Simon's Algorithm and Symmetric Crypto: Generalizations and Automatized Applications

Federico Canale, Gregor Leander, and Lukas Stennes

Ruhr University Bochum, Bochum, Germany
`firstname.lastname@rub.de`

Abstract. In this paper we deepen our understanding of how to apply Simon's algorithm to break symmetric cryptographic primitives.
On the one hand, we automate the search for new attacks. Using this approach we automatically find the first efficient key-recovery attacks against constructions like 5-round MISTY L-FK or 5-round Feistel-FK (with internal permutation) using Simon's algorithm.
On the other hand, we study generalizations of Simon's algorithm using non-standard Hadamard matrices, with the aim to expand the quantum symmetric cryptanalysis toolkit with properties other than the periods. Our main conclusion here is that none of these generalizations can accomplish that, and we conclude that exploiting non-standard Hadamard matrices with quantum computers to break symmetric primitives will require fundamentally new attacks.

Keywords: Symmetric Cryptanalysis, Simon's algorithm, Bernstein-Vazirani algorithm, Fourier transform, Walsh-Hadamard transform, automatic search, circuits normal form

1 Introduction

Unlike for many public-key schemes, for which the implications of the availability of quantum computers of suitable size were clear from the start, the situation is less well understood for symmetric primitives. The initial general consensus was essentially that only Grover's algorithm, which gives a quadratic speed-up for the problem of exhaustive search [11], is of interest to attack symmetric cryptosystems with quantum resources. This changed after Kuwakado and Morii published theoretical quantum attacks on two classically provable secure constructions, the 3-round Feistel [16] and Even-Mansour [17], using Simon's algorithm.

Simon's algorithm allows to efficiently compute the period of a Boolean function f, when f is accessible as a quantum oracle, and with the given premise that f is a 2-1 function having a unique period. The fact that f has to be accessible by a quantum oracle, often referred to as the Q_2 setting, makes those attacks less relevant in practice for now, but certainly an interesting research topic. Moreover, ideas like the ones presented in [2] show how this class of attacks can have implications in the Q_1 setting, where f can only be queried classically.

Another closely related algorithm to efficiently compute periods uses the Bernstein-Vazirani routine [1], as was already observed in [26]. This routine, see Sect. 2.1 for details, is very similar to the one used in Simon's algorithm. It outputs a vector x that belongs to the support of the Walsh-Hadamard (or Fourier) transformation $\hat{f}$ of f, defined as

$$\hat{f}(x) := \sum_{y}(-1)^{f(y)+\langle x,y\rangle} = (H_0\phi)_x \tag{1}$$

where $(H_0\phi)_x$ is the x.th component of the multiplication vector between the standard Hadamard matrix H_0 and the vector

$$\phi = \left((-1)^{f(0)}, \ldots, (-1)^{f(2^n-1)}\right)^T .$$

Therefore, with enough outputs of this routine we can compute the orthogonal of the support of $\hat{f}$, which is the space of linear structures of f. Since the 0-linear structures of f are exactly the periods, the above algorithm essentially corresponds to Simon's algorithm, but highlights better its relation to the Walsh-Hadamard transformation of f.

The interest sparked by Kuwakado's and Morii's work resulted in the publication of attacks on many other constructions using quantum period finding [14,25,19], a better understanding of how the algorithm works with a relaxed premise on f [14], or without quantum oracle access to f [2,3], in the presence of noise [22], and when trying to minimize the amount of qubits required [21]. The most recent work in this context is by Bonnetain et al. who introduced Quantum Linearization attacks in [4].

The idea behind most of these attacks is to build a function f, based on the target cryptographic scheme E, that has a non-trivial period. So far, by more and more sophisticated and hand-optimized constructions, this class of attacks has made possible to come up with distinguishers on many constructions, like Feistel ciphers up to 6 rounds [16,13,7], MISTY [7,10] or forgery attacks on different kind of authenticated encryptions [14,24,12]. It should be noted that some of these attacks are highly non-trivial and can actually look fairly involved, see e.g. the attack for 6-round Feistel-FK [13]. Searching for new attacks and understanding the security of new constructions has become a cumbersome and error-prone task.

Moreover, those improvements and applications build on exploiting the periodicity of the involved construction only. However, other criteria are of interest as well. As an example, it would be of great value to be able to compute the algebraic degree and related properties of Boolean functions efficiently on a quantum computer. However, the search for efficient quantum algorithms exploiting criteria of Boolean functions other than linear structures has not yet been successful.

