Breaking Block Ciphers with Quantum Computing

Daniele Mammone
daniele.mammone@mail.polimi.it

Gianmarco Naro
gianmarco.naro@mail.polimi.it

Massimo Parisi
massimo.parisi@mail.polimi.it

Abstract: Nowadays lightweight ciphers are becoming widely used and for this reason it is important to analyze and understand their functioning and their weaknesses. In this paper we want to analyze how these ciphers are implemented at quantum level and compare the various types of attacks that can be carried out on them.

Keywords: Quantum Computing, block cipher, SIMON, SPECK, Grover, cryptanalysis

1 Introduction

In the last years IoT devices widely spread in industrial and domestic context. Since these devices now are crucial in managing industrial processes and are now used in managing domestic security more than ever, it's important that messages exchanged between them and processing devices are protected and tamper-proof. On the other hand, since IoT devices should be low energy consumption devices, new lightweight block ciphers have been developed. Here, we'll discuss about their implementation on a quantum system, and their resistance to possible attacks on this class of computing systems.

1.1 Research Contribution

In this paper we want to show various type of attacks on lightweight ciphers in order to break them in a quantum way and the main aim is to compare them. In particular, we'll focus our work on the SIMON and SPECK block ciphers. Both

of them are two lightweight block cipher based on the Feistel scheme. We want to analyze the state of the art of breaking methods for these two ciphers, and how these methods can be run on a quantum computer. We'll learn that, at the time we are writing, we can only make a Key Exhaustive Search. On a quantum computer, that's done by the Grover algorithm, well known for giving a quadratic speedup from $O(2^n)$ to $O(2^{n/2})$, where n is the cardinality of the key space. Our contribution is to compare the complexity, both as time complexity and circuit complexity, of the different approaches of the specific implementation of the attacks. Moreover, we want to analyze if the proposed attacks can effectively run on an actual implementation of a quantum computer, and how much the given solution is reliable. Finally, we want to find if quantum computing offers some type of attack completely different from the already described Exhaustive Search ones, or if we can only boost the time complexity required by the classical ones known in literature.

1.2 Conditions of Research

We are conducing a work based on the state-of-the-art of quantum computing, and our aim is to provide details about some of the main lightweight ciphers used nowadays, describing and comparing the attacks known at the time we are writing this paper, and concluding if they are correct and/or feasible to be executed. Here, we'll consider standardized ISO version of Speck and Simon, respectively defined in ISO/29167-22 and ISO/29167-21. For quantum computers, we are considering the machines provided by IBM-Q platform.

2 Related Works

In this section, we describe the ciphers we're going to analyze and the algorithms that are used to accomplish our goals.

2.1 Modeling of Quantum Adversaries

- **Type 1**: the input data is obtained in a classical way, and than processed with quantum operations.
- **Type 2**: the adversary is able to query a quantum oracle and than obtain the output.

2.2 Different type of crypto attacks

2.2.1 Differential cryptanalysis

It's a chosen plain text attack to symmetric ciphers, so an attacker must be able to generate ciphertext from any given plaintext. The main steps performed by an attacker to do a differential cryptanalysis attack are:

- given a pair (x, y) of text, define an operation to compute the difference $\Delta(x, y)$ between x and y. The XOR $\bigoplus$ one is the most common chosen, but there may be cases where a different approach may lead to better results.
- define a constant difference Δ_x between plaintexts, and collects a certain number of pair of plaintexts (h, k), where $\Delta(h, k) = \Delta_x$.
- given an encryption function called C, for each plaintext pair (h, k) collected in the previous step, calculate the pair $(C(h), C(k))$ and compute $\Delta_y = \Delta(C(h), C(k))$.
- compare each Δ_x with the respective Δ_y, in order to find correlations between the initial difference, and the difference obtained after the encryption.

The goal of differential cryptanalysis is to find if there is a correlation between the difference of two input plaintexts, and the one of the corresponding ciphertexts. In fact, in order to be robust, the output of a symmetric cipher must be indistinguishable from the one of a random number generator. If a correlation is present, it may be used to infer some information on the key, and so break the entire encryption system.

2.2.2 Key Exhaustive Search

It's the naivest attack that can be performed on a symmetric cipher. It's an attack that can be performed in various ways, e.g.

