed in the Q1 model, where the attacker has access to a quantum computer but is restricted to classical encryption/decryption queries. Next we apply these families of attacks, quantum square attacks and quantum Demirci-Selçuk Meet-in-the-middle attacks (DS-MITM from now on)[2], to the AES and obtain:

- Square attacks: we design quantum square attacks on 6-round AES, 7-round AES-192 and 7-round AES-256.
- Demirci-Selçuk Meet-in-the-Middle: by rewriting and reordering the phases of the attack, we are able to design a quantum 8 rounds of AES-256, hence effectively speeding up the classical attack by nearly a quadratic factor. In the classical setting, DS-MITM provide the best single-key attacks, along with impossible differentials (for which we did not find a significant speed-up). It covers up to 9 rounds of AES-256.

Second, we also provide:

- A detailed evaluation of the cost of Grover exhaustive search, that defines the security of the corresponding AES instances.
- Quantum tools to efficiently leverage the differential properties of the AES S-Box with a very small memory, a building block which could find applications outside the scope of this paper, and justification on our extensive usage of nested Grover procedures.
- New classical TMD trade-offs for DS-MITM attacks: the ideas that allow us to accelerate quantumly this type of attacks can also be applied classically on AES-256 and AES-128, giving reductions in memory needs and new tradeoffs. We believe that studying this new line of research might improve even more the overall complexities when combined with other technical ideas. We are able to improve the best known attack on 9 rounds of AES-256.

Organization. Section 2 presents some preliminaries. In Section 3, we present an efficient circuit to solve the AES S-Box differential equation. In Section 4, we discuss some families of quantum attacks on AES, and the various limitations they encounter in a quantum setting. In Section 5, we propose a framework to present quantum and classical structured search. In Section 6, we propose the first quantum DS-meet-in-the middle attack on 8

[2] In an independent and simultaneous work [HS18], DS-MITM attacks were analyzed in the particular case of Feistel networks. The generic speed-up provided in that paper is significant when using big amounts of qubits.

rounds of AES-256. In Section 7, we show how some ideas we had in Section 6 can be used to improve some of the best classical attacks. We conclude in Section 8.

2 Preliminaries

In this section we provide a description of our quantum adversary and computation models, a brief description of AES, and a summary of the best known classical attacks on reduced-round versions of AES in the single-key setting.

2.1 Quantum Adversary and Computation Model

We consider two types of adversaries, with a terminology used in [KLLN16b, HS17].

Model Q1. The adversary is allowed to make quantum computations, but she can only query *classical* oracles (*e.g.*, secret-key encryption or decryption oracles).

Model Q2. The adversary has quantum superposition access to encryption or decryption oracles. This model has also been referred to as IND-qCPA in the literature [BZ13, ATTU16, GHS16], since it allows for "quantum chosen-plaintext" queries.

From an oracle point of view, there is a strict separation between both models. Although the first one seems, at first sight, to represent a *realistic* situation (in which an adversary attacks a *classical* primitive using quantum computing power), the second one has also received much insight for multiple reasons. It is non-trivial, it completely exhausts the power of a quantum adversary and subsumes all intermediate models, and several primitives classically proven secure have been shown broken in this setting [KM10, KM12, KLLN16a, Bon18, Kap14].

Regardless of her oracle access capacities, our adversary uses a universal quantum computer. The computations are described in the well-studied quantum circuit model [NC02]. The *time complexity*, which is of main interest to us, is the gate count of the circuit (we did not consider its depth). The quantum memory complexity is the number of qubits.

We do not exclude the usage of quantum RAM (qRAM), which is a special storage allowing for superposition queries of its content. However, our main goal is to design attacks which do not make use of qRAM, as its potential prohibitive costs have been underlined by several authors [GR04]. To date, it seems more relevant to avoid it, in order to model adversaries from a nearer future.

2.2 Description of AES

AES [DR99], designed by Daemen and Rijmen, is the current encryption standard, chosen by an open competition organized by the NIST in 2000. It is a Substitution-Permutation Network alternating between linear layers, non-linear layers and round key additions. It has three different key sizes: 128, 192 and 256, with different key schedules, and respectively 10, 12 and 14 rounds.

0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15

Figure 1: AES byte ordering [Jea16]

AES State and Round Function. The cipher encrypts blocks of 128 bits, split into 16 bytes organized in a square (Figure 1). The round function has four operations, **AddRoundKey** (ARK), which xors the round key with the current state, **SubBytes** (SB), which applies the AES S-Box to each byte, **ShiftRows** (SR), which shifts the i-th row by i bytes left and **MixColumns** (MC), which multiplies each column by the AES MDS matrix.

While the round function has a strong design (it ensures total diffusion after two rounds), the key schedule has been widely acknowledged as the weakest point of AES. In particular, the key-schedule relations can be used to speed up cryptanalysis of reduced-round AES-192 and 256.

Notations. We write x_i, y_i, z_i, w_i the successive AES states (see Figure 4) after applying the 4 round operations. We note $x[0, 1, 2]$ when selecting bytes from these states. We use the usual AES byte numbering. When we consider a pair, the states are denoted x_i, x'_i. We also note $\Delta x_i = x_i \oplus x'_i$. Furthermore, equalities such as $x_4[1, 2, 3] = x'_4[1, 2, 3]$ are to be understood byte per byte.

2.2.1 Summary of Classical Cryptanalysis on AES.

Table 1 provides a summary of the best known classical attacks on AES. We have included in this table the new significant trade-offs that we introduce in section 7 of this paper.

Table 1: Summary of classical cryptanalysis on AES in the single secret key setting. Time is given in equivalent trial encryptions and memory in 128-bit blocks. We omit generic attacks, including the ones that perform an intelligent exhaustive search on the key like [BKR11].

Version	Rounds	Data	Time	Memory	Technique	Reference
Any	6	2^{32}	2^{44}	2^{32}	Square	[FKL+00]
	7	2^{113}	$2^{113} + 2^{80}$	2^{80}	DS MITM	[DFJ13]
	7	2^{105}	$2^{105} + 2^{99}$	2^{90}	DS MITM	[DFJ13]
	7	2^{97}	2^{99}	2^{98}	DS MITM	[DFJ13]
	7	2^{113}	$2^{113} + 2^{84}$	2^{74}	DS MITM	Section 7
	7	2^{105}	$2^{105} + 2^{95}$	2^{81}	DS MITM	Section 7
	7	$2^{113.1}$	$2^{113.1} + 2^{105.1}$	$2^{74.1}$	ID	[BLNS18]
	7	2^{105}	$2^{106.88}$	2^{74}	ID	[BLNS18]
192	7	2^{34}	2^{155}	2^{32}	Square	[FKL+00]
	7	2^{99}	2^{99}	2^{96}	DS MITM	[DFJ13]
	8	2^{113}	2^{172}	2^{82}	DS MITM	[DFJ13]
	8	2^{107}	2^{172}	2^{96}	DS MITM	[DFJ13]
256	7	2^{99}	2^{98}	2^{96}	DS MITM	[DFJ13]
	7	2^{34}	2^{172}	2^{32}	Square	[FKL+00]
	8	2^{113}	2^{196}	2^{82}	DS MITM	[DFJ13]
	8	2^{107}	2^{196}	2^{96}	DS MITM	[DFJ13]
	9	2^{113+x}	$2^{210-x} + 2^{196+x}$	2^{210-x}	DS MITM	[DFJ13]
	9	$2^{113+x/2}$	$2^{210-x} + 2^{194+x}$	2^{194+x}	DS MITM	Section 7

3 Quantumly Exploiting the AES S-box Differential Property

In this section we present an efficient (in time and memory) way to solve the differential equation of the AES S-Box. This operation is classically neglected, as it can be solved with a $2^8 \times 2^8$ lookup table. To make such a table quantum-accessible would mean, however, to use a few kilobytes of qRAM, a component that might be extremely costly; we did not want to rely on it. This analysis is crucial for the attack in Section 6, which would not have beaten Grover otherwise.

When counting quantum gates, we rely on the Clifford+T family, as in [GLRS16].

Lemma 1 (Solving quadratic equations in characteristic 2)**.** *There exists a quantum unitary* `QUAD`*, that, given $d^{-1} \in \mathbb{F}_{2^8}^*$, outputs a solution of the equation $x^2 \oplus x \oplus d = 0$ in the field $\mathbb{F}_{2^8} = \mathbb{F}_2[X]/(X^8 + X^4 + X^3 + X + 1)$ and a flag indicating if such a solution exists, using 7312 gates and 7 ancilla qubits. If we only need to know if there is a solution, then the cost is reduced by 448 gates.*

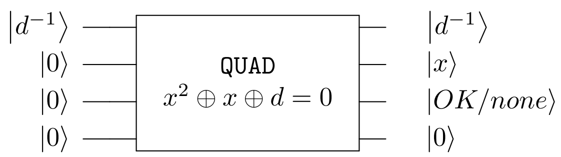

Figure 2: Quantum circuit that computes a solution of the equation $x^2 \oplus x \oplus d = 0$

Proof. In order to construct a solution, we can precompute the 127 d that accept a solution (as $d = 0$ cannot occur) and their corresponding root $R(d)$, check if the corresponding d matches, and write the corresponding solution in that case. As we check against precomputed values, we can do the check against d^{-1} instead of d, which allows us to avoid doing an inversion.

The circuit will sequentially test the value in the input register against the possible d^{-1}. For this, it will negate it, xor a fixed value, and compute the `and` of all its bits, using 7 Toffoli (and 7 ancilla qubits). The first ancilla qubit contains $d_7 \wedge d_6$, the second one $d_7 \wedge d_6 \wedge d_5$, and so on. This will be enforced at each step. The last one contains the `and` of all the bits in register d. If we xor a value to this register, it will be one if and only if the original value is equal to the xored one.

At each step, it will xor a value to the d register (which is done with NOT gates) and recompute the `and` from the first affected bit. It is to be noted that the first change can be computed with a CNOT gate (as $a \wedge \neg b = (a \wedge b) \oplus a$), while the other ones needs two Toffoli (one to uncompute the preceding computation, one to compute). We then have a bit that checks for equality in our circuit. We CNOT it to an output qubit (that will carry the OK/none information), and do a control-write of the solution associated to the given d to an external register. This can be done with one CNOT per bit at 1 in the solution, as they are precomputed.

We choose the order corresponding to sorting the possible values of d^{-1} in increasing order. The sequence is presented in Table 4 of Supplementary Material B. As two consecutive values are close, the total cost is reduced. In order to compute consecutively all the values for d^{-1}, we need 273 NOT. To check for equality, we need 127 CNOT plus 408 Toffoli (this may be lowered by using another ordering, but we did not investigate further). The writing of the solution needs 448 CNOT. The OK qubit can be updated with one CNOT per step. We also need 14 Toffoli for the initialization and finalization of the equality testing, plus 7 NOT to initialize the first value to be tested. As we do a sequential test, and as the last value we test is 0xff, the input value will be restored at the end. The total number of gates is then $7 + 273 = 280$ NOT, $127 + 127 + 448 = 702$ CNOT and $14 + 408 = 422$ Toffoli. As one Toffoli costs 15 (Clifford+T) gates, the total cost is then of 7312 gates.

As we only write x when we find a match, it will be 0 if there is no solution. If we only need existence check and not an explicit solution, we can reduce the cost by the 448 gates that write the solution. □

Remark 1. The sequential test uses $a \wedge \neg b = (a \wedge b) \oplus a$ to compute one $\wedge$ using one CNOT instead of two Toffoli gates. This saves $127(2 \times 15 - 1) = 3683$ gates overall, which is more than half the number of gates in this circuit.

Remark 2. For each d, we have chosen the even solution of the equation. Hence, we only need 7 qubits to output the solution.

Lemma 2 (S-Box differential property)**.** *Given Δ_x and Δ_y such that $\Delta_x\Delta_y \neq 0$, there exists either zero, two or four pairs x, y, x', y' such that $S(x) = y, S(x') = y'$, $x \oplus x' = \Delta_x$, $y \oplus y' = \Delta_y$.*

There exists a quantum unitary `SBDiff` *that, given such Δ_x and Δ_y, finds a solution x if it exists and output (x, OK) in this case, and outputs (0, none) otherwise. The time complexity of* `SBDiff` *is around 2 S-Box computations and it uses 22 ancilla qubits. If we only want to know if a solution exists and not an explicit solution, the cost drops to 1 S-Box computation and 15 ancilla qubits.*

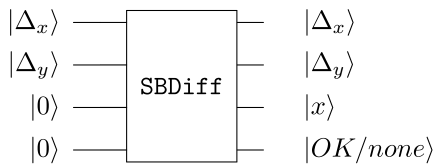

Figure 3: Circuit that computes a solution of a differential equation on the AES S-Box.