One large class of possible algorithms arise naturally from Simon's algorithm by replacing H_0 in Eq. (1) by any other Hadamard matrix H, which corresponds to changing the second Hadamard gate H_0 in the Bernstein-Vazirani routine with a gate that computes the unitary transformation H, as we will see in Sect. 4.

Those algorithms are therefore worth studying. In particular it is of interest to understand if they could lead to new items in the quantum toolbox of symmetric cryptanalysis.

Our Contribution

Our contribution is twofold. First, we simplify the search for new applications of Simon's algorithm and thereby overcome the increasing complexity of the attacks in the literature.

Second, we study the usefulness of the natural extensions of Simon's algorithm mentioned above in the context of symmetric cryptanalysis.

Automatizing. Towards achieving the first goal, introduced in Sect. 3, we propose to automate the search of such functions. More precisely, we present a generic algorithm that aims at finding, given a symmetric cryptographic scheme E, non-trivial periodic functions f, that can then be efficiently computed by a quantum computer.

Our approach here is to represent those functions f dependent on E by a class of circuits. Those circuits can make use of oracle gates for E and potentially further oracle gates for internal parts of the scheme E. We then automatically examine all circuits up to a certain number of gates and test each of them for periodicity, by instantiating the respective function on small dimensions. Of course, this means that many useless circuits, as well as many useless periods, are generated. The main technical contribution and work is aimed at addressing this problem, and keeping the process efficient by pruning the search tree. We discuss the details in Sect. 5.

As a proof of concept, we rediscover many of the attacks already known. Moreover, and more importantly, our algorithm automatically leads to new attacks. Indeed, it finds new periodic functions for 4-round Feistel-FK and 5-round Feistel-FK with internal permutation, as well as 4-round MISTY R-FK and 5-round MISTY L-FK. Those lead to the first known key-recovery attacks on these constructions in polynomial time. Further, we show that our approach is also applicable in the Grover-Meets-Simon case. We give new attacks for the permutation-based Encrypted Davies-Meyer and sum of key alternating ciphers constructions.

Generalizations. Regarding the generalization of Simon's algorithm we argue in Sect. 4 that none of those algorithms is likely to be helpful for speeding-up known attacks on quantum computers.

We do so by arguing that none of the new algorithms arising from this generalizaion of Simon's algorithm allow the computation of any property of Boolean functions that is invariant under linear equivalence. Since most of the properties used in cryptography, like the algebraic degree, the balancedeness, the nonlinearity (order) or differential uniformity of f are indeed linear invariant, this brings us to conclude that any property related to this is unlikely to be of relevance for existing attack vectors.

While this result might not be surprising, it (i) sheds some light on the lack of alternative quantum algorithms and (ii) might be of independent interest. Indeed, it is technically based on a new characterization of the standard Hadamard matrix, proved using a general result on the structure of the automorphism group of the general linear group over $\mathbb{F}_2$ due to Dieudonné [8].

Outline

We explain preliminaries in Sect. 2, followed by the new attacks in Sect. 3. Sect. 4 presents our main result regarding the generalization of Simon's algorithm. Sect. 5 and 6 respectively give more details about our automated search and the proof of our result on generalized Simon. We conclude our results in Sect. 7.

2 Preliminaries

Let $\mathbb{F}_2$ be the finite field with two elements and $\mathbb{F}_2^n$ a vector space of dimension n over $\mathbb{F}_2$. We will often identify the elements of $\mathbb{F}_2^n$ with the integers $\{0, \ldots, 2^n - 1\}$, without making it explicit unless required by the context. We denote by $\mathrm{GL}(n, \mathbb{F}_2)$ the general linear group of invertible matrices of order n over $\mathbb{F}_2$. We will ignore normalization factors for quantum states in order to simplify notation and concepts. We denote the rising factorial $x \cdot (x+1) \cdots (x+n-1)$ as $x^{\bar{n}}$.

2.1 Some Generalities on Boolean Functions

A Boolean function is any function $f : \mathbb{F}_2^n \to \mathbb{F}_2$. We denote the set of Boolean functions by $\mathcal{B}_n$. We may identify the set of Boolean functions over $\mathbb{F}_2^n$ with the set of polynomials $\mathbb{F}_2[X_1, \ldots, X_n]/(X_1^2, \ldots, X_n^2)$, in which case a Boolean function f can be written as

$$f(X) = \sum_{u \in \mathbb{F}_2^n} a_u X^u \tag{2}$$

for some $a_u \in \mathbb{F}_2$ and where we have denoted by X^u the monomial $X_1^{u_1} \cdots X_n^{u_n}$. Eq. (2) is also known as the Algebraic Normal Form of f.