- **Known Plaintext and Ciphertext attacks**: if the pair plaintext and ciphertext is known, it's possible to try each key in the key space to re-encrypt the plaintext, until some key allows to find the initial ciphertext.
- **Known Ciphertext attacks**: given a ciphertext, it's possible to try all the keys in the key space to decrypt the given ciphertext until a meaningful plain text is obtained. This attack is possible only if the cipher is not a perfect one. In fact, in the last case, the cardinality of the key space is greater or equal than the one of plain texts, so it may happen that two different keys lead to different meaningful texts.

2.3 Brief description of SIMON block cipher

SIMON is a new family of lightweight ciphers, which development started in 2011 and was completed in 2013 by NSA. It's a balanced Feistel Cipher based on the ARX (add-rotate-xor) technique. A specific version of SIMON is denoted by the pair $(2n, nm)$, where $2n$ is the block size, and nm is the length of the key. Here, there is a recap of the most common versions used in SIMON.

Block size	**Key size**	**Word size**	**Keywords**	**Rounds**
32	64	16	4	32
48	72,96	24	3,4	36,36
64	96,128	32	3,4	42,44
96	96,144	48	2,3	52,54
128	128,192,256	64	2,3,4	68,69,72

Table 1: SIMON versions.
Each quantity, expect for the rounds, is expressed in bits

2.3.1 Definitions

Here there are some basic definitions, used in SIMON:

- $S^j(x)$ denotes a left circular of j on x
- c is a constant and its value is $2^n - 4$
- k_{i+m} denotes the key schedule at the *i-th iteration*, and is defined as:

$$\begin{cases} c \oplus (z_j)_i \oplus k_i \oplus (I \oplus S^{-1})(S^{-3}k_{i+1}), & \text{m} = 2 \\ c \oplus (z_j)_i \oplus k_i \oplus (I \oplus S^{-1})(S^{-3}k_{i+2}), & \text{m} = 3 \\ c \oplus (z_j)_i \oplus k_i \oplus (I \oplus S^{-1})(S^{-3}k_{i+3} \oplus k_{i+1}), & \text{m} = 4 \end{cases}$$

2.4 Brief description of SPECK block cipher

SPECK is a family of lightweight block ciphers released by the NSA in 2013. It is an add-rotate-xor (ARX) cipher and has been optimized for performance in software implementations.
SPECK also provides different security parameters [Table 2].

Block size	Key size	Word size	Keywords	Rounds
32	64	16	4	22
48	72,96	24	3,4	22,23
64	96,128	32	3,4	26,27
96	96,144	48	2,3	28,29
128	128,192,256	64	2,3,4	32,33,34

Table 2: combinations of different security parameters on SPECK (block/key/word size are expressed on bits)

2.4.1 Definitions

Here there are some basic definitions, used in SPECK:

- $S^j(x)$ denotes a left circular of j on x
- α and β are two variables that change according to the size of the analyzed block
- k denotes the key

$$R_k(x, y) = ((S^{-\alpha}x + y) \oplus k, S^{+\beta}y \oplus (S^{-\alpha}x + y) \oplus k$$

2.5 Grover's Algorithm

All the attacks to the previously described ciphers exploits Grover's algorithm in recovering the secret key used to encrypt the plaintext. Grover's algorithm solves the problem of finding a certain element (the "winner" ω) among a database of N elements. In our case, ω is the encryption key, while the database is the key-space. Grover's algorithm brings a quadratic speedup in solving this types of search problems w.r.t a classic computation approach. Here there is described the version that doesn't make use of any ancillary bit. Grover relies on:

- **Oracle Function**, defined as $U_w |x\rangle = \begin{cases} |x\rangle & \text{if } x \neq \omega \\ -|x\rangle & \text{if } x = \omega \end{cases}$
- **Diffuser Operator**, defined as $2 |s\rangle \langle s| - I$, where s is the uniform superposition of N qubits.

The basic principle of Grover's algorithm is the **Quantum Amplitude Amplification**: from a uniform superposition as input, we obtain an output state where the probability that a $x \neq \omega$ is measured is really low, while the probability that $x = \omega$ is measured is really high.