Proof. The AES S-Box can be written $S(x) = L(x^{-1})$, with L a linear operation and x^{-1} the inversion in $\mathbb{F}_{2^8}$ seen as $\mathbb{F}_2[X]/(X^8 + X^4 + X^3 + X + 1)$ (where 0 is mapped to 0). The main cost of the function is the inversion, which costs around 8 multiplications in the finite field [GLRS16]. Using the same source, we consider that a multiplication costs 981 gates with the multiplier in [CMMP08], and L costs 30 gates.

We want to solve the equation

$$S(x) \oplus S(x \oplus \Delta_x) = \Delta_y. \tag{1}$$

It can be rewritten as $x^{-1} \oplus (x \oplus \Delta_x)^{-1} = L^{-1}(\Delta_y)$. We note $L^{-1}(\Delta_y)$ as Δ'_y. There are two cases here. If $\Delta_x\Delta'_y = 1$, then 0 and Δ_x are solutions. It corresponds to the case where the differential equation has 4 solutions.

For every other x, we can multiply the equation by $x(x \oplus \Delta_x)$, and it becomes

$$\Delta'_y x^2 \oplus \Delta_x\Delta'_y x \oplus \Delta_x = 0. \tag{2}$$

To solve this quadratic equation, as we are in characteristic 2, we put it in the canonical form

$$(x/\Delta_x)^2 \oplus (x/\Delta_x) \oplus (\Delta_x\Delta'_y)^{-1} = 0. \tag{3}$$

We then only need to find a root $R(d)$ of the polynomial $X^2 + X + d$. The solutions will be $\Delta_x R(d)$ and $\Delta_x(R(d) + 1)$.

We do this using the preceding unitary, `QUAD`. The total cost is 7312 gates (or 6864 if we only need to know if a solution exists).

If we only want to know if a solution exists, the complete circuit is:

- Compute $\Delta_x L^{-1}(\Delta_y)$ (uses 8 ancilla qubits and costs 1011 gates).
- Check if a solution exists (6864 gates and 7 ancilla qubits) to the output.
- Uncompute $\Delta_x L^{-1}(\Delta_y)$ (costs 1011 gates).

The complete circuit for existence performs on 32 qubits : 16 inputs, 1 output, 15 ancilla, and costs 8886 gates, which is around 1 S-Box computation.

If we want an explicit solution, then the circuit is:

- Compute $\Delta_x L^{-1}(\Delta_y)$ (uses 8 ancilla qubits and costs 1011 gates).
- Check for an explicit solution (costs 7312 gates and uses 14 ancilla qubits).
- Compute Δ_x times the found solution (costs 861 gates) to the output.
- Uncompute the explicit solution (costs 7312 gates).
- Uncompute $\Delta_x L^{-1}(\Delta_y)$ (costs 1011 gates).

The complete circuit to get an explicit solution performs on 47 qubits : 16 inputs, 9 output, 22 ancilla, and costs 17507 gates (around 2 S-Box computations). □

Remark 3. The cost for the explicit solution can be reduced if we output $x\Delta_x^{-1}$ instead of the real solution x, as we would not need the uncomputation.

Remark 4. If we are in a case where 4 solutions exist, the routine will miss 2 solutions. This slightly reduces the success probability (as we fail to find 1 pair in 128), but allows to greatly simplify the generation of a superposition of solutions.

3.0.1 Other approaches.

We also considered different ways to solve this problem. We can test sequentially all the couples (Δ_x, Δ_y). There are 2^{15} of them, so we can estimate that it would cost 100 times more. We could also do a Grover Search on the equation. As it has 2 S-Boxes, it would cost around 2^5 times more.

3.0.2 Further applications.

This analysis can be used for any attack on AES that relies on the S-Box differential equation. Moreover, it can be generalized to other S-Boxes based on the inverse (such as the one in SM4 [OoSCCA]).

4 Discussion on Quantum Attacks on AES

To design quantum attacks on reduced-round versions of AES, it is natural to review design patterns that have been successful in the classical setting. We highlight in this section some of the most interesting families of attacks.

As exhaustive search can be accelerated by a square root using Grover's algorithm, classical attacks will become more expensive than generic ones in most of the cases. The quantized version of some classical attacks might benefit from smaller speed ups than the square root, and therefore, the quantum attack might also become worse than the generic one. For now, the only classical attacks that have been accelerated more than a square root are slide attacks [KLLN16a] using Simon's algorithm in the Q2 model.

In this section we discuss whether or not the best known attacks on AES can easily be quantized and give insights why. We also summarize the conclusions we have obtained after trying to quantize the best known attacks on AES, and explain why square attacks and DS-MITM attacks are the two most promising ones. Indeed, for some cases of square attacks and DS-MITM, we have managed to provide a competitive speed up. We provide the technical applied description in Section 6 and in the supplementary material A.

4.1 Quantum Exhaustive Search on AES

First of all, we focus on AES quantum resource estimates and Grover search. We derive from [GLRS16] quantum gate counts for AES components and reduced-round versions. Indeed, such precise counts are necessary in order to assess whether a key-recovery procedure of a given time is an *attack* or not.

4.1.1 Resource Estimates for Reduced-round AES.

Precise quantum resource estimates for the AES have been done in [GLRS16], with various technicalities to reduce the number of qubits. We adopt the same convention of using as universal set of gates the family Clifford+T, where The Clifford group is generated by the Hadamard gate H, the transform $S = |0\rangle\langle 0| + i|1\rangle\langle 1|$, and CNOT, and the T-gate is $T = |0\rangle\langle 0| + e^{i\pi/4}|1\rangle\langle 1|$. The reversible implementation of the AES S-Box in [GLRS16] costs 3584 T-gates, 4569 Clifford gates and 40 qubits.

We gather from these estimations that the main factor in time complexity (we count here the total number of Clifford+T gates applied) is the number of S-Boxes. Indeed, unless we use a qRAM to hold a lookup table for the AES S-Box , it needs to be computed on-the-fly. From now on, unless specified otherwise, our most precise counts will be in equivalent (reversible) S-Boxes.

We also extrapolate from [GLRS16] the costs of reversible implementations of reduced-rounds AES versions. In order to minimize the number of ancillary qubits used (which would otherwise be as high as 128 times the number of rounds), the authors uncompute rounds inside the AES black-box. We do the same for the reduced-round versions. Our benchmarks, in equivalent (reversible) S-Boxes, are summarized in Table 2.

Table 2: Cost benchmarks for quantum reversible AES components.

Component	Number of S-Boxes	Qubits
S-Box	1	40
128-bit full key expansion	40	320
192-bit full key expansion	32	256
256-bit full key expansion	52	664
6-round AES	144	408
7-round AES	160	536
8-round AES	192	536
6-round AES-128 (with key schedule)	168	856
8-round AES-256 (with key schedule)	224	1200

4.1.2 Resource Estimates for Grover Search

In order to compare our attacks to Grover search, we count the precise number of S-Box computations performed. It depends on the AES reversible implementation (which in turn depends on the number of rounds attacked), and on the optimal number of plaintexts to take to retrieve the good key with high probability. Let us take an example.

Lemma 3 (Grover Search on 8-round AES-256)**.** *Using Grover search and three classical queries to a secret-key 8-round AES-256 oracle, the key can be recovered in approximately* $\left\lceil \frac{\pi}{4} 2^{128} \right\rceil \times 224 \times 6 = 2^{138.04}$ *reversible S-Boxes, using approx.* 1500 *qubits.*

Proof. The search space (all possible keys) is of size 2^{256} and there is only one solution expected. Notice that with only two plaintexts, we have a wrong key that passes the test with probability $1 - \left(1 - \frac{1}{2^{256}}\right)^{2^{256}-1} \simeq 1 - \frac{1}{e}$. A second factor 2 stems from the necessity to uncompute the AES black-box inside Grover's iterations. This number of qubits comes from Table 2. □

We keep at hand the value $2^{138.04}$ S-Boxes for 8-round AES-256, as it will be needed below for precise comparisons. Remark that, from Table 2, the AES black-box oracle for other variants of Grover search can be safely assume to cost between 2^7 and 2^8 S-Boxes.

4.1.3 Bicliques.

The exhaustive search attack with bicliques [BKR11] is not a good candidate for a better speed up than the classical generic attack, as each candidate key is associated to an internal state that has to be stored in memory, and would be difficult to accelerate when applying Grover.

4.2 Quantum Impossible Differential Attacks

Impossible differential attacks use the fact that some events cannot occur in the cipher (for example, a differential transition that implies an impossibility). This provides a distinguisher for several middle rounds, which is extended a few rounds backwards and forwards, involving some key bits in the path. For good key guesses, the event will not occur by definition, and for bad guesses, it *can* occur. Hence, the attacker sieves the key space by removing the wrong key guesses, and the right one will be the only one left. For the interested reader, a generalization and improvements of impossible differentials applied to SPN networks can be found in [BLNS18].

The best impossible differential attack on AES-128 targets 7 rounds (Table 1), and provides a comparable trade off, with some advantages, to the best meet-in-the-middle attacks [BLNS18].

The most efficient way of building impossible differential attacks is to first obtain a set of pairs that might lead to the impossible middle differential, and next discard the possible keys associated to each pair in a quite efficient way, thanks to the early abort technique. The good key will be among the ones that have not been discarded.

We took several attempts at quantizing these attacks. To date, none seemed better than the existing ones on AES.

Computing the pairs turns out to be already difficult to optimize using quantum computations: we make heavy use of classical memory here, storing a whole structure in order to get efficiently the pairs which collide on the expected bytes in output. Although quantum collision search for random functions is known to be faster than classical collision search, using qRAM [BHT98] as well as without [CNS17], we emphasize that the problem here is too structured, much closer to *element distinctness*, for which a faster quantum algorithm is known, but only using an exponential amount of quantum memory [Amb07].

Using this memory-heavy method, it is possible to reduce the data complexity in the Q2 model (not in the Q1), and to speedup the computation of the pairs, but less than quadratically.

The sieving phase can be accelerated: given a key guess, we can perform a quantum search on the pairs that satisfy the impossible differential. This can be used as a test for the existence of such a pair for the outer search. Other classical improvements (like state-test techniques or multiple differentials as in [BNS14]) would need specific quantum implementations.

4.3 Quantum Square Attacks

The square attack has been proposed in [DKR97], and studied in the original specification document AES [DR99] targeting 6 rounds. It has been extended to 7 rounds for AES-192 and 256 [FKL+00]. It uses an integral distinguisher on 3-round AES, that needs 256 chosen plaintexts (if a byte takes all of its possible 2^8 values while the others remain constant, three rounds later all the bytes of the internal state will be balanced). It is extended by adding some rounds before and after it, at a cost of an increased data complexity (2^{32} chosen plaintexts) and some guesses of key bytes. This attack family performs classically worse than the best known meet-in-the-middle attacks [DFJ13], but is nevertheless interesting as it provides low complexities. Low data attacks (for instance when compared to DS-MITM) are of independent interest, as shown by new trends like [BDD+12].

This attack is already a *quantum attack* for 6-round AES-128, in the sense that it costs less time than Grover's exhaustive search (approximately 2^{64} encryptions).

We were able to propose quantized versions of the square attack, detailed in supplementary material A. They run in the Q1 model, in which using the partial sums technique from [FKL+00] is essential; we did not find a way to do it without relying on qRAM. This is the main limitation on the results (see Table 3).

Table 3: Quantum Q1 square attacks on reduced-round AES. Quantum time is given in reversible S-Boxes. Memory is counted in 128-bit registers.

Version	Classical counterpart	Queries	Quantum time	Quantum memory	Classical memory	Grover on keys
6-rd. AES-128	[FKL+00]	2^{35}	2^{44}	2^{25}	2^{36}	$2^{72.2}$
7-rd. AES-256	[DKR97]	2^{37}	2^{121}	negligible	2^{38}	$2^{137.3}$
7-rd. AES-256	[FKL+00]	2^{37}	2^{107}	2^{27}	2^{38}	$2^{137.3}$
7-rd. AES-192	[FKL+00]	2^{37}	$2^{103.4}$	2^{27}	2^{38}	$2^{105.6}$

4.4 Quantum DS-MITM

The DS-Meet-in-the-Middle attack was introduced in [DS08] to analyze AES. Many improvements have been proposed since. The most efficient ones on reduced-round AES are described in [DFJ13].

