A first attempt at cryptanalyzing a (toy) block cipher by means of QAOA

Luca Phab[1], Stéphane Louise[2], and Renaud Sirdey[2]

[1] Université Paris-Saclay, France
[2] Université Paris-Saclay, CEA, List, F-91120, Palaiseau France

luca.phab@tutanota.com, stephane.louise@cea.fr, renaud.sirdey@cea.fr

Abstract. The discovery of quantum algorithms that may have an impact on cryptography is one of the main reasons of the rise of quantum computing. Currently, all quantum cryptanalysis techniques are purely theoretical and none of them can be executed on existing or near-term quantum devices. So, this paper investigates the capability of already existing quantum computers to attack a toy block cipher (namely the Heys cipher) using the Quantum Approximate Optimization Algorithm (QAOA). Starting from a known-plaintext key recovery problem, we transform it into an instance of the MAX-SAT problem. Then, we propose two ways to implement it in a QAOA circuit and we try to solve it using publicly available IBM Q Experience quantum computers. The results suggest that the limited number of qubits requires the use of exponential algorithms to achieve the transformation of our problem into a MAX-SAT instance and, despite encouraging simulation results, that the corresponding quantum circuit is too deep to work on nowadays (too-)noisy gate-based quantum computers.

Keywords: Cryptography · Quantum Computing · QAOA · MAX-SAT · Block cipher

1 Introduction

Quantum computing offers a new paradigm that can solve certain problems much more efficiently than classical computing. At the same time, a large part of modern cryptography is precisely based on the difficulties to solve specific problems that are conjectured hard to solve with classical computing.
On one hand, Shor's algorithm [16] can solve factorization and discrete logarithm problems, that are of huge importance in cryptography. On the other hand, Grover's algorithm [8] can be utilized as a quantum brute-force algorithm that is much more efficient than the classical brute-force (but remains exponential). Hence, quantum computing may be a serious threat to cryptography and that is why a lot of research is conducted on quantum cryptanalysis. For a few years, we have seen an emerging field on quantum non-black box cryptanalysis [12,13,11,5,4,6], meaning that those approaches are cipher specific by exploiting their internal structure. Yet all those works assume a large scale and noise-free quantum computer that does not currently

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08760-8_19

exist.
In this context, this paper investigates (for the first time) whether Noisy Intermediate Scale Quantum (NISQ) machines may have an impact in cryptanalysis of symmetric ciphers. To that end, we will study an attack on a toy cipher, namely the Heys cipher, using the Quantum Approximate Optimization Algorithm (QAOA) that is expected to be less sensitive to decoherence due to its hybrid nature. The proposed attack can be summarized as follows: We first need a plaintext and the corresponding ciphertext. Then, we transform it into an instance of a combinatorial optimization problem. Finally, we execute the quantum algorithm to solve the instance and if an optimal solution is found then we can deduce a key that can encrypt the plaintext into the ciphertext.
This paper is organized as follows: In Section 2, we review QAOA. Section 3 provides some background on cryptography and describes the studied toy cipher and the attack steps. Then, in Section 4, we show how to build the quantum circuit and indicate its complexity according to the basis gates of the quantum hardware utilized. Finally, Section 5 details the implementation and the experimental results before concluding.

2 Quantum Approximate Optimization Algorithm

QAOA [7] is an algorithm created in 2014 by Farhi et al. that approximates solutions of combinatorial optimization problems. It is a quantum circuit whose parameters are optimized through a classical optimization algorithm. Its hybrid nature allows this algorithm to limit the depth of the quantum circuit, hence, it seems relevant as an algorithm of choice to solve our problem on NISQ devices.
It is based on a well-known quantum mechanics theorem, called the "adiabatic theorem", stating that a quantum system in a ground state for a hamiltonian will remain in a ground state for that hamiltonian if it changes slowly enough over time. Let H be the hamiltonian such that:

$$H(t)=(1-s(t))\cdot H_D+s(t)\cdot H_P \tag{1}$$

where $s:[0,T]\to[0,1]$ is a smooth function with $s(0)=0$ and $s(T)=1$, H_D is the "driver hamiltonian" that has an easy-to-build ground state and H_P is the "problem hamiltonian" whose ground states encode the optimal solutions of our problem. So, according to the adiabatic theorem, a quantum system in an easy-to-build ground state for H_D will evolve into a ground state for H_P, which encodes optimal solutions of the problem, after waiting a time T.
Let C be the cost function that we want to minimize and which takes n binary variables as input. The corresponding problem hamiltonian can be chosen as:

$$H_P=\sum_{x\in\{0,1\}^n} C(x)|x\rangle\langle x| \tag{2}$$

QAOA simulates the approximate time evolution of a quantum system for the hamiltonian H starting from one of its ground states, using n qubits.