For a given Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$, the Walsh-Hadamard transform of f is defined as the function $\hat{f} : \mathbb{F}_2^n \to \mathbb{Z}$ such that

$$\hat{f}(\alpha) = \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x) + \langle \alpha, x \rangle},$$

where we indicate with $\langle \cdot, \cdot \rangle$ the scalar product over $\mathbb{F}_2^n$ defined by $\langle x, y \rangle = x_1 y_1 + \ldots + x_n y_n$ for any $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$.

Finally, let us denote the support of $\hat{f}$ by $\mathrm{supp}(\widehat{f}) = \{\alpha \in \mathbb{F}_2^n : \hat{f}(\alpha) \neq 0\}$.

From the perspective of the Fourier transformation, Simon's algorithm can be interpreted as using the following relation between the Walsh-Hadamard transformation and the linear structures of a function f.

Theorem 1 (Proposition 29 in [5]). *Let $e \neq 0$ be an element of $\mathbb{F}_2^n$. We have that*

$$f(x) + f(x+e) = 0 \quad \forall x \in \mathbb{F}_2^n$$

(resp. $f(x) + f(x+e) = 1$) if and only if

$$\{0, e\} \subset supp(\widehat{f})^{\perp}$$

(resp. $supp(\widehat{f}) \cap \{0, e\}^{\perp}$ is the empty set).

We refer to [5] for more background on Boolean functions and in particular for a discussion of cryptographic criteria. We here limit ourselves to highlighting the fact that most of those criteria are invariant under linear equivalence. That is, given two Boolean functions $f, g \in \mathcal{B}_n$ such that for all $x \in \mathbb{F}_2^n$ it holds that $f(x) = g(L(x))$, where L is an isomorphism, then f and g behave identical with respect to the main criteria. Those criteria that are linear invariant include the algebraic degree, the non-linearity, the differential uniformity, and the balancedness of f and g.

2.2 Quantum Period Finding and the Hadamard Gate

We will briefly recall the Hadamard gate which will be in many ways the focus of the second part of the paper. To this end, we simply remind that the state of n qubits can be represented as a unitary vector in $\mathbb{C}^{2^n}$ and that a quantum transformation is represented as a unitary transformation. We indicate a basis for $\mathbb{C}^{2^n}$ as $|i\rangle$, where i is an integer between 0 and $2^n - 1$ written in its binary representation. No further knowledge about quantum computation is necessary for the purpose of this paper, and we refer to [23] for details.

The Hadamard gate on one qubit is such that

$$\begin{aligned} |0\rangle &\mapsto (|0\rangle + |1\rangle) \\ |1\rangle &\mapsto (|0\rangle - |1\rangle). \end{aligned}$$

In other words, it is represented by the matrix

$$H_0 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

Applied to an n-qubit vector, it is represented by the matrix $H_0^{\otimes n}$, given by

$$\left(H_0^{\otimes n}\right)_{x,y} = (-1)^{\langle x,y \rangle} \text{ for all } 0 \leq x, y \leq 2^n - 1,$$

which will act on the basis vector $|x\rangle$ as

$$|x\rangle \mapsto \sum_{y=0}^{2^n-1} (-1)^{\langle x,y \rangle} |y\rangle .$$

For the rest of the paper, we will indicate the above transformation simply by H_0, regardless of the dimension n, unless there is possibility for ambiguity.

Let us now consider the quantum implementation of a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$

$$U_f : |x\rangle\,|y\rangle \mapsto |x\rangle\,|y \oplus f(x)\rangle \text{ for any } x \in \mathbb{F}_2^n \text{ and } y \in \mathbb{F}_2.$$

The unitary transforms just presented are the building blocks for the quantum routine used in Bernstein-Vazirani algorithm (as well as Simon's), which allow to efficiently compute the support of the Walsh-Hadamard transformation, since the state before measurement is

$$\sum_{y=0}^{2^n-1}\left(\sum_{x=0}^{2^n-1}(-1)^{f(x)+\langle x,y\rangle}\right)|y\rangle\,|-\rangle = \sum_{y=0}^{2^n-1}\hat{f}(y)\,|y\rangle\,|-\rangle \qquad (3)$$

where $|-\rangle = |0\rangle - |1\rangle$. Therefore, measuring the first n-qubit register will yield y with probability proportional to $\hat{f}(y)^2$. With a sufficient number of y, it is therefore possible to compute the space generated by the support of the Walsh-Hadamard transform of a random function f, which is exactly the orthogonal of the space of linear structures for f due to Theorem 1.