This attack uses also a distinguisher in the middle rounds. In this case, the distinguisher considers a (small) set of possible inputs to the middle rounds, such that, if one of the inputs follows a certain differential path, the set of possible associated values for a part of the state in the output will have a limited number of possibilities (much smaller than in a random case). This distinguisher can also be extended some rounds backward and forward involving some secret key bits. Previously proposed attacks are always built the following way: first, all the possible sets of inputs-outputs for the middle rounds distinguisher are computed and stored. Next, in an online phase, pairs of inputs are queried. The candidates following the differential path are kept, and for each, an exhaustive search on the involved key bits is done, computing the corresponding middle set. Next we check if these values are stored in the precomputed table. When this is the case, we have found a candidate for the secret key-bits.

The memory needs when storing all the possibilities for the middle-rounds property were the main bottleneck of these attacks. In Section 7 we propose to reorder the steps, which allows in some cases to reduce the memory needs. This improvement is directly inspired by our quantum attack from Section 6. The complexities of the best DS-MITM attacks on AES (our new ones provide some of the best interesting trade-offs and might be considered the best attacks in some cases) are detailed in Table 1.

The classical DS-MITM attack on 7-round AES is already a "quantum attack" on AES-256, since its time complexity is below that of Grover search (approximately 2^{128} encryptions). For AES-192, as soon as we consider precise implementation costs, the gap between the quantum gate cost of a reversible AES implementation and an optimized classical AES implementation makes the classical attack competitive against Grover.

There was a trivial open question then to see if quantum DS-MITM could reach more than 7 AES rounds. We answer this question in Section 6 by proposing a quantum attack on 8-round AES-256. In order to make the attack work, we have to counter-intuitively invert the order of the steps.

4.5 Simon-based Attacks on AES

Various efficient quantum attacks using Simon's algorithm have been proposed [LM17, KLLN16b, KM12]. One might then want to apply a quantum slide attack or a variant of the Grover-meets-Simon attack on AES. After studying these approaches, both seem to be inapplicable, due to the presence of a good key-schedule with round constants. Quantum slide attacks have similar constraints as the classical ones, and AES seems immune to them. Grover-meets-Simon would require an exhaustive search on $n-2$ rounds, which is essentially the cost of a complete exhaustive search. As a consequence, AES does not seem vulnerable to attacks benefiting from exponential accelerations in the Q2 model.

5 A General Framework for Quantum Structured Search

The attack procedures that we intend to develop below explore a search space (typically some subkey byte guesses) and find elements satisfying certain conditions, using possibly a nested search (typically on some state byte guesses). Grover's algorithm and its generalization, Amplitude Amplification [BHMT02] (AA in what follows) are well-known to be the quantum analogues of classical exhaustive search. In this section, we describe classical nested searches with natural quantum counterparts. The quantum attacks on AES that we describe in this paper (Square and DS-MITM) arise from this framework, although it does not help in quantizing most classical attacks (*e.g.* impossible differentials). First, let us review some cornerstones of our classical-to-quantum correspondence. In Section 6 we will describe our 8-round attack under this framework. The quantum square attacks are described in additional material A as an illustrative example.

Exhaustive Search. Consider the situation of looking among a search space S of size N for some element x satisfying the predicate P. Assume that testing P costs classically $ct(P)$ and quantumly $qt(P)$. Assume that $Pr(P(x)|x \in S) = p$; furthermore, enumerating the elements of S costs $O(1)$ (or negligible time anyway). Finding x with exhaustive search costs a classical time, denoted cc:

$$cc(S, P) = \frac{1}{p} ct(P) \ .$$

Indeed, we are producing elements of S and testing P until we find a "good one".

Denote $c = \pi/4$. The quantum equivalent of this search is Grover's algorithm. It requires $\frac{c}{\sqrt{p}}$ iterations, each of which builds the superposition of elements in S, $\sum_{y \in S} |y\rangle$ (costing negligible time again), and tests P in superposition.

Remark 5 (Memory usage). If the test for P requires access to a memory index which is computed and known only at runtime, *i.e.* RAM access, then its superposition version needs qRAM accesses. Quantum random-access gates are capable of querying not only a memory index, but a quantum superposition of such indexes (returning the same superposition of results). One may choose to forbid qRAM and use instead "plain" quantum circuits, without these qRAM gates. In that case, the classical test for P must not use any classical RAM. Memory accesses, if needed, must be known beforehand and not depend on the input. As a consequence, memory cannot be sorted in place. Testing if $x \in L$ for some list L and element x requires to go through the whole list L.

The quantum time complexity, denoted qc, is:

$$qc(S, P) = \frac{c}{\sqrt{p}} qt(P) \ .$$

Lazy Boolean And Test. Suppose that we are now interested for a more complex predicate $P_1 \wedge P_2$ over the same set S. We can evaluate this condition lazily. Suppose for example that we first evaluate P_1. A "good element" is found with probability p_1. Then, given $x \in S_{|P_1} = \{y \in S, P_1(y)\}$, we test if it satisfies P_2 in time $ct(P_2)$, this happens with probability p_2. The classical time is:

$$cc(S, P_1 \wedge P_2) = \frac{1}{p_2}(cc(S, P_1) + ct(P_2))$$

where $cc(S, P_1) = \frac{1}{p_1} ct(P_1)$ is the time to obtain one element of $S_{|P_1}$. Quantumly, this translates to an Amplitude Amplification procedure. The AA search space is $S_{|P_1}$. Each iteration requires to build its superposition twice (computations and uncomputations), in quantum time $qc(S, P_1) = \frac{c}{\sqrt{p_1}} qt(P_1)$, and to perform the quantum test for P_2, in time $qt(P_2)$. The total time is:

$$qc(S, P_1 \wedge P_2) = qs(S_{|P_1}, P_2)(2qc(S, P_1) + qt(P_2)) \ .$$

Product Space. Suppose that we need to iterate over pairs $x, y \in S_1 \times S_2$ satisfying a product predicate $P_1(x) \wedge P_2(y)$, and find a pair x, y in $S_{1|P_1} \times S_{2|P_2}$ such that $P(x, y)$ holds (P is yet another predicate). We assume that all the tests run in $O(1)$ time, that S_1, S_2, $S_{1|P_1}$, $S_{2|P_2}$ have respective cardinality N_1, N_2, M_1, M_2 and that there is only one "good" pair for P. A simple classical strategy would be to first write down the whole set $S_{1|P_1}$, then the whole $S_{2|P_2}$, then to test all pairs against P in total time $N_1 + N_2 + M_1 M_2$. This strategy finds a quantum acceleration, but it is likely suboptimal: finding all good elements for P_1 would cost $\min(\sqrt{N_1 M_1}, N_1)$, since we have no choice but to run Grover M_1 times. The third step, searching through $S_{1|P_1} \times S_{2|P_2}$, would only be performed using qRAM accesses to the stored elements.

Instead, we look at the classical streamed variants, without memory usage. One can first filter on S_1, then on S_2. Once we have found a good element for P_1, we exhaust all elements for P_2; we do this M_1 times. This costs $M_1(\frac{N_1}{M_1} + N_2) = N_1 + M_1 N_2$. The corresponding quantum procedure consists in an AA where the search space is $S_{1|P_1}$, and given x, the test runs a search through S_2 for a corresponding y such that $P_2(y)$ and $P(x, y)$. The quantum time is approx. $\sqrt{M_1}(\sqrt{\frac{N_1}{M_1}} + \sqrt{N_2}) = \sqrt{N_1} + \sqrt{M_1 N_2}$.

5.1 Filters

We now describe in more generality our *structured exhaustive search* framework, motivated by the rewriting of quantum attacks on AES.

Definition 1 (Filter)**.** A FILTER acts on a set S and uses the evaluation of a predicate P to produce the subset $S_{|P} = \{x \in S | P(x)\}$.

Filters produce a "solution space" $S_{|P}$ but they do not store it. Instead, one may consider them as iterators over this solution space. In our applications, the outermost filter will be expected to produce only one solution: the result of the search. Furthermore, the computations should not depend on the order of the elements returned by the filter (which is why we speak of *sets*). This is required to guarantee the correspondence with quantum searches, which do not return a particular element, but instead the uniform superposition over all solutions.

Intuitively, the composition of FILTERs (nesting exhaustive searches) is analogue to that of quantum AA procedures. However, for practical applications, error handling in AA procedures must be estimated precisely. This will be the subject of the next subsection.

Classical to Quantum Time Complexity. Once we have analyzed classically the complexity of a FILTER, it is easy to estimate the quantum time complexity of the corresponding AA. The number of iterations of the FILTER and each of its subfilters are replaced by their square roots. Computations internal to the filter, which correspond to computations internal to the AA test, may not benefit from such a speedup. For example, suppose that we search among N subkey guesses for the good one, with a test running in time t. Classically, the FILTER iterates N times, repeating the test, for a classical time t. Quantumly, the corresponding Grover search performs $\sqrt{N}$ iterations, repeating the test, for a quantum time $\sqrt{N}t$.

Remark 6 (Memory). The filters themselves do not use RAM; only the non-filter boolean tests may do so. By classical RAM access, we mean access to a memory location *computed at runtime*. In the quantum equivalent of a FILTER, the boolean test is performed for all elements in superposition: a RAM access needs to be replaced by an access to a superposition of memory locations: quantum RAM. Since the non-filter computations are simply lifted to their quantum, reversible equivalent, the memory usage is also the same, although lifted to qRAM.

5.2 Precision in quantum search

There are some additional constraints in quantum searches with respect to classical ones. In this section, we study different problems and cost overheads that can arise.

Exact search. The standard quantum search does not produce an exact superposition, but a state very close to it. It can be made exact by changing the last iteration of the search, and performing some rotations instead of a phase shift, as in [BHMT02]. In practice, we cannot expect to make these rotations perfectly, but we can approximate them efficiently, using standard methods [KSV02]. As we only need to perform this step once at the last iteration, we neglect this overhead here, and consider that if the input and output space size are classically known, then the search is exact. If it is only known at runtime, we can still use this method, but as we may perform a different operation for each possible input and output size, it may add a non-negligible cost.

If the searches cannot be made exact, the following lemmas allow us to estimate the noise and the overhead to bound it.

Lemma 4 (Grover Noise Amplification)**.** *A Grover search with 2^n iterations that uses an inner test function with a precision in amplitude of $1 - 2^{-e}$ will produce the expected state with a precision in amplitude of $1 - 2^{n-e}$.*

Proof. We consider here a sequential quantum computation, which consists in applying the operators $O_1, O_2, \dots O_m$ on the state $|s_0\rangle$, with $O_i|s_{i-1}\rangle = |s_i\rangle$. The final result will be $|s_m\rangle$. For example, this can be a perfect quantum search, with $|s_0\rangle$ the superposition of all the possible values, and $|s_m\rangle$ the superposition of selected values.

Now, we consider a sequence $O'_1, O'_2, \dots O'_m$, such that $O'_i|s_{i-1}\rangle = |s_i\rangle + |n_i\rangle$. $|n_i\rangle$ is an unknown noise component, with $||n_i\rangle| \leq 2^{-e}$.

The sequence will produce the state $|s_m\rangle + \sum_{i=0}^{m} O'_m \dots O'_{i+1}|n_i\rangle$. The amplitude of $|s_m\rangle$ will then be greater than $1 - \sum_{i=1}^{m} ||n_i\rangle| \geq 1 - m2^{-e}$. □

Lemma 5 (Grover Variation Noise)**.** *If the ratio between the input size N and the output size T of a Grover search lies in $\left[\frac{T_0}{N_0}(1-\epsilon), \frac{T_0}{N_0}(1+\epsilon)\right]$, then it will have a precision in amplitude greater than $1 - \epsilon$ after $\left\lceil \frac{\pi}{4}\sqrt{\frac{N_0}{T_0}} \right\rceil$ iterations.*

Proof. A Grover search is an iterated rotation in the plane spanned by the superposition of the N elements of the input and the superposition of the T elements of the output. An iteration makes an angle of $2 \arcsin \sqrt{\frac{T}{N}} \simeq 2\sqrt{\frac{T}{N}}$. If we do an exact Grover search with the parameters N_0 and T_0, it will result in a rotation angle θ in the range $\frac{\pi}{2}\sqrt{1 \pm \epsilon}$. The cosine of θ is the noise amplitude. We have $|\theta - \frac{\pi}{2}| \leq \frac{\pi}{2}(\sqrt{1+\epsilon} - 1) \leq \frac{\pi}{4}\epsilon$, hence the lemma holds. □

Lemma 6 (Grover Noise Reduction)**.** *If we repeat k times a Grover search with a precision in amplitude in $1 - 2^{-e}$ and a perfect test function, we can obtain the same Grover search, with a precision in amplitude in $1 - 2^{-ke}$.*

Proof. The principle is to perform k independent Grover searches, and then copy the first correct output. We have the state $\bigotimes_{i=1}^{k} (|T\rangle + |n\rangle)$, with $||n\rangle| < 2^e$. By copying the first correct output, we will obtain a noise term only if in the superposition, all the terms were noisy. Hence, the amplitude of the resulting noise will be the product of each noise amplitude, 2^{-ne}, if the test function of the Grover search is perfect. □

Remark 7. If the test function is imperfect, it will add an additional noise, but due to Lemma 4, it will generally be negligible.