Based on the Schrödinger equation and the Trotter-Suzuki formula [17], the time evolution of the quantum system $|\psi\rangle$ for the hamiltonian H is approximated by:

$$|\psi(t)\rangle \approx \prod_{j=1}^{p} \exp\left(-\frac{i\Delta t}{\hbar}(1-s(j\Delta t))H_D\right)\cdot\exp\left(-\frac{i\Delta t}{\hbar}s(j\Delta t)H_P\right)|\psi(0)\rangle \quad (3)$$

whose precision as an approximation is improved by increasing p.
Let $U_{H_P}(\gamma)=\exp(-i\gamma H_P)$ and $U_{H_D}(\beta)=\exp(-i\beta H_D)$ be unitary operators, the QAOA circuit computes the following state:

$$|\vec{\beta},\vec{\gamma}\rangle = U_{H_D}(\beta_{p-1})U_{H_P}(\gamma_{p-1})\cdots U_{H_D}(\beta_0)U_{H_P}(\gamma_0)|\psi(0)\rangle \quad (4)$$

where $\vec{\beta}=(\beta_0,\ldots,\beta_{p-1})\in[0,2\pi[^p$ and $\vec{\gamma}=(\gamma_0,\ldots,\gamma_{p-1})\in[0,2\pi[^p$. We do not know the values of p, $\vec{\beta}$ and $\vec{\gamma}$ such that $|\vec{\beta},\vec{\gamma}\rangle$ is a ground state for H_P so we try to find them empirically, at constant p, by minimizing the function:

$$\begin{aligned} F_p : [0,2\pi[^p \times [0,2\pi[^p &\to \mathbb{R} \\ (\vec{\beta},\vec{\gamma}) &\mapsto \langle\vec{\beta},\vec{\gamma}|H_P|\vec{\beta},\vec{\gamma}\rangle \end{aligned} \quad (5)$$

using a classical optimizer. That function calculates the average of the eigenvalues of H_P weighted by the probability distribution of states of $|\vec{\beta},\vec{\gamma}\rangle$ and its minimum is reached exactly when the superposition is in a ground state for H_P. It can be computed with the probability distribution that we can approximate by executing several times the quantum circuit and measuring the resulting final state:

$$\forall x\in\{0,1\}^n,\ \mathbb{P}_{\vec{\beta},\vec{\gamma}}(\text{measuring state } |x\rangle)\approx\frac{k_x}{k} \quad (6)$$

where k is the number of executions and k_x is the number of times that the state $|x\rangle$ was measured.
Finally, we just need to compute the quantum circuit with the parameters $\vec{\beta}^*$ and $\vec{\gamma}^*$, found by optimization, and then to measure the superposition to get a state that encodes a solution z where $C(z)$ is close to $\min_x C(x)$.

3 Problem statement

3.1 Cryptography background

Cryptography is all around us, especially when we use our communication devices, as smartphones or computers. It aims to ensure the security properties of an exchange, which are confidentiality, integrity, authentication and non-repudiation. To protect the confidentiality of a message, the sender utilizes a cipher that transforms (encrypts) the message (plaintext) into an unintelligible one (ciphertext), sends it on a channel, that can be public, to the recipient, who applies the inverse of the transformation (decrypts) to obtain the original message. An extra parameter (key) is needed for the encryption and decryption functions.

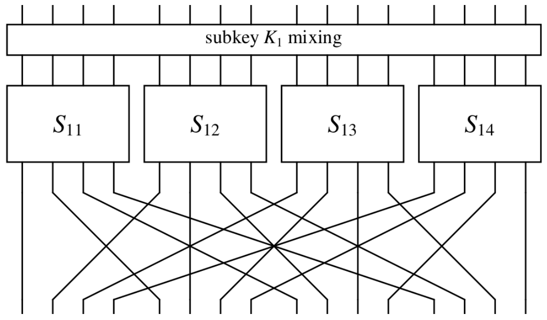

Fig. 1: First round of the Heys cipher.

In a symmetric cipher, as studied in the present work, the encryption key and the decryption key are computed from a secret shared between the sender and the recipient. Among them, a large part are iterated block ciphers, that encrypt a predefined fixed-size plaintext (block) into a ciphertext with the same size, using the repetition of several operations. Each iteration (round) needs a key (round key) derived from a master key. The number of rounds is a significant parameter of the cipher because increasing it generally improves its security but reduces its efficiency, which is also an important criterion. Doing a cryptanalysis on a reduced number of rounds is an usual way to study that type of ciphers, which is easier and that can give insights on its security. In this paper, the proposed cryptanalysis is a known-plaintext attack, meaning that the attacker has access to a set of plaintext-ciphertext pairs, which is a realistic possibility. Indeed, that kind of attack was already utilized previously, during World War II for instance. Allies guessed the content (or partial content) of some messages (mainly weather forecasts) and utilized that to break the German cipher Enigma [19].

3.2 Heys Cipher

The Heys cipher [9] is an iterated block cipher created by Howard M. Heys for educational purposes to teach linear and differential cryptanalysis. It is a toy substitution-permutation network (SPN) with a block size of 16 bits and where each of its rounds is composed of:

- a mixing operation between the round key and the current block
- a substitution
- a permutation

The first round is described in Fig. 1. The cipher finishes with a new mixing operation which enables the decryption function to be as similar as the encryption function. That final step can be considered as a round because it requires the utilization of a round key. The mixing operation is commonly an exclusive or (xor). The cipher has the advantage to be particularly simple and has the same construction as a lot of ciphers in practical utilization currently or formerly, like e.g. DES or AES.