Given a system of n-qubits, the steps run by Grover's Algorithm are:

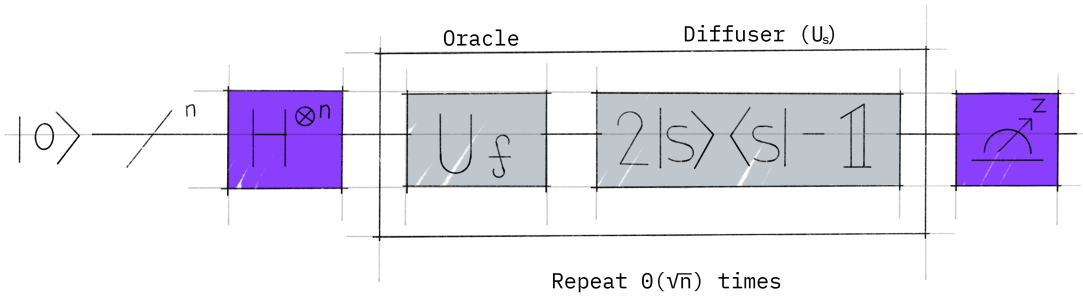

Figure 1: High Level Description of Grover's Circuit from [5]

1. build a uniform superposition over all the states,
   e.g. $|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$
2. for $\prime(\sqrt{n})$ times, run in sequence the Oracle and the Diffuser
3. measure the resulting state

Thank to Quantum Amplitude Amplification, as previously explained, at the end of the procedure, with a high probability, it will be measured $x = \omega$.

3 First definitions on block ciphers

Before going in deep on our work, there is the need to introduce some properties, essential to quantify the security of a block ciphers. Than, we going on analyzing the quantum techniques used for breaking the ciphers of our interest; finally, we'll then compare them in term of efficiency.

– **Secure Block Cipher**: a block cipher is defined as secure if there isn't any available attack to decode a certain ciphertext, but with bruteforce, or guess, of the key.

– **PRP**: a Pseudo Random Permutation is a function that, given a input x, permute x, whose output isn't distinguishable from the one of a random permutation function.

4 Comparison between SIMON and SPECK

4.1 Classical attacks

Before focusing our discussion on SIMON and SPECK from a quantum computing point of view, we'd like to better establish the state of play on classical computers. The best known attack on SIMON in standard attack model exploits linear hull (except for the 32/64 version that uses an integral attack), while on SPECK the most used attack is the differential one. As shown on Table 3 - 4, the performances between the two cipher attacks are quite different; we can notice a general pattern in which SIMON attack performances are better than SPECK ones for the time complexity, while SPECK attack performances are better than SIMON ones for the data complexity.

Variant	**Rounds attacked**	**Time complexity**	**Data complexity**
Simon 128/256	53/72(74%)	2^{148}	$2^{127.6}$
Simon 128/192	51/69(74%)	2^{184}	$2^{127.6}$
Simon 128/128	49/68(72%)	2^{120}	$2^{127.6}$
Simon 96/144	38/54(70%)	2^{136}	$2^{95.2}$
Simon 96/96	37/52(71%)	2^{88}	$2^{95.2}$
Simon 64/128	31/44(70%)	2^{120}	$2^{63.5}$
Simon 64/96	30/42(71%)	2^{88}	$2^{63.5}$
Simon 48/96	25/36(69%)	2^{80}	$2^{47.9}$
Simon 48/72	24/36(67%)	2^{56}	$2^{47.9}$
Simon 32/64	24/32(75%)	2^{63}	2^{32}

Table 3: Best known attack on SIMON (in standard attack model)

Variant	Rounds attacked	Time complexity	Data complexity
Speck 128/256	25/34(74%)	$2^{253.35}$	$2^{125.35}$
Speck 128/192	24/33(73%)	$2^{189.35}$	$2^{125.35}$
Speck 128/128	23/32(72%)	$2^{125.35}$	$2^{125.35}$
Speck 96/144	21/29(72%)	$2^{143.94}$	$2^{95.94}$
Speck 96/96	20/28(71%)	$2^{95.94}$	$2^{95.94}$
Speck 64/128	20/27(74%)	$2^{125.56}$	$2^{61.56}$
Speck 64/96	19/26(73%)	$2^{93.56}$	$2^{61.56}$
Speck 48/96	17/23(74%)	$2^{95.8}$	$2^{47.8}$
Speck 48/72	16/22(73%)	$2^{71.8}$	$2^{47.8}$
Speck 32/64	15/22(68%)	$2^{63.39}$	$2^{31.39}$