6 Quantum DS-MITM Attack on 8-round AES-256

In this section we propose a Quantum DS-MITM attack on 8-rounds of AES-256 whose quantum time complexity is below Grover's algorithm. The attack is summarized in figure 4 and Algorithm 1. It reaches the highest number of rounds in the quantum setting.

6.1 Classical DS-MITM Attack

The DS-MITM attack is based on a middle rounds property which can be formulated as follows.

Lemma 7 (4-round property [DFJ13, Proposition 2])**.** *Suppose that we are given a plaintext-ciphertext pair active in one byte before and after 4 AES rounds. If we make the difference in input take all 2^8 values (δ-sequence) and collect the multiset of output differences in output, there are only 2^{80} (10 byte-conditions) possibilities.*

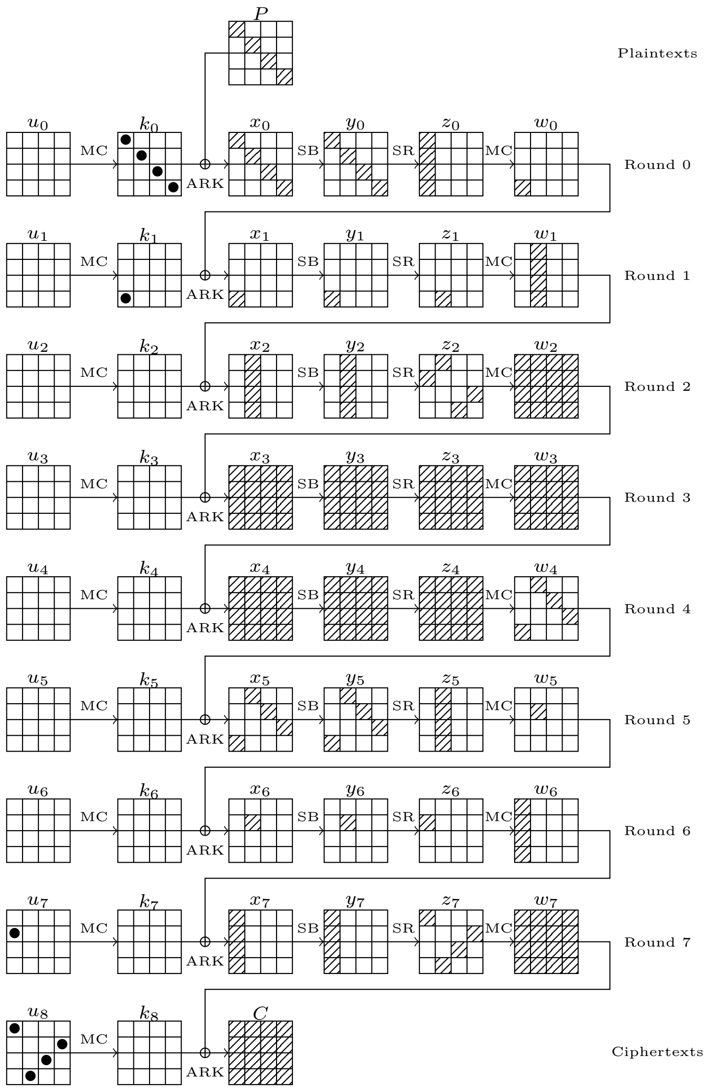

Figure 4: Full differential path used in the quantum attack. Key bytes guessed in the outer Grover procedure are denoted by •.

Proof. The proof uses rebound arguments and the fact that knowing input and output differences of the subbytes operations constrains the states by the AES S-Box differential equation. With only 10 byte-degrees of freedom, we can go through all sequences of states for x. Given a sequence of states, taking all differences in input, we can compute the unordered multiset of differences in output. □

Using this middle property, the attack works in three steps:

1. Find 2^{48} input-output pairs to the whole 7-round cipher satisfying the differential path;
2. Populate a table of all 2^{80} possible multisets;
3. Go through the 9 outer key bytes and recover them as follows: given a guess of these key bytes, there exists a corresponding input-output pair which satisfies the differential path. We can compute the expected multiset of values and check

whether it is in the table. If the answer is yes, the key guess is the right one. With overwhelming probability, we get the right key guess.

6.2 Attack Ideas

Adding a Round in the Middle. A classical DS-MITM attack on 8-round AES-256 and AES-192 already exists, but it adds the 8th round at the end of the cipher. Unfortunately, with this attack setting, it seems difficult to run Grover's algorithm and variants. Instead, we go to the idea of adding a round in the *middle* of the differential path, which is classically used in the 9-round attack on AES-256 [DFJ13].

Lemma 8 (5-round property [DFJ13, Section 4.2])**.** *Suppose that we are given a plaintext-ciphertext pair active in one byte before and after 5 AES rounds. If we make the difference in input take all* 2^8 *values (*δ*-sequence) and collect the multiset of output differences in output, there are only* $2^{80+8\times16}$ *(26 byte-conditions) possibilities.*

Proof. The property is the same as the 4-round one, but there is a whole 16-byte state in the middle that is unknown and must be added to the degrees of freedom. □

Unfortunately, there are two issues: first the table is too big to be constructed quickly enough. Second, even if we managed to somehow reduce its size, this would require massive amounts of qRAM, since we need to query the table in superposition. Hence, we *will not construct the table* and, instead, search on the fly if the multiset is a good one. The procedure becomes:

1. Find 2^{48} input-output pairs to the whole 8-round cipher satisfying the differential path (procedure is the same as the classical);
2. Grover some outer key bytes. The test function requires:
   - To find the right pair (this can be done by going through all possibilities, without any qRAM);
   - Given this pair, to compute the output multiset;
   - To perform a Grover search over all the possibilities in the middle (this search space is intricate and remains to be defined properly). If one of these possibilities yields the multiset, then the key guess test returns True (this is the right key guess). Otherwise it returns False.

Complexity Estimation. As there are 9 outer key bytes necessary and possibly 10+16 byte-degrees of freedom in the middle, we could already attain a complexity $2^{35\times4}$. Moreover, each inner possibility requires the computation of a multiset (2^8 partial encryptions). We overcome these issues:

- Since the input-output pair is known at the time of test, there are two less degrees of freedom (input and output difference are known).
- Key-schedule relations remove some degrees of freedom.
- The known key bytes enable us to replace multisets by *sequences* and reduce the number of partial encryptions from 2^8 to 2^5 (we cannot go below, as the probability of false positives would become significant).

All in all, we now expect the complexity to be little below exhaustive search. The same should go for a corresponding quantum search.

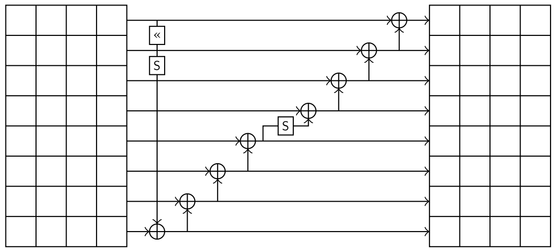

Figure 5: AES-256 key schedule [Jea16]

Key-schedule Properties. We prove some properties of the AES-256 key expansion.

Lemma 9 (Key-schedule Properties). *Let $k_0, \dots k_8$ be the 8-round expansion of the key-schedule of AES-256. The following relations hold:*

$$\begin{aligned} k_0[10] &= k_4[2] \oplus k_4[10] \\ k_0[15] &= k_4[7] \oplus k_4[15] \\ k_5[3] &= k_1[3] \oplus S(k_4[15]) \oplus S(k_4[11] \oplus k_4[15]) \end{aligned}$$

Proof. The AES-256 key schedule is represented on Figure 5, where the symbol « denotes shifting upwards the bytes of one column. We have:

$$k_2[0] = k_0[0] \oplus S(k_1[13]); k_2[4] = k_0[0] \oplus k_0[4] \oplus S(k_1[13]); \dots$$

and in particular, $k_0[10] = k_2[10] \oplus k_2[6]$ and $k_0[15] = k_2[11] + k_2[15]$.

Besides, the same relations hold between k_4 and k_2, so: $k_2[10] = k_4[10] \oplus k_4[6]$, $k_2[6] = k_4[6] \oplus k_4[2]$, $k_2[11] = k_4[7] \oplus k_4[11]$ and $k_2[15] = k_4[11] \oplus k_4[15]$, so:

$$k_0[10] = k_4[2] \oplus k_4[10] \text{ and } k_0[15] = k_4[7] \oplus k_4[15]$$

When i is odd, the first column of k_i is equal to the first column of k_{i-2}, to which we add the last column of k_{i-1}, going through S-Boxes. Hence:

$$k_5[3] = k_3[3] \oplus S(k_4[15]) \text{ and } k_3[3] = k_1[3] \oplus S(k_2[15]),$$

$$\text{so: } k_5[3] = k_1[3] \oplus S(k_4[15]) \oplus S(k_4[11] \oplus k_4[15]) \ .$$ □

6.3 Classical Description

In Algorithm 1, we describe our attack classically, in the FILTER framework. The remaining of this section is devoted to its details and complexity. In the quantum as in the classical setting, we do not authorize random access to memory. This means it cannot be *queried* with indexes known at runtime (although we can go through sequentially).

We count the running time in AES S-Box evaluations. If RAM (resp. qRAM) was authorized, evaluating the S-Box and solving the differential equation would be done using a lookup table. With this caveat, the running time comparisons between our attack and exhaustive search would still hold.

Classical exhaustive search of the key takes approximately $224 \times 2^{256} = 2^{263.8}$ S-Boxes. We prove that our procedure goes below that. We use the fact (Section 3) that solving the S-Box differential equation costs approximately 1 S-Box existentially (when we only test if there is a solution) and 2 S-Boxes to output the solution. For simplicity in the quantum operators involved, we suppose that all the differential equations have either zero or two solutions. As only 40 S-Box equations will be involved in the middle path, the probability that the "good" path encounters an S-Box equation with 4 solutions, and requires one of these, is $1 - \left(\frac{127}{128}\right)^{40} \simeq 27\%$.

Finding the Pairs. Our quantum 8-round AES-256 attack requires the same set of plaintext-ciphertext pairs as the 7-round classical one. These can be found using classical queries to the secret-key oracle and classical computations only. The difference in plaintexts is active only in a diagonal and the difference in round 7, before the last MixColumns operation, is active only in an antidiagonal.

Lemma 10 (Finding pairs [DFJ13, Section 4.1])**.** *There exists a classical procedure that, with 2^{113} encryption queries, returns 2^{48} plaintext-ciphertext pairs P, C, P', C' such that:*

- *The difference $\Delta P = P \oplus P'$ is active only in bytes $0, 5, 10, 15$,*
- *The difference $\mathrm{MC}^{-1}(\Delta C) = \mathrm{MC}^{-1}(C \oplus C')$ is active only in bytes $0, 7, 10, 13$.*

This procedure can remain the same classically and quantumly, since its running time is below what we expect of Grover's algorithm.

Lemma 11 (The Good Pair)**.** *Given the set of pairs of Lemma 10, given a guess for $k_0[0, 5, 10, 15]$ and $u_8[0, 7, 10, 13]$, there exists approximately one pair $(P, C), (P', C')$ that satisfies the full inner differential characteristic. Besides, there exists a quantum unitary that, on input $k_0[0, 5, 10, 15]$ and $u_8[0, 7, 10, 13]$, returns this pair. It runs in approximately 2^{53} S-Box computations.*

Proof. We test sequentially each of the 2^{48} possible pairs. There are 16 S-Box computations to do for each pair (4 in round 0 and round 8 for both members of the pair), to check if it is the good one, and to uncompute. □

Computing the δ-sequence. The classical attack uses multisets, we replace them by sequences. We fix a list of $\alpha = 2^5$ differences in $y_1[3]$. This list of differences is arbitrary and fixed throughout the attack.