3.3 Principle of the attack

After getting a plaintext and the corresponding ciphertext for which we want to find the key utilized for the encryption, we transform them into an instance of the SAT problem, namely a propositional formula, where the variables embody the bits of the sought key. If we had an oracle solving the SAT problem, we could utilize it to find an assignment satisfying the formula and thus deduce a key that can encrypt the plaintext into the ciphertext. But, as SAT is NP-complete, all known classical algorithms cannot solve all formula polynomially. In particular, formulas that come from cryptanalytic attacks can be expected to be difficult to solve (otherwise, the underlying primitive would have significant security issues). Among quantum algorithms, QAOA is a very promising one that could run on NISQ machines, so we convert the formula into an equivalent formula in Conjunctive Normal Form (CNF), which is an instance of the combinatorial optimization version of the SAT problem (MAX-SAT). As our formula is satisfiable by construction, an optimal solution is an assignment satisfying the formula, so the algorithm will act as the required oracle. It is important to notice that we need to find algorithms to convert a plaintext-ciphertext pair that do not add too many auxiliary variables, given that the number of qubits required depends on the number of binary variables of our problem and that the number of qubits of a quantum machines (real devices or simulators) is very limited.

4 Retrieving the encryption key from a plaintext-ciphertext pair using QAOA

4.1 Plaintext-ciphertext pair into Conjunctive Normal Form formula (CNF) conversion

The algorithm aim is to transform a plaintext and the corresponding ciphertext into a propositional formula where the variables embody the utilized cipher key bits (without adding any auxiliary variables). It is based on a theorem saying that a formula where we substitute variables with formulas remains a formula. So, we start with a simple formula and we substitute variables with formulas, as it goes along the cipher operations, in the current formula.
Let $(p_i)_{i\in\{1,\ldots,16\}}$ be the plaintext bits, $(c_i)_{i\in\{1,\ldots,16\}}$ the ciphertext bits and $(k_i)_{i\in\{1,\ldots,16\}}$ the key bits. The starting formula is:

$$\psi_{init} = \bigwedge_{i=1}^{16} (p_i \leftrightarrow x_i) \tag{7}$$

Then, for each operation, some variables of the current formula are replaced with formulas according to the operation type. We denote $(a_i)_{i\in\{1,\ldots,16\}}$ the input variables of an operation and $(b_i)_{i\in\{1,\ldots,16\}}$ the output variables.

j-th xor: For all $i \in \{1,\ldots,16\}$, we have $b_i = a_i \oplus k_{16(j-1)+i}$, hence $a_i = b_i \oplus k_{16(j-1)+i}$. So, variables $(x_i)_{i\in\{1,\ldots,16\}}$ in the current formula are substituted with formulas $(\phi_{1,j,i})_{i\in\{1,\ldots,16\}}$ as follows:

$$\phi_{1,j,i} = y_i \oplus k_{16(j-1)+i} \equiv \big((y_i \vee k_{16(j-1)+i}) \wedge (\neg y_i \vee \neg k_{16(j-1)+i})\big) \tag{8}$$

Table 1: (a) Table representing the inputs and outputs of a random sbox on 2 bits. (b) Table representing the possible outputs of the sbox described in (a) where $a_1 = 1$ and the corresponding formulas.

(a)

input (a_2a_1)	00	01	10	11
output (b_2b_1)	10	00	01	11

(b)

output	formula
00	$\psi_{00} = (\neg z_1 \wedge \neg z_2)$
11	$\psi_{11} = (z_1 \wedge z_2)$

***S_{jk}* substitution:** The S_{jk} substitution acts on bits of index from $4(k-1)+1$ to $4(k-1)+4$. The replacement formula $\phi_{2,j,i}$ where $i = 4(k-1)+h$, $h \in \{1,2,3,4\}$ is a disjunction of conjunctions created from the outputs for which the bit of index i in input is equal to 1.

For example, suppose that we want to construct the replacement formula $\phi_{2,j,i}$ from the sbox[3] described in Table 1(a) for $i = 1$. The outputs where $a_1 = 1$ in input are "00" and "11". For each of those outputs, the corresponding formula, as shown in Table 1(b), is the conjunction of $(l_s)_{s \in \{1,2\}}$ such that:

$$l_s = \begin{cases} z_s & \text{if } b_s = 1 \\ \neg z_s & \text{otherwise} \end{cases} \tag{9}$$

That way, if $a_1 = 1$ then one of the formulas ψ_{00} or ψ_{11} must be True thus the replacement formula is:

$$\phi_{2,j,1} = \psi_{00} \vee \psi_{11} \tag{10}$$

σ permutation: For all $i \in \{1,...,16\}$, we have $b_i = a_{\sigma(i)}$, hence $a_i = b_{\sigma^{-1}(i)}$. So, variables $(z_i)_{i \in \{1,...,16\}}$ are replaced with formulas $(\phi_{3,j,i})_{i \in \{1,...,16\}}$ as follows:

$$\phi_{3,j,i} = x_{\sigma^{-1}(i)} \tag{11}$$

Finally, after substituting the variables for all cipher operations, all it is required is to replace $(x_i)_{i \in \{1,...,16\}}$ according to the value of the ciphertext bits to obtain a formula where the only variables embody the key bits.

It is a naive algorithm where the size of the output formula increases exponentially with the number of cipher rounds. That is due to the substitution operations where the linked replacement formulas contain many multiple copies of the same variable. However, with a number of rounds small enough, the size of the formulas remains acceptable.