From this, it follows that this routine can be used to efficiently compute periods of a random Boolean function, as was already noted in [27]. More details about this will be given in Sect. 4.1. Note that this can be extended to find linear structures in the case where the codomain of f has dimension larger than 1 by considering each component separately, as it is discussed in [27]. A brief description of Simon's algorithm and a comparison with Bernstein-Vazirani can be found in Appendix A.

In this work, we are mostly interested in finding new meaningful cryptographic periods (Sect. 3), and less interested in how they are computed. Therefore, we always assume that the functions we deal with are random enough to make the quantum period finding efficient. For a more thorough discussion about the precise conditions that a function has to satisfy in order for this to be possible, we refer to [2] and [27].

Furthermore, the way of efficiently computing the Walsh-Hadamard transform of f thanks to Eq. (3) leads to the question of whether it is possible to generalize the construction in order to efficiently compute other kinds of transform of f that could possibly capture different properties of f (Sect. 6.1). For this, we recall the definition of the class of Hadamard matrices.

Definition 1. *Let $H \in \mathbb{R}^{N\times N}$ be a matrix. We say that H is a Hadamard matrix of order N if*

$$(H)_{x,y} \in \{-1, 1\} \text{ for all } 1 \le x, y \le N \text{ and } H^T H = N I_N,$$

where I_N is the identity matrix and $N \in \mathbb{N}$.

2.3 Description of Feistel and MISTY

In this work, we present new attacks on the Feistel and MISTY construction using quantum period finding. Therefore, we now briefly describe these families of ciphers.

Feistel The Feistel cipher, also known as Luby–Rackoff cipher, is a simple way of turning random functions into a pseudorandom permutation. To do so, given r round functions $F_0, F_1, \ldots, F_{r-1} : \mathbb{F}_2^n \to \mathbb{F}_2^n$ and input $(L_0, R_0) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$, one computes the output (L_r, R_r) by computing

$$L_{i+1} = R_i \qquad R_{i+1} = F_i(R_i) \oplus L_i$$

for $i = 0, 1, \ldots, r-1$. If we replace the secret random permutations with one public permutation F and round keys k_i which we xor to the output of F, we obtain the so called Feistel-FK construction which we depict in Fig. 1.

MISTY Matsui [20] proposed the MISTY structure as a way to design block ciphers with provable security against differential and linear cryptanalysis. In some ways, MISTY is similar to Feistel. That is, there is a left and a right part that are altered and swap in every round. But in contrast to Feistel, MISTY decryption requires the inverse of the round function. Furthermore, there is a left and a right version of MISTY, which we denote by MISTY L and MISTY R resp. Let (L_i, R_i) be the input to the i-th round of MISTY L-F with round function F_i. Then the output (L_{i+1}, R_{i+1}) is defined by

$$L_{i+1} = R_i \qquad R_{i+1} = F_i(L_i) \oplus R_i.$$

For MISTY R-F we have

$$L_{i+1} = R_i \oplus F_i(L_i) \qquad R_{i+1} = F_i(L_i).$$

Analogous to Feistel, in practice we might replace the secret round functions F_i by one public permutation F in combination with round keys k_i. We can inject k_i either before we apply F or afterwards. We call the former KF and the latter FK. In this work, we study 5-round MISTY L-FK and 4-round MISTY R-FK. For both, one round is depicted in Fig. 1.

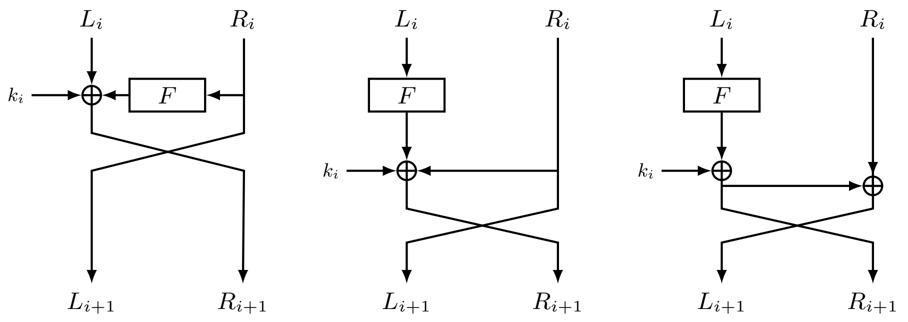


Fig. 1. Feistel and MISTY constructions.

3 Finding New Applications

In this section, we first briefly describe how new applications of Simon's algorithm can be found automatically. After that, we present new attacks against MISTY and Feistel. We found these with a straightforward `sage` implementation of our algorithm.[1] A more detailed analysis, especially regarding the efficiency of our approach, is presented in Sect. 5.