The associated plaintexts are computed thanks to $k_1[3]$ and $k_0[0, 5, 10, 15]$. We encrypt these 2^5 plaintexts with our secret-key oracle. We partially decrypt the ciphertexts thanks to our guesses of u_8 and u_7 in order to obtain the sequence of differences in $x_6[5]$. This list contains 2^5 byte values, hence 256 qubits are sufficient to store it. This overhead is small w.r.t the amounts (thousands) needed to perform reversible AES encryption.

Lemma 12 (Computing the δ-sequence)**.** *There exists a quantum unitary that given the subkey guesses in k_0, k_1, u_7, u_8, and a pair satisfying the inner differential, computes the expected δ-sequence. It runs in 2^{13} S-Box computations.*

Proof. Each of the 2^5 calls needs 2^8 S-Boxes. The other computations are negligible. □

State Equations. The key-schedule relations that we use can be translated in the following equations on state bytes:

$$k_0[10] = x_4[2] \oplus x_4[10] \oplus \ell_2(y_3[0, 5, 10, 15]) \oplus \ell_2(y_3[8, 13, 2, 7]) \tag{4}$$

$$k_0[15] = x_4[7] \oplus x_4[15] \oplus \ell_3(y_3[3, 4, 9, 14]) \oplus \ell_3(y_3[1, 6, 11, 12]) \tag{5}$$

$$\begin{aligned} x_5[3] \oplus \ell_3(y_4[0, 5, 10, 15]) = k_1[3] \oplus S\Big(x_4[15] \oplus \ell_3(y_3[1, 6, 11, 12])\Big) \\ \oplus S\Big(x_4[15] \oplus \ell_3(y_3[1, 6, 11, 12]) \oplus x_4[11] \oplus \ell_3(y_3[8, 13, 2, 7])\Big) \end{aligned} \tag{6}$$

$$y_1[3] \oplus \ell(x_2[4\text{–}7]) = \ell(x_4[0\text{–}3] \oplus y_3[0, 5, 10, 15] \oplus x_4[4\text{–}7] \oplus y_3[3, 4, 9, 14]) \tag{7}$$

where ℓ_2 and ℓ_3 are the linear functions that, on input a column, give the third (resp. fourth) byte of this mixed column, and ℓ is the linear function which, on input a column, gives the third byte of this inverse-mixed column.

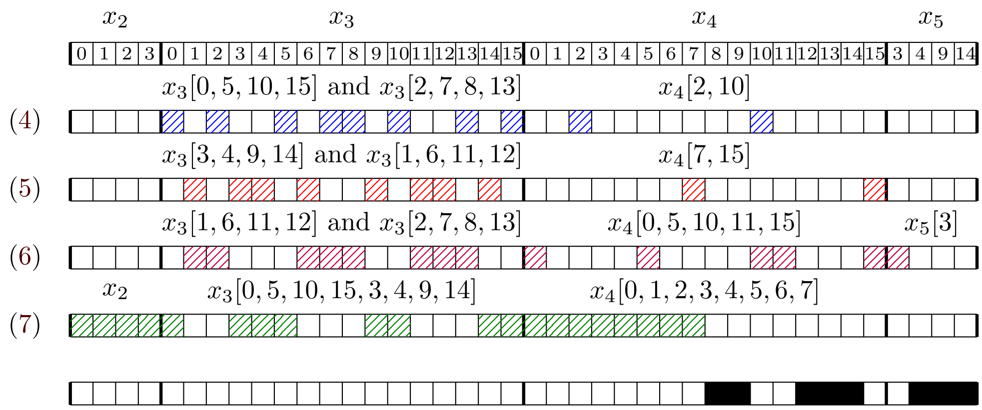

Figure 6: All key relations (4), (5), (6), (7) and the 8 remaining "free' bytes.

Sieving with the State Equations. At some point in our FILTER attack, we have two choices for each byte of $x_2[0–3]$ (one column of x_2), each byte of x_3, each of x_4 and each byte of $x_5[3,4,9,14]$. We then sieve these possible choices with the 4 key relations obtained above, translated into relations between the bytes of these states. As there are 40 bit-degrees of freedom and 4 byte constraints, we expect 2^8 possibilities to pass. These relations turn out to constrain completely 32 of the byte values and leave the 8 others free. This is represented in Figure 6, where we represent the bytes of x_2, x_3, x_4 and x_5 concerned. For each relation, (4) to (7), we represent the bytes of the states that appear in it. These values are always either mixed or passed through an S-Box , so we may consider the relations to be independent. In the end, 8 bytes appear in none of the relations: these are exactly the 8 free choices remaining.

Classical Complexities. There are three levels of filtering. We go from the outermost to the inner one:

1. Filtering on key byte guesses: there are $2^{10\times 8}$ guesses to look at, among which we expect exactly one solution. Given these byte guesses, we find a good pair and start computing the states in the middle. The cost is in total:

$$2^{80}\left(2^{53}+\mathbf{f}\right)$$

   where $\mathbf{f}$ is the next filter. The 2^{53} term is the cost to find the good pair.

2. Filtering on the $16+8$ differences: there are 40 S-Box differential equations to solve in total. In the middle, we can match between x_4 and y_4 column by column: this is more efficient than solving all 2^{32} equations at once. At this point, we have obtained 2^{40} possibilities for the full sequence of states, as each state byte has two possibilities. We then pass the key-conditions.

   (a) Equations 4 and 5 each need 2^{10} computations, without S-Boxes involved. This is negligible.

   (b) To check Equation 6 for all possibilities, one needs actually few S-Boxes. Indeed, due to the constrained choices, $x_4[15]\oplus\ell_3(y_3[1,6,11,12]$ can take on average only 4 values and $x_4[11]\oplus\ell_3(y_3[8,13,2,7])$ only 8. So in total we need not more than $4+8$ S-Boxes evaluations, which is negligible.

   (c) Again, to check Equation 7, we need only linear computations, without any S-Boxes.

Algorithm 1: Classical FILTER attack on 8-round AES-256

Result: The key bytes $k_0[0,5,10,15]$, $k_1[3]$, $u_7[1]$, $u_8[0,7,10,13]$

Compute 2^{48} plaintext-ciphertext pairs with input difference active in bytes $0,5,10,15$ and output difference active in (mixed) bytes $0,7,10,13$

Filter $k_0[0,5,10,15]$, $k_1[3]$, $u_7[1]$, $u_8[0,7,10,13]$ **such that:**

- Find a pair P, C, P', C' which satisfies the differential path `(`2^{53} `S-Boxes)`
- Compute the 2^5-sequence of differences $\delta w_5[5]$ by making $x_1[3]$ vary `(chosen-plaintext queries)`
- Compute $x_1[3], x'_1[3]$ and obtain $\Delta x_2[4–7]$
- Compute $x_6[5], x'_6[5]$ and obtain $\Delta y_5[3,4,9,14]$
- **Filter** $\Delta y_2[4–7]$, $\Delta x_5[3,4,9,14]$, Δx_4 **such that:**
  - If $\Delta y_2[4–7]$ and $\Delta x_2[4–7]$ do not match, **Abort** `(prob.` 2^{-4}`)`
  - If $\Delta x_5[3,4,9,14]$ and $\Delta y_5[3,4,9,14]$ do not match, **Abort** `(prob.` 2^{-4}`)`
  - Each time a S-Box equation is solved, it gives two possibilities for each byte: store them
  - From $\Delta y_2[4–7]$, compute Δx_3
  - From $\Delta x_5[3,4,9,14]$, compute Δy_4
  - Match Δx_4 against Δy_4 and Δx_3, column by column `(prob.` 2^{-8} `for each column)`. If they do not match, **Abort**
  - At this point, one guess over 2^{40} has passed the S-Box differential equations

  - Write equation 4 for all 2^{10} choices of $x_4[2], x_4[10], y_3[0,5,10,15,8,13,2,7]$: 4 of them are expected to pass. Store them.
  - Write equation 5 for all 2^{10} choices of $x_4[7], x_4[15], y_3[3,4,9,14,1,6,11,12]$: 4 of them are expected to pass. Store them.
  - For each of the 4×4 stored choices and 2^4 choices of $x_5[3], x_4[0,5,11]$, write equation 6: one of these 2^8 possibilities is expected to pass
  - The full state y_3 and the bytes $x_4[0,2,5,7,10,11,15], x_5[3]$ are now determined.
  - For these bytes and each 2^8 choices of $x_2[4–7], x_4[1,3,4,6]$, write equation 7: one choice is expected to pass
  - In the end, the full state x_3 is determined, alongside $x_5[3]$, $x_2[4–7]$ and $x_4[0–7,10,11,15]$. Bytes $x_4[8,9,12,13,14]$ and $x_5[4,9,14]$ remain free.
  - **Filter** *Choices for* $x_4[8,9,12,13,14]$ *and* $x_5[4,9,14]$ **such that:**
    - `(There are (a fixed number of)` 2^8 `possibilities to explore)` Since the whole sequence of states $x_2[4,5,6,7]$, x_3, x_4, $x_5[3,4,9,14]$ is known, compute the expected δ-sequence $\delta w_5[5]$ `(`$2^5 \times 40$ `S-Boxes)`
    - If it does not equal the expected sequence, **Abort**
  - If the filter failed, **Abort**
- If the filter failed, **Abort**

So the full cost of the filter is:

$$2^{(16+8)\times 8-40}\left(16+4\times(2^8\times 8)+\mathbf{f}'\right)$$

where 16 stems from the outer S-Box equations, $4\times(2^8\times 8)$ is the term for the inner by-columns equations, and **f**' is the next filter.

3. Filter on the state sequences: there are 8 remaining bit-degrees of freedom, that is, bytes that can take two values. For each possibility, we compute the δ-sequence using $2^5\times 40=1280$ S-Boxes and match against the expected one. We expect one or zero solution. The cost is $2^8\times 1280$.

In total we obtain $\mathbf{2^{80}}\left(2^{53}+\mathbf{2^{152}}\left(\left(4\times(\mathbf{2^8}\times 8)\right)+\mathbf{2^8}\times 1280\right)\right)$, where we highlight the terms on which we are going to take the square root, to obtain the corresponding quantum complexity.

A direct computation gives a classical complexity of $2^{250.3}$ S-Boxes, which is actually the optimal cost for our algorithm. Indeed, the differential path we use contains in total 30 byte-degrees of freedom (including key byte guesses), when discounting the key-schedule relations. The best expected complexity is then 2^{240} times the computation of a δ-sequence, which is exactly what we get (δ-sequences are the dominant term). We now turn ourselves towards the quantum time complexity of this procedure.

6.4 Quantum Complexity

By the correspondence of Section 5, the whole FILTER program that we wrote classically has a quantum equivalent in terms of nested Amplitude Amplification procedures. If we dismiss negligible factors and non-S-Box computations, we rewrite the classical complexity as:

$$\mathbf{2^{80}}\left(2^{53}+\mathbf{2^{152}}\left(\left(4\times(\mathbf{2^8}\times 8)\right)+\mathbf{2^8}\times 1280\right)\right)$$

The corresponding quantum time complexity should be as follows:

$$\left\lceil c2^{40}\right\rceil\left(2\left\lceil c2^{76}\right\rceil\left(2^7\left\lceil c2^4\right\rceil+2\left\lceil c2^4\right\rceil\times 2\times 1280\right)\right)$$

with $c=\frac{\pi}{4}$ the Grover constant and 2 factors added for uncomputations.

S-Box property. The differential property can yield 4 solutions, and if we only consider two solutions, as we go through 40 S-Boxes, we succeed with probability $\left(\frac{127}{128}\right)^{40}\simeq 2^{-0.45}$. Moreover, if we want to be more precise in the analysis, we can remark that the differential property is not fulfilled for half of the differentials, but for $\frac{127}{255}$ of the non-zero differentials. Moreover, we can restrict our search space over non-zero differentials.

Hence, we do not have 2^{192} differentials, but $2^{191.86}$, and the 40 differential equations filter $2^{40.23}$ values, for a total space size in the second filter of $2^{151.64}$.

Precision. This complexity will hold if the output space size of every Amplitude Amplification procedure is known in advance for the good guesses. Hence, we have to review each part, and estimate the deviation, if there is any. Once it is known, we can derive the precision requirements from the first quantum complexity estimate, and estimate the increase in complexity for each part.

1. Amplification of $k_0[0,5,10,15]$, $k_1[3]$, $u_7[1]$, $u_8[0,7,10,13]$: There is no filtering or nontrivial states to consider here, and the test function is expected to accept only the good key guesses.