Then, the formula must be in conjunctive normal form to be an instance of the MAX-SAT problem. There are two main algorithms to achieve that transformation. The first one [10] use the De Morgan laws but the output formula grows exponentially with the size of the input formula, while the second, called Tseitin transformation [18], is polynomial but add auxiliary variables. We chose the algorithm that does not increase the number of variables (and thus the number of qubits needed by the quantum

[3] a sbox (substitution box) is an algorithm component that compute a substitution in a cipher.

devices) to be able to perform experiments on real hardware on our instances.
It is worth noting that we probably could transform the plaintext-ciphertext pair directly into a CNF by introducing auxiliary variables as in [14] where the output formula has a polynomial number of variables and clauses.

4.2 Solving the MAX-SAT problem using QAOA

Let φ be a propositional formula in conjunctive normal form and $P_\varphi = \{x_1,...,x_n\}$ the set of the variables in φ such that:

$$\varphi = \bigwedge_{j=1}^{m} C_j = \bigwedge_{j=1}^{m} \left(\bigvee_{k=1}^{m_j} l_{j,k} \right) \tag{12}$$

where $l_{j,k} = x_{j,k}$ or $l_{j,k} = \neg x_{j,k}$. We denote $M = \max_{j\in\{1,...,m\}} m_j$ the maximum number of literals in a clause. An assignment of n variables is embodied by $z = (z_1,...,z_n) \in \{0,1\}^n$ such that:

$$\forall l \in \{1,...,n\},\ (z_l = 1 \Leftrightarrow x_l = True) \text{ and } (z_l = 0 \Leftrightarrow x_l = False) \tag{13}$$

The cost function associated with φ is defined as follows:

$$C(z) = -\sum_{j=1}^{m} C_j(z) \text{ where } C_j(z) = \begin{cases} 1 \text{ if } z \text{ satisfies } C_j \\ 0 \text{ otherwise} \end{cases} \tag{14}$$

and can be transformed into a problem hamiltonian such that:

$$H_P = \sum_{x\in\{0,1\}^n} C(x)|x\rangle\langle x| = -\sum_{j=1}^{m} \hat{C}_j \text{ where } \hat{C}_j = \sum_{x\in\{0,1\}^n} C_j(x)|x\rangle\langle x| \tag{15}$$

$$= -\sum_{j=1}^{m} (I - \frac{1}{2^{m_j}} \prod_{k=1}^{m_j} (I + \varepsilon_{j,k} \cdot \sigma^z_{j,k})) \text{ where } \varepsilon_{j,k} = \begin{cases} -1 \text{ if } l_{j,k} = \neg x_{j,k} \\ 1 \quad \text{otherwise} \end{cases} \tag{16}$$

Using the definition of H_P in Eq. 15 and the Trotter-Suzuki formula, U_{H_P} can be written as the product:

$$U_{H_P}(\gamma) = \prod_{j=1}^{m} U_{-\hat{C}_j}(\gamma) \tag{17}$$

So, the quantum circuit of $U_{H_P}(\gamma)$ is composed of the quantum circuits of $U_{-\hat{C}_j}(\gamma)$ with $j \in \{1,...,m\}$, in any order since the $\hat{C}_j$ commute. For all $j \in \{1,...,m\}$, we can note that:

$$U_{-\hat{C}_j}(\gamma)|z\rangle = \exp\left(i\gamma\hat{C}_j\right)|z\rangle = \begin{cases} \exp(i\gamma)|z\rangle & \text{if } z \text{ satisfies } C_j \\ 1|z\rangle & \text{else} \end{cases} \tag{18}$$

But, the satisfiability of $C_j = \bigvee_{k=1}^{m_j} l_{j,k}$ only depends on variables $(x_{j,k})_{k\in\{1,...,m_j\}}$ so $U_{-\hat{C}_j}(\gamma)|z\rangle$ only depends on qubits $(z_{j,k})_{k\in\{1,...,m_j\}}$. Moreover, there is a unique

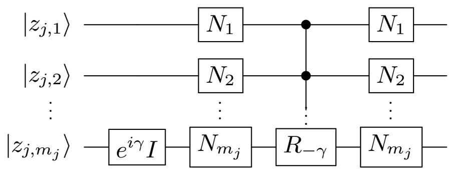

Fig. 2: Quantum circuit modelling $U_{-\hat{C}_j}(\gamma)$ where N_k is the X gate if $l_{j,k}=x_{j,k}$ and the identity gate if $l_{j,k}=\neg x_{j,k}$, for all $k\in\{1,...,m_j\}$.

assignment of $(z_{j,k})_{k\in\{1,...,m_j\}}$ such that $U_{-\hat{C}_j}(\gamma)|z\rangle=|z\rangle$, for all others, $U_{-\hat{C}_j}(\gamma)|z\rangle=\exp(i\gamma)|z\rangle$. Using those statements, we can model the quantum circuit of $U_{-\hat{C}_j}(\gamma)$ as shown in Fig. 2. To the best of our knowledge, the multi-controlled R_ϕ gate cannot be decomposed into 1-qubit and 2-qubit gates in a polynomial number of gates. So, the complexity of the circuit is:

$$\text{Comp}(U_{H_P})=\mathcal{O}(m\cdot M\cdot 2^M) \quad (19)$$

$$\text{Depth}(U_{H_P})=\mathcal{O}(m\cdot 2^M) \quad (20)$$

We can also use the definition of H_P in Eq. 16 using the Pauli operator σ^z. In that case, for all $I\subseteq\{1,...,n\}$ with $|I|\leq M$, there exists $a_I\in\mathbb{R}$, such that:

$$U_{H_P}(\gamma)=\exp\left(i\gamma\sum_I a_I\prod_{k\in I}\sigma_k^z\right)=\prod_I\exp\left(i\gamma a_I\prod_{k\in I}\sigma_k^z\right) \quad (21)$$