Table 4: Best known attack on SPECK (in standard attack model)

4.2 Implementation on a quantum computer

4.2.1 SIMON quantum implementation

Previously, it have been disclosed that Grover's algorithm exploits a **Oracle Function** to find ω among the database of N objects. Here, there will be reported the circuit of SIMON's Oracle Implementation, developed in [2], that's at the basis of both the attacks we'll describe. The round function F is defined as:

$$F(x, y) = (y \oplus S^1(x)S^8(x) \oplus S^2(x) \oplus k, x)$$

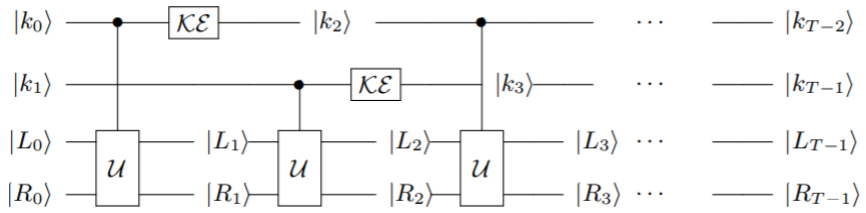

Figure 2: SIMON Implementation with two key words, from [2]

4.2.2 SPECK quantum implementation

In this section, we'll report a description about the curcuital parts of SPECK's Grover oracle, described and developed in [3].

$$R_k(x, y) = ((S^{-\alpha}x + y) \oplus k, S^{+\beta}y \oplus (S^{-\alpha}x + y) \oplus k)$$

The formula shown above represents the round function of SPECK, where S^+ refers to a left rotation while S^- refers to a right rotation. α and β are two variables that change in base of the size of the block and m denotes the number of words of a key, so the round key is composed in this way:

$$K = k_1, k_2, k_3, ..., k_{N-1}$$

In detail, the round function consists of 5 steps:

- **Step 1**: The rotation operation is performed on one of the two part of the plaintext.
- **Step 2**: The addition operation between the two plaintext is performed
- **Step 3**: CNOT gate is used in order to perform the add-round-key
- **Step 4**: The second rotation operation is performed on the plaintext
- **Step 5**: Lastly, the XOR operation between two plaintexts is performed and the result is returned

Algorithm 1 Round function of SPECK block cipher in quantum gate.

Input: $PT = \{PT_0, PT_1\}$, Round key k_i, Round i
Output: $PT = \{PT_0, PT_1\}$
1: $PT_1 \leftarrow PT_1 \ggg \alpha$
2: $PT_1 \leftarrow \text{ADD}(PT_0, PT_1)$
3: $PT_1 \leftarrow \text{CNOT}(k_i, PT_1)$
4: $PT_0 \leftarrow PT_0 \lll \beta$
5: $PT_0 \leftarrow \text{CNOT}(PT_1, PT_0)$
6: **return** PT

Figure 3: Speck Implementation: round function, from [3]

The most important step is the Step 3, because the key value used in the first round is k_0, while, in all rounds except the first, the key used is generated by the **key scheduling**. So, for the next round, the key scheduling is performed so the key is continuously replaced. Hence, **key scheduling** is really important because it allows us to use the key multiple times during different rounds. In fact, we update the qubits value of k_0 used in the first round and use it as the next round key. In this way, k_0 is used as a round key from start to finish, so, qubits assigned to the initial keywords K are reused to the end. We can notice that this process is very convenient because we can recycle the qubits and the encryption phase can be completed with only the initial key words qubits. So, instead of generating all round key K for each round, key scheduling is performed only once and then the round function is performed immediately with the updated k_0 value. In this way, without using the qubits for the new key value, the use of the qubit is minimized. At the end, in each round including encryption and key-scheduling, 4 rotation, 3 XOR and 2 addition operations are performed. So, since the circuit is reversible, the advantage is that the qubit is recycled several times and can be used many times.