3.1 The Idea: Finding Circuits

One goal of this work is to automate the step of searching a suitable periodic function f for some cryptographic construction E so that f leads to an attack on E based on Simon's algorithm. In a nutshell, the idea for this is to simply test all sensible functions that depend on E, which we denote informally by $\mathscr{F}^E$, for periodicity. Although this idea is simple, it raises two questions. First, what functions $\mathscr{F}^E$ do we need to test and second, how can we efficiently check whether f is periodic or not? The latter is easy to answer. We just instantiate E with a small input size s.t. we can evaluate f on all possible inputs. The former is more profound. Our approach is to represent $\mathscr{F}^E$ by a class of circuits. That is, we fix the inputs of f and the operations f consists of. We call these gate functions. Most importantly, the set of gate functions contains oracle gates for E. Then we can automatically examine all functions that consists of one gate, all functions that consists of two gates and so on. E.g., consider the quantum attack on 3-round Feistel from [16] which is based on the function

$$f(x,b) = ENC_L(x,\alpha_b) \oplus \alpha_{\bar{b}} = F_1(x \oplus F_0(\alpha_b)) \oplus \alpha_b \oplus \alpha_{\bar{b}}$$

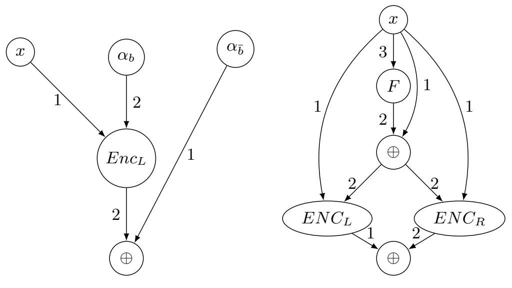
with period $s = (F_0(\alpha_0) \oplus F_0(\alpha_1))||1$. Here, $\alpha_0, \alpha_1 \in \mathbb{F}_2^n$ are arbitrary distinct constants. Now consider Fig. 2a which shows a circuit C_f that represents f. To find this, we start with a circuit that only contains the input nodes $X = \{x, \alpha_b, \alpha_{\bar{b}}\}$. Then, we investigate all circuits that consist of one additional node that represents a function from $\mathscr{G} = \{\oplus, ENC_L, ENC_R\}$ where ENC_L and ENC_R correspond to the left and right part of the output of the 3-round Feistel cipher. After that, we investigate all the circuits that consist of two additional nodes, and thereby we encounter C_f with period s.

3.2 New Attacks on MISTY and Feistel

We now present new key-recovery attacks on MISTY and Feistel. The corresponding circuits that were found by our implementation are depicted in Fig. 2b for the first attack and in Fig. 9 for the other attacks. The proofs of periodicity of the used functions are also presented in Appendix B.

Recall that for the sake of simplicity, we assume that Simon's algorithm will always yield the correct period. For rigorous proofs of correctness the strategy in

[1] see `https://www.doi.org/10.5281/zenodo.6623768`



(a) Circuit C_f. (b) Circuit C_{f_1}.

Fig. 2. Circuits for attack against 3-round Feistel cipher from [16] and against 4-round MISTY R-FK from Sect. 3.2.

[14] can be adopted to argue that either Simon's promise is sufficiently fulfilled, or differential-based attacks exist.

Because here the focus is to automatically find periodic functions, we leave this and other details such as how one can implement the functions on a quantum computer for future work.

Key-recovery attack on 4-round MISTY R-FK For 4-round MISTY R-FK, Cui et al. [7] mention a distinguishing attack but they do not give it explicitly. Our automated search yields

$$f_1(x) = ENC_L(x, x \oplus F(x)) \oplus ENC_R(x, x \oplus F(x)) \tag{4}$$

with period $s_1 = k_0$. This immediately leads to a full key-recovery attack. After we have recovered k_0 using Simon's algorithm, we can simply uncompute the first round before we make a query and extend the output of the oracle by an additional round, i.e., we use

$$\begin{aligned} L_0' &= F^{-1}(R_0 \oplus k_0) \\ R_0' &= L_0 \oplus R_0 \\ ENC_R'(L_0, R_0) &= F(ENC_L(L_0', R_0')) \oplus k_4 \\ ENC_L'(L_0, R_0) &= ENC_R(L_0', R_0') \oplus ENC_R'(L_0, R_0) \end{aligned}$$

instead of ENC_L and ENC_R to recover k_1. Here, k_4 is of course not a secret key, but a random value chosen by us to simulate an additional round. We repeat this procedure to recover k_2 and k_3. Notice that there may be more efficient ways to recover the other round keys once k_0 is uncovered. But for sake of simplicity, we will stick to this generic argument.