So, the quantum circuit of $U_{H_P}(\gamma)$ is composed of quantum circuits of $i\gamma a\prod_k\sigma_k^z$, in any order, which can be modelling by the quantum circuit in Fig. 3(a). Its complexity is:

$$\text{Comp}(U_{H_P})=\mathcal{O}\left(\sum_{k=1}^{M}\binom{n}{k}\cdot(2k-1)\right) \quad (22)$$

$$\text{Depth}(U_{H_P})=\text{Comp}(U_{H_P}) \quad (23)$$

The driver hamiltonian is often the same, namely:

$$H_D=-\sum_{j=1}^{n}\sigma_j^x \quad (24)$$

It is a well-known mixing operator for which a ground state is:

$$|+\rangle^{\otimes n}=H^{\otimes n}|0\rangle^{\otimes n} \quad (25)$$

The quantum circuit of $U_{H_D}(\beta)=\bigotimes_{j=1}^n R_x(-2\beta)$ is modelled in Fig. 3(b) and its complexity is:

$$\text{Comp}(U_{H_D})=\mathcal{O}(n) \quad (26)$$

$$\text{Depth}(U_{H_D})=\mathcal{O}(1) \quad (27)$$

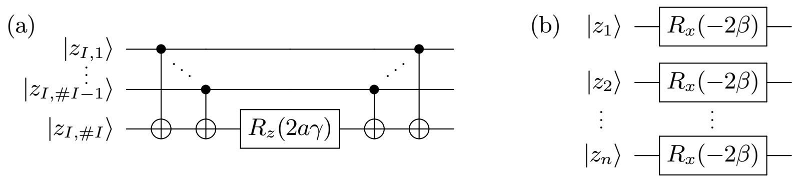

Fig. 3: Quantum circuits modelling (a) $i\gamma a\prod_k \sigma_k^z$ and (b) $U_{H_D}(\beta)$

5 Experimental results

First, we generate random plaintexts and keys and we encrypt them using our implementation of the Heys cipher, on two over five rounds, to obtain a set of triplets (key, plaintext, ciphertext). Then, we transform each one into a Conjunctive Normal Form formula (CNF) using our implementation of propositional formula that can do some simplifications to limit the size of the output formula. To study the formulas for different number of variables, we replace in the formula some variables with the corresponding key bits.
Then, we utilize the SymPy library [3] to build the hamiltonian as a symbolic expression based on the formula with the Pauli operator σ^z in Eq. 16. The final step is to construct the corresponding quantum circuit with the Qiskit library [2] and to give it as input to our own implementation of QAOA. It utilizes the Qiskit Constraint Optimization BY Linear Approximation (COBYLA) optimizer [15] (that seems to be the best among the Qiskit optimizers for this work).
Before trying the algorithm on IBM Q Experience [1] real quantum devices, we have studied the theoretical performance of QAOA on our problem with the Qiskit Aer simulators. To estimate the performances, the success probability and the following approximation ratio was used as a metric:

$$r^* = \frac{F_p\left(\vec{\beta}_p^*, \vec{\gamma}_p^{\,*}\right)}{\min\limits_x C(x)} \tag{28}$$

where $\vec{\beta}_p^*$ and $\vec{\gamma}_p^*$ are the optimal parameters found by the classical optimizer at step p.

5.1 Experimental results on simulators

For all instances, we utilize the "qasm_simulator" that returns only one state at the end of the circuit. So we approximate the value of F_p, as described in Eq. 6, by running then measuring 256 times the quantum circuit. For the smallest instances, we can also use the "statevector_simulator" that returns the entire superposition state so the value of F_p is computed exactly. The results from "qasm_simulator" are considered as "experimental" while the results from "statevector_simulator" are considered as "theoretical". After doing some experiments at $p \in \{1,2,3,4\}$, we notice similarities between different instances or between the same instance at p and $p+1$. As expected, we observe that