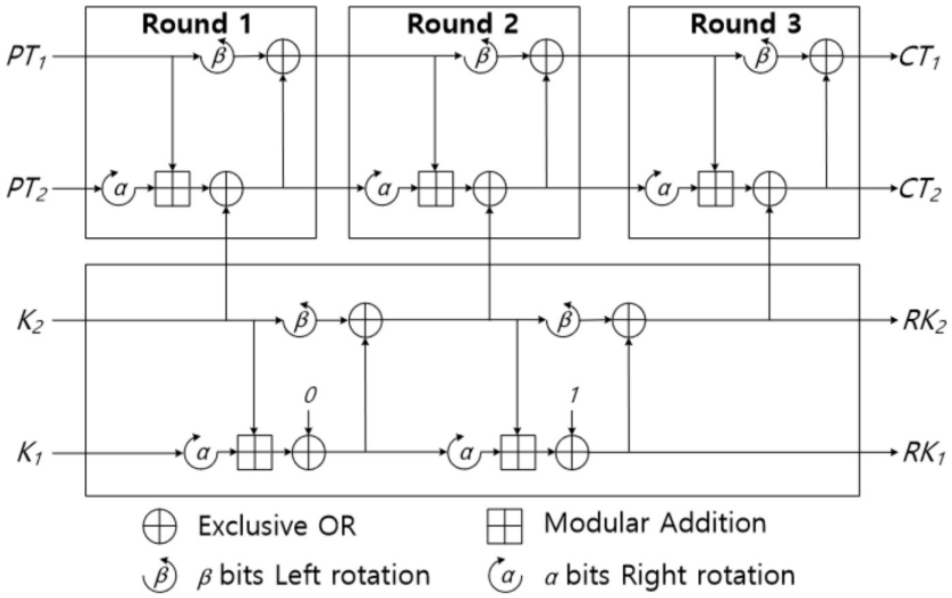

Figure 4: Speck Implementation: key schedule and encryption, from [3]

4.2.3 Differences and similarities

As we can notice from the Table 5, SPECK needs an additional bit compared to SIMON; the main trend (in each $n/2m$ implementation) is that SPECK tends to use:

- $\approx \times 2$ Toffoli gates used by SIMON;
- $\approx \times 1.3$ CNOT gates used by SIMON;
- $\approx \times 0.05$ X gates used by SIMON

Quantum Resources	SPECK32/64	SIMON 32/64	SPECK 64/128	SIMON 64/128
Qubit	97	96	193	192
Toffoli Gates	1.290	512	3.286	1.408
CNOT Gates	3.706	2.816	9.238	7.396
X Gates	42	448	57	1.216

Table 5: Implementation differences between SIMON and SPECK

4.3 Breaking methods on a quantum computer

4.3.1 SIMON breaking method

In this section we focus on the quantum key recovery of SIMON using Grover's algorithm. In all of the attacks described above, we deal with a **Type 2** adversary: in fact, the attacker is able to make quantum queries from classical input, through the oracles introduced to exploit Grover's quadratic speedup.

4.3.1.1 Grover Attack on SIMON

The attack described here is from [2]. It's a **Exhaustive Key Search** of type *chosen plaintext and ciphertext* attack, in order to retrieve the key used to encrypt a given plaintext. It exploits Grover's algorithm to do this, with the consequent quadratic speedup w.r.t. the same attack run on a classical machine. The steps to do this are

- define a circuit that, given the pair plain text and encryption key, generates the corresponding ciphertext (and that's the one we reported previously)
- define a boolean function for Grover's oracle, that uses the previously described circuit, defined as follow

$$\begin{cases} 1 & if \quad \text{E(K, T)} = \text{C} \\ 0 & otherwise \end{cases}$$

where (P, C) is the chosen plain and cipher text pair.

- measure the output. With $p \geq \frac{1}{2}$, we observe the desired key.
- a short note: it may happen that more than one key lead to the same cipher text from a given plain text. To avoid finding a wrong key, we can use more than one (ciphertext, plaintext) pair, in order to find the key that transform each plain text into the desired cipher text.