Key-recovery attack on 5-round MISTY L-FK Cui et al. [7] also give a distinguisher for 5-round MISTY L-FK. Again, we improve on this and give a key-recovery attack based on the function

$$f_2(x) = ENC_L(F^{-1}(F^{-1}(x) \oplus x), F^{-1}(x)) \oplus F(x). \tag{5}$$

The corresponding period is $s_2 = k_0$. Therefore, we can use the same idea as for 4-round MISTY R-FK to recover all keys.

Key-recovery attack on 4-round Feistel-FK For 4-round Feistel-FK, our automated search algorithm yields

$$f_3(x) = ENC_R(x, \alpha) \oplus F(ENC_L(x, \alpha)) \oplus F(x) \tag{6}$$

with period $s_3 = k_0 \oplus F(\alpha)$. Here, $\alpha \in \mathbb{F}_2^n$ is an arbitrary constant. Again, we can use the same idea as for 4-round MISTY R-FK to recover all keys.

Key recovery attack on 5-round Feistel-FK This time, we use a slightly different idea for the search. That is, once we have discovered a periodic function whose period bears some information on the keys, we use this period as a constant in our next search. Thereby, we find the following functions.

$$f_4(x) = ENC_R(F(x), x) \oplus F(ENC_L(F(x), x)) \oplus F(x) \tag{7}$$

$$f_5(x) = ENC_L(F(F^{-1}(x) \oplus s_4), F^{-1}(x) \oplus s_4) \oplus F(x) \oplus F^{-1}(x) \tag{8}$$

The corresponding periods are $s_4 = F(k_0) \oplus k_1$ and $s_5 = k_0 \oplus k_2$. Notice that f_5 uses the inverse of the internal function F, and thus we have to assume that F is bijective. It is well-known that this is not necessary for Feistel networks, but we nonetheless believe that our attack is of interest.

The first step of our attack, once again, is to use Simon's algorithm to find s_4 and s_5. After that, we use a classical query to obtain an auxiliary value

$$h = ENC_R(F(s_4), s_4) \oplus F(ENC_L(F(s_4), s_4)) \oplus F(0) = k_0 \oplus k_2 \oplus k_4.$$

Now we can restore $k_4 = h \oplus s_5$. Thereby, we have learned the last round key and can continue similarly to the attacks where we learned the first round key, i.e., we add a round before we make a query and uncompute the last round afterward to find k_3. We repeat this procedure to learn k_2, k_1 and k_0.

3.3 Grover-Meets-Simon: New Attacks on pEDM and SoKAC

Our approach cannot only be used to find quantum attacks based on Simon's algorithm but leads also to attacks based on the Grover-Meets-Simon algorithm [19]. In those attacks, the attacker first makes a guess u for part of the key, say k_1, (the Grover part). Only for the correct guess, i.e., if $u = k_1$, the attacker gets a periodic function, which is then the detected with Simon's algorithm.

To find such, we add a second input node u and then check periodicity for all values of u separately.

Key recovery attack on pEDM In [9], the authors introduced the permutation-based Encrypted Davies-Meyer construction. For a random permutation P and secret keys k_0 and k_1, we have

$$pEDM(x) = P(P(x \oplus k_0) \oplus (x \oplus k_0) \oplus k_1) \oplus k_0.$$

Now, it is easy to verify that

$$f(u, x) = pEDM(x) \oplus P(P(x) \oplus x \oplus u)$$

has period k_0 for $u = k_1$ and thus both keys can be recovered using the Grover-Meets-Simon attack.

Key recovery attack on $SoKAC$ Similar to the attack on pEDM, for the sum of key alternating ciphers [6] with two permutations P_1, P_2 and two keys k_0, k_1, we have

$$SoKAC(x) = P_2(P_1(x \oplus k_0) \oplus k_1) \oplus k_0 \oplus P_1(x \oplus k_0)$$

and

$$f(u, x) = SoKAC(x) \oplus P_2(P_1(x) \oplus u) \oplus P_1(x)$$

has period k_0 for $u = k_1$ and thus both keys can be recovered using the Grover-Meets-Simon attack.