2. Amplification of $\Delta y_2[4–7]$, $\Delta x_5[3,4,9,14]$, Δx_4: The filtering of $\Delta y_2[4–7]$ and $\Delta y_5[4,9,14]$ is done against Δx_2 and Δy_5, and there are exactly 127 solutions per byte. Hence, there is no variation here. However, for Δx_4, there is the actual meet in the middle, with a constraint from Δy_2 and a constraint from Δx_5, that can make the actual number of solutions vary.

3. Sequential test of the 4 key conditions: Here, the number of solutions can also vary. As we do a sequential test, we only have to care about the maximal number of solutions that will occur for the right guesses.

4. Last filter: there are exactly 2^8 tuples to iterate on. This has to be done once for each solution that might arise from the previous step. As we know the list of these solutions, we can also perform a quantum search on them.

For the variations of the differences, we have been able to simulate them column by column, and found that the number of solutions when fixing Δx_3 (deduced from Δy_2) and Δy_4 (deduced from Δx_5) were in 98% of the cases the interval $2^{27.95}(1 \pm 2^{-9})$. As we have 4 columns, we can estimate that in more than 90% cases we will have a number of solutions that varies of a factor less than 2^{-7} around the mean. We need to have an error smaller than 2^{-40}, hence, from Lemmas 4, 5 and 6, we have to do this amplification $40/7 < 6$ times. We choose to do it 7 times, which will produce an error in amplitude in around $2^{40-7\times 7} = 2^{-9}$, hence a probability of measuring the noise in around 2^{-18}.

The construction of the input states ($\Delta y_2[4–7]$, $\Delta x_5[3,4,9,14]$, Δx_4) has the same problem. This computation is amplified $2^{151.6/2}$ times, and we want the final error to be smaller than 2^{-8}, in order to not add too much additional noise to the previous search. This imposes to perform it at least $(152/2+8)/7 = 12$ times. As this step is not in the critical path, we choose to do it 16 times, which will ensure a negligible noise.

Regarding the key conditions, we consider that overall, the 4 equations for the good path (there is only one, corresponding to a good guess of all subkey bytes and state differences) might have 4 solutions. As the candidates (and their number) are known, we can generate the superposition of all of them, to be used in the final test. The overhead to generate this superposition is negligible, but this adds a factor 2 to the number of iterations. Moreover, the last Grover round, that gives us an arbitrary precision, depends on this number of solutions, so there are 4 different such rounds to perform. Hence, the final complexity, taking into account the success probability and the precision problems is

$$2^{0.45}\left\lceil c2^{40}\right\rceil\left(2\times 7\left\lceil c2^{75.8}\right\rceil\left(2^4 2^7\left\lceil c2^4\right\rceil + 2\left\lceil c2^{4+1}\right\rceil\times 2\times 1280\right)\right) = 2^{136.62} \text{ S-Boxes.}$$

The dominating term is the computation of δ-sequences. The cost is slightly below exhaustive search, at $2^{138.04}$ S-Boxes. If we did not consider the precision and number of solutions problems, the estimate would have been around $2^{132.3}$ S-Boxes.

6.4.1 Making the Attack Q1.

We presented the attack in the Q2 model, where superposition queries to the cipher black-box are allowed. Indeed, some of these queries (encrypting the δ-sequence) appear inside a FILTER, which turned to an Amplitude Amplification, requires to compute in superposition. Actually, it is possible to replace all the queries by classical ones.

Suppose given a guess of 9 key bytes in k_0, k_1, u_8 and a corresponding good pair. To produce the expected δ-sequence, there are encryption queries to perform, making the difference in y_1 vary. These chosen-plaintext queries depend on the 4 guessed bytes of k_0 and the guessed byte of k_1. But this represents only 2^{40} values. This means that the whole procedure needs only $2^{48}\times 2^{40} = 2^{88}$ queries, grouped by their corresponding pairs.

We now perform all these queries beforehand. In the outermost FILTER, instead of computing the δ-sequence on the fly, we go through the set of stored queries and find the ones that interest us (those which correspond to the current pair). This would be performed efficiently using RAM; without, we need an incompressible factor 2^{88} in time to go through the whole memory. These 2^{88} computations (comparisons only, not S-Boxes) are added once for each outer iteration. This term is not dominant, and does not change the attack complexity.

7 Improved Classical DS-MITM Using Quantum Ideas

While rewriting the DS-MITM attack in order to build an efficient quantum one, we realized that some of the ideas that we propose, although they might seem somewhat counter-intuitive, can also help improving the best known classical attacks on AES. In particular they allow sometimes to reduce the memory complexity, which was for a long time the bottleneck in this type of attacks.

In this section, we briefly show how to improve the DS-MITM 9-round AES-256, currently the best known 9-round AES attack, and to reduce the memory requirements of the DS-MITM 7-round AES attack.

While these improvements, due perhaps to their counter-intuitiveness, have never been used before, we believe that they would arguably provide the best known classical attack on AES-256 up to date, and open a new line of ideas for improving further the best classical complexities. As this was not the original scope of the paper we leave further improvements and analysis using this idea as an open problem, and provide here some ideas and examples.

Re-ordering the Steps. The main idea that helps reducing the memory needs is to first store the results from the previous online phase (key guesses, corresponding pairs and multisets computed from the queried messages), and next perform an exhaustive search over the middle values (what was before a precomputed table), and look for a collision. When the first term is smaller, the memory is reduced, while keeping similar complexities of data and time (as we are basically doing the same computations in a different order).

Further possible improvements. Using multiple differentials in the middle and storing the transitions will allow to provide attacks with reduced data, while partially increasing the previously reduced memory. We believe new interesting trade-offs might result from these combination.

7.1 New improved attack on 9-rounds AES-256

We consider the 9-round AES-256 attack of [DFJ13, Fig.6]. In this attack, after having obtained 2^{144} plaintext-ciphertext pairs verifying the input differential with 2^{113} queries, we sieve the outer key bytes. Each pair gives 2^{48} possible values for $k_{-1}[0, 5, 10, 15]$, k_8, u_7, that can be enumerated in time 2^{48}, such that it verifies the whole differential pattern. Then, for each of these values, we encrypt a δ-set and compare the associated multiset to the table of precomputed possibilities: as there are 26 byte parameters that determine the middle rounds, the precomputed table has size 2^{210}. This can be reduced by a factor 2^7, if we replay the attack 2^7 times (increasing the data and time complexity).

New attacks with reduced memory. By reordering the steps, we obtain a new attack based on this previous one that still needs 2^{113} data, 2^{210} in time and now only 2^{194} in memory. We can propose a trade-off with different factors as they did, by considering a factor of 2^x less states to try in the middle if we store 2^x times more possible pairs. For this we need a data complexity increased by a factor $2^{x/2}$ (that generates 2^x times more pairs), and a time complexity that will be the max between 2^{210-x} and 2^{194-x}. All in all, we are able to propose better trade offs: indeed, in order to reach a memory of 2^{194} with the attack from [DFJ13], they would need a time complexity of 2^{212} and a data complexity of $2^{124.5}$. A summary of these attacks is presented in Table 1.

7.2 New trade-offs on 7-rounds AES-128

In this case, the new trade-offs are not always interesting, in particular when compared to the best impossible differential attacks, but they improve upon previous DS-MITM attacks at least with respect to memory needs. The same way as before, we consider the DS-MITM 7-round AES-128 attack ([DFJ13, Fig.4]). When applying our improvements, we obtain for instance 2^{113} data, $2^{113} + 2^{84}$ time and 2^{74} memory. When considering the multiple differentials idea, we are able to reach 2^{105} data, $2^{105} + 2^{95}$ time and 2^{81} memory.

8 Conclusion

Among all the classically efficient single key-recovery attacks that we studied, we were able to obtain:

- A quantum square attack for 6-round AES and 7-round AES-192 and AES-256,
- A quantum DS-MITM attack on 8-round AES-256.

Although both the Q1 and the more powerful Q2 attacker setting were in the scope of our study, all the attacks presented in this paper adopted the Q1 setting, meaning that they only need *classical* queries.

The best known classical non-generic attacks target up to 9 rounds of AES-256. The quantum attacks that we obtained only reach 8 rounds. Hence, the security margin of AES-256 determined by our attacks is bigger in the post-quantum world.

Our results, however, encompasses only quantum key recovery attacks. A quantum adversary could take advantage of the relatively small internal state size of AES (128 bits), which, contrary to the key length, cannot be raised. Classical known results of cryptanalysis of popular modes of encryption, such as CBC [BL16] or CTR [LS18], prove that its 128-bit state does not offer 128-bit security when AES is used with these modes. This cryptanalysis can be quantumly improved as shown in [CNS17], so the necessity to combine AES with an improved secure mode of encryption is sharpened when taking into account quantum attacks.

References

[Amb07] Andris Ambainis. Quantum Walk Algorithm for Element Distinctness. *SIAM J. Comput.*, 37(1):210–239, 2007.

[ATTU16] Mayuresh Vivekanand Anand, Ehsan Ebrahimi Targhi, Gelo Noel Tabia, and Dominique Unruh. Post-quantum security of the CBC, CFB, OFB, CTR, and XTS modes of operation. In *PQCrypto*, volume 9606 of *Lecture Notes in Computer Science*, pages 44–63. Springer, 2016.

[BB18] Gustavo Banegas and Daniel J. Bernstein. Low-communication parallel quantum multi-target preimage search. In *Selected Areas in Cryptography - SAC 2017*, volume 10719 of *Lecture Notes in Computer Science*, pages 325–335. Springer, 2018.

[BDD+12] Charles Bouillaguet, Patrick Derbez, Orr Dunkelman, Pierre-Alain Fouque, Nathan Keller, and Vincent Rijmen. Low-data complexity attacks on AES. *IEEE Trans. Information Theory*, 58(11):7002–7017, 2012.

[BHMT02] Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002.

[BHT98] Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum cryptanalysis of hash and claw-free functions. In *LATIN*, volume 1380 of *Lecture Notes in Computer Science*, pages 163–169. Springer, 1998.

[BKR11] Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. Biclique cryptanalysis of the full AES. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology - ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 344–371. Springer, 2011.

[BL16] Karthikeyan Bhargavan and Gaëtan Leurent. On the practical (in-)security of 64-bit block ciphers: Collision attacks on HTTP over TLS and openvpn. In *ACM Conference on Computer and Communications Security*, pages 456–467. ACM, 2016.

[BLNS18] Christina Boura, Virginie Lallemand, María Naya-Plasencia, and Valentin Suder. Making the impossible possible. *J. Cryptology*, 31(1):101–133, 2018.

[BNS14] Christina Boura, María Naya-Plasencia, and Valentin Suder. Scrutinizing and improving impossible differential attacks: Applications to CLEFIA, Camellia, LBlock and Simon. In *Advances in Cryptology - ASIACRYPT 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 179–199. Springer, 2014.

[Bon18] Xavier Bonnetain. Quantum key-recovery on full AEZ. In *Selected Areas in Cryptography - SAC 2017*, volume 10719 of *Lecture Notes in Computer Science*, pages 394–406. Springer, 2018.

[BZ13] Dan Boneh and Mark Zhandry. Secure signatures and chosen ciphertext security in a quantum computing world. In *CRYPTO (2)*, volume 8043 of *Lecture Notes in Computer Science*, pages 361–379. Springer, 2013.

[CMMP08] Donny Cheung, Dmitri Maslov, Jimson Mathew, and Dhiraj K. Pradhan. Theory of quantum computation, communication, and cryptography. chapter On the Design and Optimization of a Quantum Polynomial-Time Attack on Elliptic Curve Cryptography, pages 96–104. Springer-Verlag, Berlin, Heidelberg, 2008.

[CNS17] André Chailloux, María Naya-Plasencia, and André Schrottenloher. An efficient quantum collision search algorithm and implications on symmetric cryptography. In *ASIACRYPT (2)*, volume 10625 of *Lecture Notes in Computer Science*, pages 211–240. Springer, 2017.

[DFJ13] Patrick Derbez, Pierre-Alain Fouque, and Jérémy Jean. Improved key recovery attacks on reduced-round AES in the single-key setting. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 371–387. Springer, 2013.

[DKR97] Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. The block cipher square. In *FSE*, volume 1267 of *Lecture Notes in Computer Science*, pages 149–165. Springer, 1997.

[DR99] Joan Daemen and Vincent Rijmen. AES proposal: Rijndael. 1999.

[DR12] Joan Daemen and Vincent Rijmen. On the related-key attacks against aes. *Proceedings of the Romanian Academy, Series A*, 13(4):395–400, 2012.