Comparing to the same attack ran on a classical computer, the only boost a quantum computer can give is the quadratic speedup thank to Grover's algorithm. The authors of [2] tried to run this implementation both on a local simulator and on a real quantum machine hosted on the IBM-Q platform. Through the simulator, they found that their research is correct, since the output of the simulator is in line with the expected theoretical results. Instead, on the real machine, they weren't able to have correct results on a full version of SIMON. In fact, as we analyzed before, there is a huge number of qubits and gates, that bring to some problems:

- not many machines, at this moment, have enough resources to run his code
- even if such a machine is available, the circuit's depth is huge; due to this, there is a constantly increasing noise in the circuit while running the algorithm: this leads to a non reliable output state.

4.3.1.2 Advanced Key Recovery Attack of SIMON

Here, we describe the attack developed in [4]. This attack also relies on Grover's algorithm, but reduces the space of plain texts by doing extra steps to reduce the overall complexity of the algorithm. First of all, some classic computation is done by the following steps, that involves differential cryptanalysis:

1. **Plaintext Collecting**: We pick up $2^{25.7}$ plaintexts with input truncated differential Δ_x and get their corresponding ciphertexts with which we could get $2^{51.5}$ plaintext pairs. Then for every pair, we guess $K^0[13, 15]$ to get (ΔL^2, ΔR^2). We only keep the pairs that satisfy (ΔL^2, ΔR^2) = Δ_{in} and record the corresponding $K^0[13, 15]$. We get $2^{30.5}$ plaintext pairs so that there are 4 correct pairs in expectation that satisfies:

$$E_1(x) + E_1(x \oplus \Delta x) = \Delta_{in}$$

2. **Filtering**: Decrypt each ciphertext pair for one round and only keep the pairs that satisfy a condition, that reduces the number of plaintext pairs to $2^{12.5}$

$$D^1(E(x)) \oplus D^1(E(x \oplus \Delta x)) = (*0000***0*01****, 0*00000**00001**)$$

After that, the quantum computation phase starts. Unlike the previously described attack, that guesses the whole key by using Grover's algorithm, here the key isn't entirely found, but by finding some subparts of it step by step. The phases to do this are:

1. **Partial Key Guessing**: For the remaining $2^{12.5}$ plaintext pairs, we guess the following 25 key bits, denoted by k_1.

$$D_2^K = \{K^{18}, K^{17}[4, 6-9, 13, 15], K^{16}[7] \oplus K^{17}[5], K^{16}[5] \oplus K^{17}[3]\}$$

We maintain an array of counters of size 225 for every key guess and the counter is incremented every time a condition is satisfied.

$$D_{k_1}^4(E(x)) \oplus D_{k_1}^4(E(x \oplus \Delta x)) = \Delta_{out}$$

2. **Exhaustive Search**: We randomly pick two plaintext-ciphertext pairs (m_1,c_1) and (m_2,c_2). We run an exhaustive search on the remaining 39 key bits, denoted by k_2.

$$E_{k_1||k_2}(m_1) = c_1 \wedge E_{k_1||k_2}(m_2) = c_2$$

Even if, w.r.t the previously described attack, some additional classical computations are done to filter the input space and to guess parts of the key, the core of the attack relies on Grover's algorithm. Quantum computations gives the usual quadratic speedup from the same attack ran on a classical computer. The additional classical computations allow to reduce the circuit complexity, even if it's not sufficient to have a correctly running algorithm on a real quantum machine. In fact, the number of gates is still high along with the circuit's depth; as expected, this brings to a high noise in the circuit and to a probably not correct output.

4.3.2 SPECK breaking method

In this section, we present and analyze the type of attacks currently available on SPECK: **exhaustive key search** and **quantum differential attacks**.

4.3.2.1 Exhaustive Key Search

Once we have the quantum implementation of SPECK (showed on the previous chapter), we can exploit Grover's algorithm in order to make an exhaustive key search. The quantum resources needed in order to realize the attack are shown on Table 6

Quantum resources	**SPECK 32/64**	**SPECK 64/128**
Qubit	97	193
Toffoli Gate	1290	3286
CNOT Gate	3706	9238
X Gate	42	57

Table 6: Exhaustive Key Search on SPECK - resources needed, from [3]

All the considerations made for SIMON's counterpart are valid here: we obtain the usual Quadratic Speed, but a complex circuit, not runnable in a reliable way at the time we write.