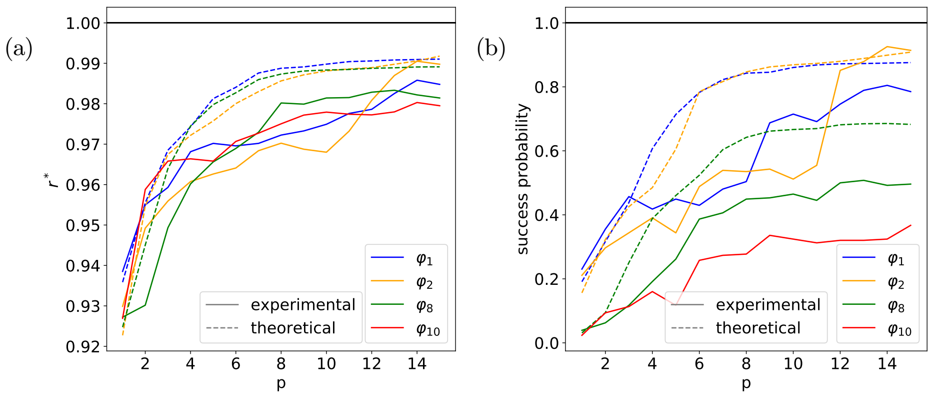

Fig. 4: Evolution of (a) r^* and (b) the success probability both as a function of p, using the first heuristic in Section 5.1, on formulas φ_1 (5 variables and 19 clauses), φ_2 (5 variables and 21 clauses), φ_8 (10 variables and 41 clauses) and φ_{10} (15 variables and 69 clauses).

the variations of F_p seem to correspond with the variations of the success probability. Besides, for all tested p and instances, for all parameters at step p leading to a low cost solution, there exists parameters at step $p+1$ with similar first $2p$ components and also leading to a low cost solution. Finally, for all tested p and instances, a value close to the minimum of F_p is obtained when the values $\beta_{p,i}$ are both small and decreasing and $\gamma_{p,i}$ are both small and increasing with i.
Hence, instead of starting the classical optimizer with several random parameters (which needs a number of initializations growing with p and becoming quickly impracticable), we utilize heuristics to select the initial parameters at step $p+1$ given the optimal parameters found by the optimizer at step p.
The first studied heuristic keeps all components of $\vec{\beta}_p^*$ and $\vec{\gamma}_p^*$ and does a grid search to find the best last component of $\vec{\beta}_{p+1}^{init}$ and $\vec{\gamma}_{p+1}^{init}$, that is:

$$\vec{\beta}_{p+1}^{init}=(\beta_{p,0}^*,\dots,\beta_{p,p-1}^*,\beta) \quad , \quad \vec{\gamma}_{p+1}^{init}=(\gamma_{p,0}^*,\dots,\gamma_{p,p-1}^*,\gamma) \tag{29}$$

with $(\beta,\gamma)\in[0,2\pi[\times[0,2\pi[$.
The performances of QAOA using that strategy for formulas with a different number of variables and clauses are shown in Fig. 4. We clearly see that r^* and the success probability follow a logarithmic evolution with a big increase at the beginning and ends up stabilizing on a plateau. However, for the success probability, the height of the plateau seems to reduce with the increase of the number of variables and clauses. So, that strategy does not seem appropriate to solve our MAX-SAT problem when the number of variables increases, given that we absolutely need an optimal solution to deduce the key. In addition, it is important to notice that we have to execute QAOA using optimization as many times as the number of grid points, for each p.
The second heuristic, called INTERP [20], is based on the regularity in the evolution of the components of certain optimal parameters, as mentioned previously. It

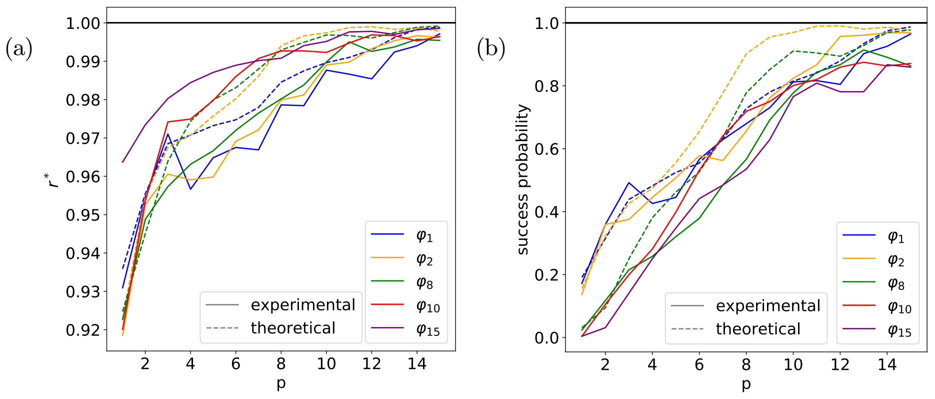

Fig. 5: Evolution of (a) r^* and (b) the success probability both as a function of p, using the INTERP heuristic, on formulas φ_1 (5 variables and 19 clauses), φ_2 (5 variables and 21 clauses), φ_8 (10 variables and 41 clauses), φ_{10} (15 variables and 69 clauses) and φ_{15} (29 variables and 164 clauses).

consists of doing a linear interpolation on the components of $\vec{\beta}_p^*$ and $\vec{\gamma}_p^*$ to get the initial parameters on $p+1$ components. That strategy seems to be efficient for the MAX-CUT problem where the optimal parameters have a similar shape with those of MAX-SAT. The components of the initial parameters are:

$$\beta_{p+1,i}^{init} = \frac{i}{p}\beta_{p,i-1}^* + \frac{p-i}{p}\beta_{p,i}^* \tag{30}$$

$$\gamma_{p+1,i}^{init} = \frac{i}{p}\gamma_{p,i-1}^* + \frac{p-i}{p}\gamma_{p,i}^* \tag{31}$$