[DS08] Hüseyin Demirci and Ali Aydin Selçuk. A meet-in-the-middle attack on 8-round AES. In *FSE*, volume 5086 of *Lecture Notes in Computer Science*, pages 116–126. Springer, 2008.

[FKL+00] Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Michael Stay, David A. Wagner, and Doug Whiting. Improved cryptanalysis of Rijndael. In *FSE*, volume 1978 of *Lecture Notes in Computer Science*, pages 213–230. Springer, 2000.

[GHS16] Tommaso Gagliardoni, Andreas Hülsing, and Christian Schaffner. Semantic security and indistinguishability in the quantum world. In *Annual Cryptology Conference*, pages 60–89. Springer, 2016.

[GLRS16] Markus Grassl, Brandon Langenberg, Martin Roetteler, and Rainer Steinwandt. Applying Grover's Algorithm to AES: Quantum Resource Estimates. In *PQCrypto*, volume 9606 of *Lecture Notes in Computer Science*, pages 29–43. Springer, 2016.

[GR04] Lov K. Grover and Terry Rudolph. How significant are the known collision and element distinctness quantum algorithms? *Quantum Information & Computation*, 4(3):201–206, 2004.

[Gro96] Lov K. Grover. A Fast Quantum Mechanical Algorithm for Database Search. In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 212–219. ACM, 1996.

[HS17] Akinori Hosoyamada and Yu Sasaki. Quantum meet-in-the-middle attacks: Applications to generic feistel constructions. Cryptology ePrint Archive, Report 2017/1229, 2017.

[HS18] Akinori Hosoyamada and Yu Sasaki. Quantum Demirci-Selçuk Meet-in-the-Middle Attacks: Applications to 6-Round Generic Feistel Constructions. In *SCN*, volume 11035 of *Lecture Notes in Computer Science*, pages 386–403. Springer, 2018.

[HSX17] Akinori Hosoyamada, Yu Sasaki, and Keita Xagawa. Quantum multicollision-finding algorithm. In *ASIACRYPT (2)*, volume 10625 of *Lecture Notes in Computer Science*, pages 179–210. Springer, 2017.

[Jea16] Jérémy Jean. TikZ for Cryptographers. http://www.iacr.org/authors/tikz/, 2016.

[Kap14] Marc Kaplan. Quantum attacks against iterated block ciphers. *CoRR*, abs/1410.1434, 2014.

[KLLN16a] Marc Kaplan, Gaëtan Leurent, Anthony Leverrier, and María Naya-Plasencia. Breaking symmetric cryptosystems using quantum period finding. In *CRYPTO (2)*, volume 9815 of *Lecture Notes in Computer Science*, pages 207–237. Springer, 2016.

[KLLN16b] Marc Kaplan, Gaëtan Leurent, Anthony Leverrier, and María Naya-Plasencia. Quantum differential and linear cryptanalysis. *IACR Trans. Symmetric Cryptol.*, 2016(1):71–94, 2016.

[KM10] Hidenori Kuwakado and Masakatu Morii. Quantum distinguisher between the 3-round feistel cipher and the random permutation. In *IEEE International Symposium on Information Theory, ISIT 2010, Proceedings*, pages 2682–2685. IEEE, 2010.

[KM12] Hidenori Kuwakado and Masakatu Morii. Security on the quantum-type even-mansour cipher. In *Proceedings of the International Symposium on Information Theory and its Applications, ISITA 2012*, pages 312–316. IEEE, 2012.

[KSV02] A. Yu. Kitaev, A. H. Shen, and M. N. Vyalyi. *Classical and Quantum Computation*. American Mathematical Society, Boston, MA, USA, 2002.

[LM17] Gregor Leander and Alexander May. Grover Meets Simon - Quantumly Attacking the FX-construction. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017, Proceedings, Part II*, volume 10625 of *Lecture Notes in Computer Science*, pages 161–178. Springer, 2017.

[LS18] Gaëtan Leurent and Ferdinand Sibleyras. The missing difference problem, and its applications to counter mode encryption. In *EUROCRYPT (2)*, volume 10821 of *Lecture Notes in Computer Science*, pages 745–770. Springer, 2018.

[MBT+17] Kerry A. McKay, Larry Bassham, Meltem Snmez Turan, , and Nicky Mouha. Istir 8114 report on lightweight cryptography. In *Technical report, U.S. Department of Commerce, National Institute of Standards and Technology*, 2017.

[MMOS18] Daniel P. Martin, Ashley Montanaro, Elisabeth Oswald, and Daniel James Shepherd. Quantum key search with side channel advice. In *Selected Areas in Cryptography - SAC 2017, Revised Selected Papers*, volume 10719 of *Lecture Notes in Computer Science*, pages 407–422. Springer, 2018.

[NC02] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

[OoSCCA] P.R. China Office of State Commercial Cryptography Administration. The SMS4 block cipher (in chinese).

[RMYK17] Sandeep Rao, Dindayal Mahto, Dilip Yadav, and Danish Khan. The AES-256 cryptosystem resists quantum attacks. 8:404–408, 04 2017.

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 124–134. IEEE Computer Society, 1994.

Supplementary Material

A Quantum Square Attacks on the AES

In this section, we study the quantum variants of the Square attack. Our cost estimates are given in S-Boxes. Our results are summarized in Table 3, all in the Q1 model. For quantum exhaustive search on 6 and 7-round AES-128, we give estimates using Table 2.

A.1 The Square Attack

The square or integral attack was proposed in [DKR97]. It was studied in the original specification document AES [DR99] targeting 6 rounds, and extended later to 7 rounds when considering AES-192 and 256 [FKL+00].

This attack relies on a distinguisher on 3 rounds of AES (figure 7). Let S be a δ-set in byte 0, that is, 2^8 ciphertexts that take all values on byte 0 but are constant on the other bytes. Let $E(S)$ be the set of encryptions of S through 3 rounds of AES, regardless of the round keys. Then any byte in $E(S)$ is balanced: the XOR of all the values it takes is 0. On the contrary, for a random function, the probability that this event happens is $\frac{1}{2^8}$.

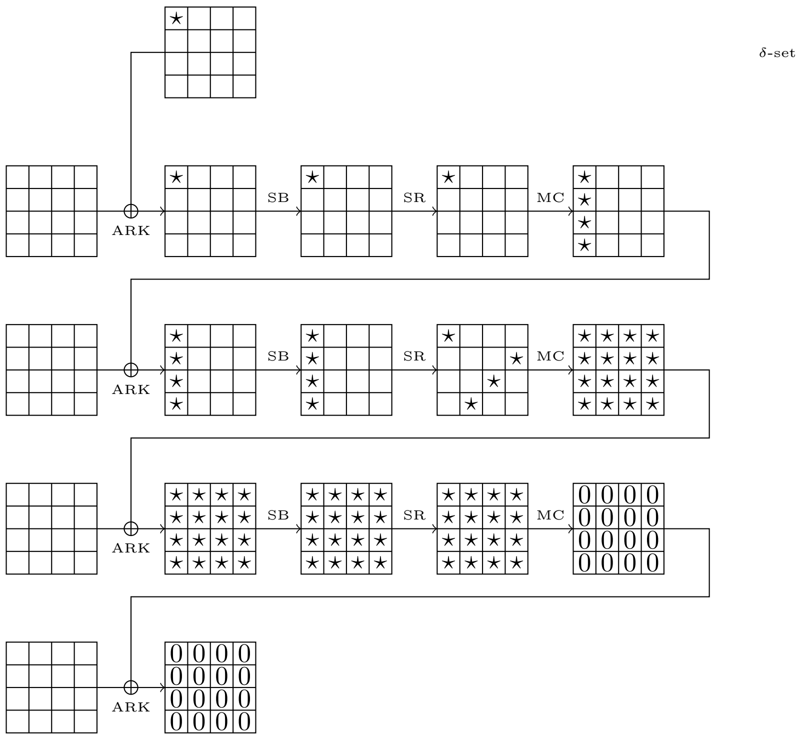

Figure 7: Integral distinguisher on 3 rounds of AES. A byte marked by 0 is balanced.

The reason for this property to hold is that the S-Box being a bijective function, it maps a δ-set to another one ; besides, each byte in the active column at the end of the first round takes 256 values. Finally, the sum of all 256 byte values in any byte position of the last state is equal to the linear combination of some sums of some bytes that each assume all values, and hence is zero.

Square Attack on 6-round AES. The resulting attack on 6 rounds, taken from [DKR97] and improved in [FKL+00] is described in figure 8. Append one round before and two rounds after the distinguisher. Encrypt a set of 2^{32} plaintexts that make the main diagonal vary but are constant on all other bytes. Then, regardless of the first round key, they give 2^{24} δ-sets at the end of the first round. When given to the three inner rounds, this gives balanced bytes as well. Using this, we don't have to guess the first round key.

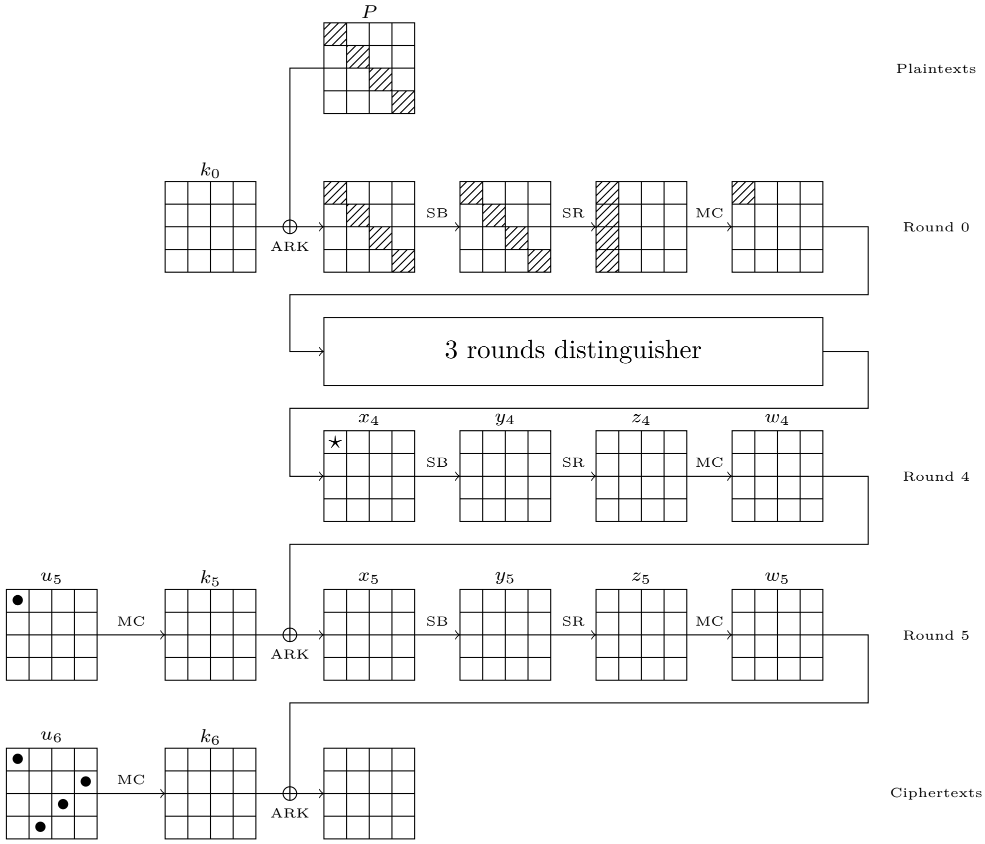

Figure 8: Square attack on 6-round AES

A.2 Q1 Square Attack on 6-round AES

First, we concentrate on the initial 6-round Square attack from [DKR97]. In Algorithm 2, we rewrite it as a classical FILTER, with a simple translation as a quantum algorithm.

Algorithm 2: Quantum square attack on 6-round AES

Input: 8 *structures of* 2^{32} *classical chosen-plaintext queries such that the main diagonal* $x_0[0, 5, 10, 15]$ *takes all values*

Result: The key bytes $u_5[0], u_6[0, 7, 10, 13]$

Filter $u_5[0], u_6[0, 7, 10, 13]$ **such that:**

`(One solution among` 2^{40}`)`

For each structure, partially decrypt the 2^{32} ciphertexts through the two last rounds. Compute the xor of all values in $x_4[0]$. If it is zero for all 8 structures, output this guess.