4.3.2.2 Quantum Differential Attacks

In this section, we want to study the quantum resources required to implement the differential attacks on SPECK, described and completely implemented in [1]. Here, there are described two main types of differential attacks - the **differential distinguisher** and the **last round attack**.

A full implementation of the attacks on SPECK are actually not possible right now, due to the restrictions in the maximum number of qubits that we can have in the state-of-the-art of quantum processors and simulators. So, we construct the circuits for a reduced cipher and show that the implementation of such attacks are theoretically possible. We then compute the resources required for mounting the attack on all variants of SPECK.

A differential distinguisher is an attack tool that works on the top of differential cryptanalisis, but with some differences. In fact, here we try to find a pair of

plaintexts (P, P'), with some additional contraints: we want that $P' = P \oplus \alpha'$ and $C \oplus C' = \beta'$. With these conditions, if the cipher is ideally random, obtaining such a pair would require 2^n trials. So, the attacker will collect 2^p plaintext pairs such that the first condition is true, and with very high probability find at least one pair which satisfies the second condition. In quantum settings, the adversary will apply a Grover's search over the set of all possible messages of size N.
The algorithm will attempt to find a message x such that $E(x) \oplus E(x \oplus \alpha') = \beta'$. Now, since the fraction of messages which satisfies the given differential is 2^{-p}, it is sufficient to make $2^{p/2}$ Grover iterations to obtain the correct messages. Hence, the time complexity for this attack will be $\frac{\pi}{4}2^{p/2}$.

5 Conclusion

After conducting our research and analysis, it can be concluded that, at the actual state of art, there isn't a particular attack that can be ran only on a quantum computer. In fact, at the time we write, SIMON and SPECK appear to not be a broken cipher. It means that the only successfully attack to retrieve an encryption key, is exhaustive key search, or, more commonly, brute force of the key. Even if some filtering, guessing and differential cryptanalysis is done to achieve a reduction in both time and circuital complexities, all the attacks we analyzed in our paper, bruteforces the key. So, actual quantum attacks relies on the same mechanisms exploited by classical attacks in trying to recover a plaintext from a ciphertext, without knowing the encryption key. Is true that these attacks are faster run on a quantum environment: the quadratic speedup given by Grover in search problems makes bruteforce runtime a lot reduced comparing to its classical one, but there is a shortcoming in this: the actual implementation of quantum computers aren't capable of running the previously described circuits with a sufficient reliability in the attack. First, the number of qubits and gates needed to do the attacks, are really huge, and dramatically reduce the number of machines capable to run the algorithms. But, even if with a machine with enough resources, the developers of of original papers found that results aren't reliable, due to the state of the art of physical implementations of qubits and circuits. In fact, due to the high depth of algorithms' circuit, the farther the running goes, more noise is introduced. The results obtained in simulators, anyways, confirms the theories supported by the developers about attacks' correctness. So, in a future this attacks could be really useful (if you are the attacker).

6 References

[1] Ravi Anand, Arpita Maitra, and Sourav Mukhopadhyay. "Evaluation of Quantum Cryptanalysis on SPECK". In: LNCS, volume 12578 (Dec. 2020), pp. 395–416. DOI: 10.1007/978-3-030-65277-7.

[2] Ravi Anand, Arpita Maitra, and Sourav Mukhopadhyay. "Grover on SIMON". In: Quantum Information Processing 19.9 (Sept. 2020). ISSN: 1573-1332. DOI: 10.1007/s11128-020-02844-w. URL: http://dx.doi.org/10.1007/s11128-020-02844-w.

[3] Kyungbae Jang et al. Grover on SPECK: Quantum Resource Estimates. Cryptology ePrint Archive, Report 2020/640. https://eprint.iacr.org/2020/640. 2020.

[4] Hui Liu and Li Yang. Quantum Key Recovery Attack on SIMON Block Cipher. 2021. arXiv: 2012.08321 [quant-ph].

[5] Qiskit Textbook. URL: https://qiskit.org/textbook.