3.4 Applications to Classical Cryptanalysis

Obviously, periodic functions as presented in Sect. 3.2 do not only lead to quantum attacks based on Simon's algorithm, but also to classical birthday-bound attacks. Similarly, attacks based on Grover-Meets-Simon can be converted to classical attacks too. Even more practical classical attacks, polynomial-time attacks, are special cases where one has to find a constant function, i.e. a function where all elements are periods. However, during our work we have not encountered any such function, which is not surprising at least for the constructions with classical security proofs. Nevertheless, we believe that our tool is a valuable addition to the toolbox of symmetric cryptanalysis. It can be applied to many more constructions in the future. In particularly, it might be of use already in the actual design process of a new construction as a tool to quickly rule out insecure approaches.

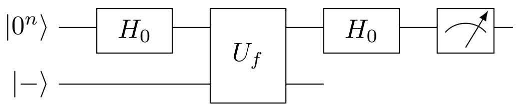
4 Generalizing Simon's Algorithm

There are many properties of Boolean functions that have been found to be meaningful from a cryptographic point of view, but most of them have not yet been found to be significantly more easy to compute with quantum resources.

In this section we discuss how we attempted to address this problem by investigating whether it is possible to generalize Simon's algorithm, or rather the Bernstein-Vazirani algorithm, in order to be able to compute other potentially interesting properties of Boolean functions, focusing on linear invariant properties: we conclude that the generalization we propose does not allow to do that, by proving that the corresponding generalization of the Walsh-Hadamard transform is not linear invariant. A more formal discussion of the precise statements and proofs will be given in Sect. 6. Note that we focus on the case of Walsh-Hadamard or Fourier-transformations over $\mathbb{F}_2$ only.

4.1 The Idea: Using Non-standard Hadamard Matrices

In order to understand our idea for generalizing Simon's algorithm, we consider the following small variation of the quantum routine used in Simon's algorithm, due to Bernstein-Vazirani [1], where H_0 corresponds to the standard Hadamard transform $(H_0)_{x,y} = (-1)^{\langle x,y \rangle}$.



This circuit outputs a vector x that belongs to the support of the Walsh-Hadamard (or Fourier) transformation $\hat{f}$ of f. This is due to the fact that the state of the n-qubit register before the final application of H_0, ignoring the constant that makes the state unitary, is represented by

$$\phi := \left((-1)^{f(0)}, \ldots, (-1)^{f(2^n-1)}\right)^T,$$

and applying H_0 to ϕ yields a vector whose components are

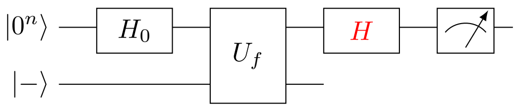
$$(H_0\phi)_x = \sum_y (-1)^{f(y)+\langle x,y \rangle} = \hat{f}(x). \tag{9}$$

Therefore, since a measurement of such a state results in a x such that $\hat{f}(x) \neq 0$, with enough measurements of the above circuit, we can compute the space generated by the support of $\hat{f}$ and recover the space of the linear structures of f thanks to Theorem 1. More precisely, the measurement yields x with probability $\frac{\hat{f}(x)^2}{2^{2n}}$. However, for the purpose of this work, we are simply interested in the fact that the measurement of the state at the end of the circuit is a vector x such that $\hat{f}(x) \neq 0$. For a more thorough analysis of the algorithm we refer to [26].

The idea behind our generalization is to consider Hadamard matrices other than the standard one. More precisely, in this paper we will study the class of transforms given by

$$\widehat{f}^H(x) := \sum_y (-1)^{f(y)+g(x,y)} = (H\phi)_x$$

with H being a Hadamard matrix $(H)_{x,y} = (-1)^{g(x,y)}$. This corresponds to studying the following generalization of the previous circuit, which we call `Simon`(H).



This circuit outputs x such that $\widehat{f}^H(x) \neq 0$ and could be used to compute efficiently (under the hypothesis that H can also be efficiently implemented) information about $\widehat{f}^H(x)$. In particular, echoing Simon's algorithm, the space generated by its support could be computed and could potentially correspond to a (cryptographically) relevant property of f (as is the case for $H = H_0$).

4.2 Results

There are several interesting discussions raising from this perspective. First, as the number of Hadamard matrices to consider for $\widehat{f}^H(x)$ (or equivalently $\texttt{Simon}(H)$) is huge even for small dimensions, reducing their amount using a suitable notion of equivalence is of interest. Second, and most importantly, the question is if computing those transforms can be used to compute anything of (cryptographic) relevance.