Using the square property as a 3-round distinguisher, each structure gives a one-byte condition. Although 5 would be enough, we use 8 structures in order to ensure that only the right key guess passes the test, with overwhelming probability. Hence the number of Grover iterations is exactly known. As each partial decryption requires 5 S-Boxes, the classical time complexity in S-Boxes is:

$$2^{40} \times 2^{32} \times 8 \times 5 \leq 2^{78}$$

and the quantum time complexity is: $\lceil \frac{\pi}{4} 2^{20} \rceil \times (2^{32} \times 2 \times 8 \times 5) \leq 2^{58}$. As the input data is only classical, the attack model is Q1 (no superposition queries needed).

Partial Sums Technique. The partial sums technique from [FKL+00] already makes the classical time complexity decrease to 2^{52} S-Boxes (2^{44} encryptions). Our next goal would be to adapt this technique. We managed this only using qRAM.

Assume for readability that the last MixColumns and ShiftRows operations are omitted: the five guessed key bytes are now $u_5[0]$ and $k_6[0,1,2,3]$. We want to find the guesses of those bytes such that the sum of $x_4[0]$ on all 2^{32} ciphertexts c_i is zero:

$$\sum_i x_4[0]^i = \sum_i S^{-1}(y_4[0]^i) \ .$$

We write $y_4[0]$ as a combination of $x_5[0,\dots,3]$:

$$\sum_i x_4[0]^i = \sum_i S^{-1}\left(a_0 x_5[0]^i + a_1 x_5[1]^i + a_2 x_5[2]^i + a_3 x_5[3]^i + u_5[0]\right)$$

for some known coefficients a_0, a_1, a_2, a_3. In turn this rewrites:

$$\sum_i S^{-1}\big(a_0 S^{-1}(c_i[0] + k_6[0]) + a_1 S^{-1}(c_i[1] + k_6[1]) \\ + a_2 S^{-1}(c_i[2] + k_6[2]) + a_3 S^{-1}(c_i[3] + k_6[3]) + u_5[0]\big) \ . \quad (8)$$

The presentation from [FKL+00] constructs successive tables containing *partial sums*. The final table contains, for each 5-byte key guess (2^{48}), the value of the whole sum. In Algorithm 3, we rewrite this procedure as a composition of FILTERs. Having 8 structures ensures that only the right key guess passes at each step, with overwhelming probability.

The classical time complexity of this procedure (in S-Boxes and inverse S-Boxes) is:

$$\underbrace{2^{2\times 8}}_{\text{Over } k_6[0,1]} \left(16 \cdot 2^{32} + \underbrace{2^8}_{\text{Over } k_6[2]} \left(2^{24} \cdot 8 + \underbrace{2^8}_{\text{Over } k_6[3]} \left(2^{16} \cdot 8 + \underbrace{2^8}_{\text{Over } u_5[0]} \cdot 2^8 \cdot 8\right)\right)\right) \ .$$

We can bound it as less than 2^{54} S-Boxes. Furthermore, this procedure uses 2^{35} classical queries and 8×2^{24} 32-bit registers of classical RAM (each step needs efficient random-access to the table of the previous step).

Quantum Equivalent. The quantum equivalent of Algorithm 3 performs nested Amplitude Amplification procedures. As its classical counterpart, it uses random-accessible memory. The memory amounts are the same classically and quantumly. The algorithm requires 8×2^{24} 32-qubit registers of quantum RAM, accessible in superposition, and 2^{35} 256-bit registers of *classical* memory to store the chosen-plaintext queries.

We write $c = \pi/4$. The time complexity, adapted from the classical one, is:

$$\lceil c2^8 \rceil \left(2 \times 2^{36} + 2 \lceil c2^4 \rceil \left(2 \times 2^{27} + 2 \times \lceil c2^4 \rceil \left(2 \times 2^{19} + 2 \lceil c2^4 \rceil \times 2 \times 2^{11}\right)\right)\right)$$

where additional factors stem from uncomputations. We approximate the number of iterations of each subprocedure as $\lceil c2^4 \rceil = 13$ and $\lceil c2^8 \rceil = 201$. Thanks to using 8 structures, we are ensured of only one solution at each step (with high probability) so that the exact number of Grover iterations is known, and error corrections are efficient. We obtain a quantum time equivalent of $2^{44.73}$ reversible S-Boxes.

Algorithm 3: Square attack on 6-round AES with the partial sums technique

Input: 8 *structures of* 2^{32} *classical chosen-plaintext queries such that the main diagonal* $x_0[0, 5, 10, 15]$ *takes all values*

Result: The key bytes $u_5[0]$, $k_6[0, 1, 2, 3]$

Filter $k_6[0], k_6[1]$ **such that:**

`(One solution among` 2^{16}`)`

Do the following for each input structure:

For each ciphertext c_i, compute the three byte-value $a_0 S^{-1}(c_i[0] + k_6[0]) + a_1 S^{-1}(c_i[1] + k_6[1]), c_i[2], c_i[3]$

Build a table T_1 of 2^{24} entries which stores, for each three-byte value, how many times it appears when c_i runs over all the ciphertexts.

> **Filter** $k_6[2]$ **such that:**
>
> `(One solution among` 2^8`)`
>
> Do the following for each input structure:
>
> Using the entries of T_1, compute the two-byte value $a_0 S^{-1}(c_i[0] + k_6[0]) + a_1 S^{-1}(c_i[1] + k_6[1]) + a_2 S^{-1}(c_i[2] + k_6[2]), c_i[3]$ for each c_i
>
> Build a table T_2 of 2^{16} entries which stores, for each two-byte value, how many times it appears when c_i runs over all the ciphertexts.
>
> > **Filter** $k_6[3]$ **such that:**
> >
> > `(One solution among` 2^8`)`
> >
> > Do the following for each input structure:
> >
> > Using the entries of T_2, compute the byte value $a_0 S^{-1}(c_i[0] + k_6[0]) + a_1 S^{-1}(c_i[1] + k_6[1]) + a_2 S^{-1}(c_i[2] + k_6[2]) + a_3 S^{-1}(c_i[3] + k_6[3])$ for each c_i
> >
> > Build a table T_3 of 2^8 entries which stores, for each byte value, how many times it appears when c_i runs over all the ciphertexts.
> >
> > **Filter** $u_5[0]$ **such that:**
> >
> > > `(One solution among` 2^8`)`
> > >
> > > Do the following for each input structure:
> > >
> > > Using table T_3, compute the sum (8)
> > >
> > > If the xor is zero for each structure, return this guess of $u_5[0]$
> >
> > If there is a result for $u_5[0]$, return this guess of $k_6[3]$
>
> If there is a result for $k_6[3]$, return this guess of $k_6[2]$

If there is a result for $k_6[2]$, return this guess of $k_6[0, 1]$

A.3 Q1 Square Attack on 7-round AES

To attack 7 rounds of AES, we append a round to the previous attack and guess completely the last round key k_7. With this method, the 256 and 192 variants are within reach.

Without Partial Sums. Using the attack framework of [DKR97], we retrieve the key with 2^{37} chosen-plaintext queries, a quantum time equivalent to 2^{121} reversible S-Boxes, a small number of qubits, 2^{37} classical memory and no qRAM.

First of all, we increase the key search space to 20 unknown bytes, so we need more chosen plaintext queries as before. 2^5 sets of 2^{32} plaintexts are sufficient and ensure to have only one result with high probability. We perform Grover search over a search space of size $2^{20\times 8}$ (the partial key bytes) and expect one solution; testing is done sequentially in time $2^{32} \times 2^5 \times 5$ S-Boxes, by computing the 2^5 XORs in $x_4[0]$. So the quantum time complexity is $c2^{20\times 4}\left(2^{32} \times 2^5 \times 10\right) \leq 2^{121}$ S-Boxes.

This constitutes an attack for AES-256, as the time complexity beats Grover's $2^{138.04}$ S-Boxes. However, it is above the AES-192 Grover search ($2^{105.25}$). This procedure does also not better than the classical 7-round impossible differential and Meet-in-the-middle attacks (see Section 2.2.1), unless we strictly compare the number of S-Boxes.

With Partial Sums. We obtain a better time complexity by wrapping Algorithm 3 inside a Grover search over the additional key bytes. For AES-256, there are 15 more bytes to search for in the "outer" Grover, hence $c2^{60}$ iterations. This gives 2^{107} reversible S-Boxes in total (as there are also more structures needed) and 2^{29} 32-qubit registers of quantum RAM. For AES-192, there is one less key byte to guess, due to key-schedule properties. We get $2^{103.4}$ S-Boxes against $2^{105.25}$ for Grover search;

B Sequence of values to test to solve the quadratic equation

Remark 8. We could gain 5 CNOT gates by writing 0x0 and not 0xbc in the case $d^{-1} = 1$. This would not be a solution of the equation, but as it corresponds to the case where we have 4 solutions to the differential equation, it would be a valid solution for us.

Table 4: sequence of values to test, associated with the solution to write, from left to right and high to low.

(d^{-1},x)	(d^{-1},x)	(d^{-1},x)	(d^{-1},x)	(d^{-1},x)	(d^{-1},x)
(0x1, 0xbc)	(0x3, 0x7e)	(0x5, 0x88)	(0x8, 0xd6)	(0x9, 0xb6)	(0xc, 0xf2)
(0xd, 0xec)	(0xe, 0xc4)	(0x11, 0xda)	(0x12, 0x72)	(0x13, 0x9e)	(0x17, 0x9a)
(0x18, 0x24)	(0x1a, 0x6e)	(0x1c, 0x44)	(0x1d, 0x8e)	(0x1e, 0xd4)	(0x1f, 0xd8)
(0x22, 0xe)	(0x23, 0xc0)	(0x24, 0x36)	(0x25, 0x9c)	(0x26, 0x58)	(0x29, 0xac)
(0x2b, 0xb8)	(0x2d, 0xa6)	(0x2e, 0xf4)	(0x31, 0x1a)	(0x34, 0xea)	(0x37, 0xa4)
(0x38, 0x56)	(0x3d, 0xe2)	(0x3e, 0x98)	(0x3f, 0x16)	(0x40, 0x3e)	(0x41, 0xf8)
(0x46, 0xe8)	(0x48, 0x60)	(0x4a, 0xce)	(0x4b, 0xba)	(0x4c, 0x8a)	(0x4e, 0x6a)
(0x4f, 0x3a)	(0x50, 0x42)	(0x51, 0xc)	(0x52, 0x94)	(0x54, 0x20)	(0x57, 0xca)
(0x58, 0xaa)	(0x5b, 0x7c)	(0x5c, 0x1e)	(0x5d, 0xfe)	(0x5f, 0x92)	(0x60, 0x2e)
(0x61, 0x26)	(0x62, 0xe6)	(0x64, 0xb4)	(0x65, 0xdc)	(0x67, 0x18)	(0x68, 0x54)
(0x69, 0x30)	(0x71, 0x4c)	(0x74, 0x2c)	(0x75, 0x66)	(0x76, 0x5e)	(0x78, 0xf0)
(0x7b, 0x2)	(0x7c, 0x62)	(0x7d, 0xd0)	(0x86, 0x76)	(0x87, 0xa0)	(0x8c, 0xc2)
(0x8d, 0x2a)	(0x8e, 0xc8)	(0x8f, 0xf6)	(0x99, 0x4)	(0x9c, 0x46)	(0xa0, 0x6c)
(0xa5, 0x74)	(0xa7, 0x8)	(0xaa, 0x6)	(0xab, 0x22)	(0xad, 0xee)	(0xb0, 0xae)
(0xb1, 0x50)	(0xb2, 0x14)	(0xb3, 0x68)	(0xb4, 0x90)	(0xb8, 0x4a)	(0xbc, 0xe0)
(0xbd, 0x5c)	(0xbf, 0x1c)	(0xc0, 0x10)	(0xc3, 0x48)	(0xc6, 0x78)	(0xc7, 0x38)
(0xc8, 0xe4)	(0xca, 0x34)	(0xcb, 0x28)	(0xcc, 0x3c)	(0xcd, 0xd2)	(0xce, 0x70)
(0xcf, 0x52)	(0xd1, 0xbe)	(0xd2, 0x5a)	(0xd6, 0x7a)	(0xd7, 0xfc)	(0xdd, 0xfa)
(0xe0, 0x4e)	(0xe1, 0x12)	(0xe3, 0x40)	(0xe5, 0x84)	(0xe7, 0xcc)	(0xe8, 0x86)
(0xe9, 0xa)	(0xeb, 0xc6)	(0xec, 0xb0)	(0xed, 0xa2)	(0xee, 0xa8)	(0xef, 0x64)
(0xf0, 0xb2)	(0xf5, 0x8c)	(0xf6, 0x96)	(0xfb, 0xde)	(0xfd, 0x80)	(0xfe, 0x32)
(0xff, 0x82)					