with $\beta_{p,-1}^* = \beta_{p,p}^* = 0 = \gamma_{p,-1}^* = \gamma_{p,p}^*$ and starting with $\vec{\beta}_1^{init} = (0.5)$ and $\vec{\gamma}_1^{init} = (0.5)$. By using that heuristic, we observe in Fig. 5(a) that the evolution of r^* runs through three steps. Indeed, we see first a large increase, that becomes moderate before reaching a stabilization near to 1. The height of the plateau does not seem to be affected by the number of variables, contrary to the previous strategy. The general shape of the curves for the success probability is different, as shown in Fig. 5(b). All curves seem to increase in a more or less linear way before reaching a plateau, which is always high, no matter what the number of variables and clauses of the instance are. That strategy seems to be particularly interesting for our MAX-SAT problem, especially as we need to perform QAOA with optimization only once for each p. Its main drawback is that we need to ensure an adequate shape of $\vec{\beta}_p^*$ and $\vec{\gamma}_p^*$ at each p. If one of the $\beta_{p,i}^*$ or $\gamma_{p,i}^*$ is not in the continuity of the others then it could impair the efficiency of the strategy, as shown in Fig. 6. Indeed, during one of the two executions, at $p=10$, β_9 is too high and thus r^* and the success probability collapse from that p.

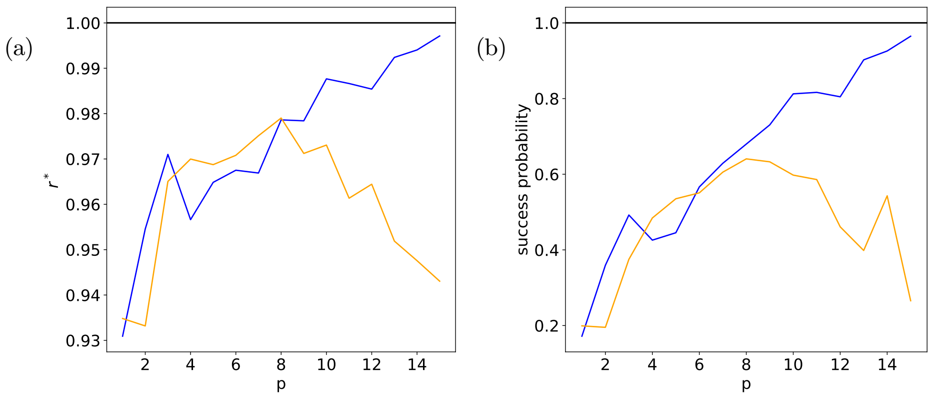

Fig. 6: Comparison of (a) r^* and (b) the success probability both as a function of p between two executions of QAOA using the INTERP heuristic on the same formula φ_1 (5 variables and 19 clauses).

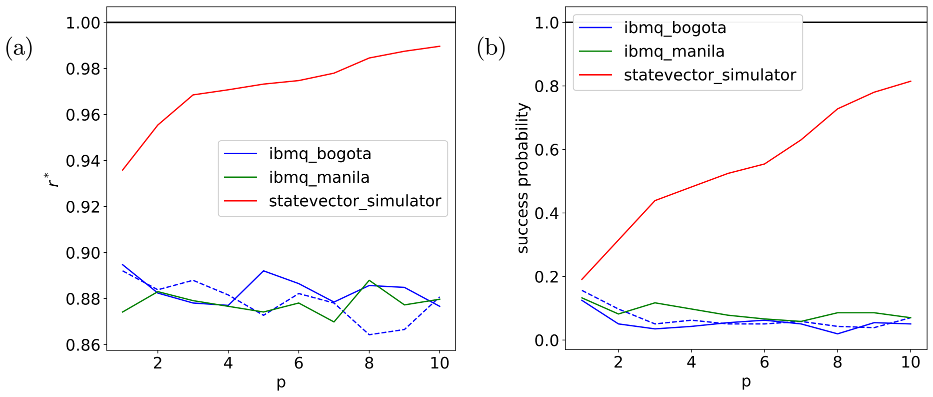

Fig. 7: Comparison of (a) r^* and (b) the success probability both as a function of p between executions of QAOA using INTERP (solid line) or INTERP$_{th}$ (dashed line) heuristics for the same formula on real quantum computers "ibmq_bogota" and "ibmq_manila" (5 qubits and quantum volume of 32) and on a statevector simulator.

5.2 Experimental results on real quantum hardware

We try to solve our problem on the smallest studied instance φ_1 (5 variables with 19 clauses), using the INTERP heuristic until $p=10$, with real quantum computers (5 qubits and quantum volume of 32) from IBM Q Experience. As with simulators, we run the quantum circuit then measure 256 times the final quantum state every time we need to approximate F_p.
Given that the noise from errors and decoherence of the quantum computer can be potentially significant, there is a high probability that the optimal parameters $\vec{\beta}_p^*$ and $\vec{\gamma}_p^*$, found by the optimizer at each p, have not an adequate shape for the INTERP heuristic.

So, we also conduct another experiment where we try to reduce the impact of this INTERP drawback by starting the optimization with the optimal parameters found with the statevector simulator as initial parameters for each p (that we call "INTERP$_{th}$"). The results of all these experiments are shown in Fig. 7. We observe, at $p=1$ and for all tested quantum computers, that the success probability reaches between 13% and 16% instead of the theoretical 20% of the statevector simulator, with optimal parameters close to the expected ones. But when $p \geq 2$, the success probability remains more or less constant and does not exceed 10%. The approximation ratio r^* also remains constant and low (around 0.88) for all p. It can be explained by the fact that the quantum computers utilized to execute QAOA have a small quantum volume (32 at most) and the quantum circuit has a depth of $1+27p$ gates which is already high considering the precision of each individual gate. Notice that the optimal parameters found at each p do not have the expected shape, except for $p=1$.

6 Discussion and perspectives

This paper was a first attempt to estimate the capability of already existing quantum computers on solving a straightforward cryptanalytic problem. First of all, as the number of qubits of NISQ devices that we had access to was very low, we were constrained to utilize algorithms that do not add any auxiliary variables to transform a plaintext-ciphertext pair into a Conjunctive Normal Form formula (CNF). But, currently and as far as we know, such polynomial algorithms do not exist. Moreover, our modelling of the MAX-SAT problem into a quantum circuit is not polynomial as long as we do not know how to decompose the multi-controlled R_ϕ gate polynomially into 1-qubit and 2-qubit basic gates of the quantum hardware.
We have studied the performances of QAOA using the INTERP heuristic and our simulation results show that it seems to be efficient to solve our instances with a success probability that increases linearly until it reaches a high plateau, at least at low p. However, we were not able to confirm that on real quantum hardware given that the resulting quantum circuits were too deep from $p \geq 2$, leading to a hardware-induced noise which is too significant to have reliable results.
Those encouraging results on simulators of QAOA using INTERP are promising but should be treated cautiously. Indeed, all the experiments were realized on few specific small instances of the MAX-SAT problem so we of course cannot affirm that it will work as well on larger instances or on all instances of that problem. Furthermore, the probable link between the number of optimal solutions and r^* or the success probability was not taken into account in the present work.
To summarize, the proposed attack appears not to be feasible, at short and medium term, because of the complexity of the transformation of the plaintext-ciphertext pair into the quantum circuit utilized in QAOA and because of the fact that publicly available quantum computers are not allowing deep enough quantum circuits.
More exploration of QAOA capabilities to solve efficiently some instances of the MAX-SAT problem on a simulator in an attempt at determining which specific instances are "QAOA-friendly" is a relevant perspective. Furthermore, we can conjecture that the INTERP heuristic is sensitive to the noise given that the optimal parameters

found by the optimizer need to have a specific shape, so it would be interesting to design other heuristics more resistant to such noise.

References

1. Ibm q experience. `https://quantum-computing.ibm.com`
2. Qiskit source code. `https://github.com/QISKit/`
3. Sympy source code. `https://github.com/sympy`
4. Bonnetain, X., Hosoyamada, A., Naya-Plasencia, M., Sasaki, Y., Schrottenloher, A.: Quantum attacks without superposition queries: the offline simon's algorithm. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 552–583. Springer (2019)
5. Bonnetain, X., Naya-Plasencia, M.: Hidden shift quantum cryptanalysis and implications. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 560–592. Springer (2018)
6. Bonnetain, X., Naya-Plasencia, M., Schrottenloher, A.: Quantum security analysis of aes. IACR Transactions on Symmetric Cryptology **2019**(2), 55–93 (2019)
7. Farhi, E., Goldstone, J., Gutmann, S.: A quantum approximate optimization algorithm. arXiv preprint arXiv:1411.4028 (2014)
8. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. pp. 212–219 (1996)
9. Heys, H.M.: A tutorial on linear and differential cryptanalysis. Cryptologia **26**(3), 189–221 (2002)
10. Jukna, S., et al.: Boolean function complexity: advances and frontiers, vol. 5. Springer (2012)
11. Kaplan, M., Leurent, G., Leverrier, A., Naya-Plasencia, M.: Quantum differential and linear cryptanalysis. arXiv preprint arXiv:1510.05836 (2015)
12. Kuwakado, H., Morii, M.: Quantum distinguisher between the 3-round feistel cipher and the random permutation. In: 2010 IEEE International Symposium on Information Theory. pp. 2682–2685. IEEE (2010)
13. Kuwakado, H., Morii, M.: Security on the quantum-type even-mansour cipher. In: 2012 International Symposium on Information Theory and its Applications. pp. 312–316. IEEE (2012)
14. Massacci, F., Marraro, L.: Logical cryptanalysis as a sat problem. Journal of Automated Reasoning **24**(1), 165–203 (2000)
15. Powell, M.J.: A direct search optimization method that models the objective and constraint functions by linear interpolation. In: Advances in optimization and numerical analysis, pp. 51–67. Springer (1994)
16. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th annual symposium on foundations of computer science. pp. 124–134. Ieee (1994)
17. Sun, Y., Zhang, J.Y., Byrd, M.S., Wu, L.A.: Adiabatic quantum simulation using trotterization. arXiv preprint arXiv:1805.11568 (2018)
18. Tseitin, G.S.: On the complexity of derivation in propositional calculus. In: Automation of reasoning, pp. 466–483. Springer (1983)
19. Welchman, G.: The Hut Six Story: Breaking the Enigma Codes. M. & M. Baldwin (1997)
20. Zhou, L., Wang, S.T., Choi, S., Pichler, H., Lukin, M.D.: Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. Physical Review X **10**(2), 021067 (2020)