In this paper we discuss those points. We show in Sect. 6.1 that the usual notion of equivalence for Hadamard matrices[2] nicely translates into equivalent quantum algorithms. More precisely, given two equivalent Hadamard matrices H and H', the $\texttt{Simon}(H')$ circuit can be turned into the $\texttt{Simon}(H)$ circuit by classical pre- and post-processing (Proposition 1).

Regarding the question of whether anything relevant can be computed with Hadamard matrices that are not equivalent to the standard Hadamard (and therefore not to the standard Fourier-transform), we argue that this is not the case for any property that is invariant under linear equivalence. This brings us to conclude that any property related to this is unlikely to be of relevance for existing attack vectors (since most of the properties used in cryptography, like the algebraic degree, the balancedeness, the nonlinearity (order) of f are indeed linear invariant).

Indeed, we observe that the fact that $\texttt{Simon}(H_0)$ allows to compute such a criterion, i.e. the existence of linear structures, can be seen as a consequence of the fact that, for any isomorphism A

$$\widehat{f \circ A} = \widehat{f} \circ B,$$

where $B^{-1} = A^T$.

Our main result, stated in the next theorem and proved in Sect. 6.2, is that this property already classifies the standard Hadamard transformation.

Theorem 2. *If for any $A \in \mathrm{GL}(n, \mathbb{F}_2)$ there exists $B \in \mathrm{GL}(n, \mathbb{F}_2)$ such that for all Boolean functions f we have*

$$\widehat{f \circ A}^H = \widehat{f}^H \circ B$$

then H is equivalent to H_0.

This result implies that, if H is not equivalent to H_0, then the transform $\widehat{f}^H$ cannot capture any linear invariant property of f, because the transform itself does not vary linearly.

Note that there is a technical caveat: indeed, we cannot strictly exclude that the circuit $\texttt{Simon}(H)$ could be used to compute properties that are linear invariant even if H is not equivalent to H_0. For example, for the space V_f *generated* by

[2] A list of possible representatives of Hadamard matrices up to dimension 28 is known and can be found here: `http://neilsloane.com/hadamard/`

$\mathrm{supp}(\widehat{f}^H)$ Theorem 2 does not directly imply that V_f is also not linear invariant. However, such a behaviour would be very surprising.

Therefore we conclude that, unless H is equivalent to H_0, it is unlikely that the algorithm given by `Simon`(H) could compute any property of f that is linear invariant.

5 Constructing Circuits: Efficiency Considerations

We already described the basic idea of finding periodic functions in Sect. 3.1. In this section, we explain our approach in more detail. We define a notion of circuits, establish a normal form for these, and work out how many circuits there are. Furthermore, we describe how we exclude useless and enumerate good circuits, and also how we filter out trivial periods. After that, we use our algorithm to rediscover known results automatically. Thereby, we do not only demonstrate that our approach is indeed sensible, but we also gather valuable experience that we reuse when we look for novel attacks.

5.1 Circuits

Different variants of circuits are used in a variety of contexts. Our idea of a circuit is mostly inspired by Boolean circuits used in computational complexity theory and related fields. Since we are mainly interested in a practical way of automatically generating circuits, we choose to define circuits from the ground up. Thereby, we can make sure that our formalism and our implementation are well-matched.

Definition 2 (Syntax of Circuits). *A circuit $C = (D, X, \mathscr{G}, v_{out}, n)$ is defined by*

- *a directed acyclic graph $D = (V, E)$ with labeled nodes and edges,*
- *a set of input nodes $X \subset V$,*
- *a set of gate functions $\mathscr{G} = \{G_0, G_1, \ldots, G_{g-1}\}$ where $G_i : \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2^n$,*
- *an output node $v_{out} \in V$.*

We assume $V = \{0, 1, \ldots, p-1\}$ and $X = \{0, 1, \ldots, q-1\}$ for some $p, q \in \mathbb{N}$. Nodes $v \in V \setminus X$ are labeled with gate functions. We denote the gate function associated with v by v_G. For all $v \in V \setminus X$ there is a left and a right predecessor of v which we denote by v_L and v_R resp. It must hold that $v_L < v$ and $v_R < v$. The edge (v_L, v) is labeled with 1 and (v_R, v) is labeled with 2. If $v_L = v_R$, then (v_L, v) is labeled with 3. There are no other edges. We denote $|V \setminus X|$ as the size of C.

Throughout this work, we will omit parts of this formal definition if they are clear from the context. We want to stress that gate functions only take two inputs and have one output.

Definition 3 (Semantics of Circuits). *For a circuit $C = (D, X, \mathscr{G}, v_{out}, n)$ and an input assignment $A : X \to \mathbb{F}_2^n$, the output of a node v is defined as*

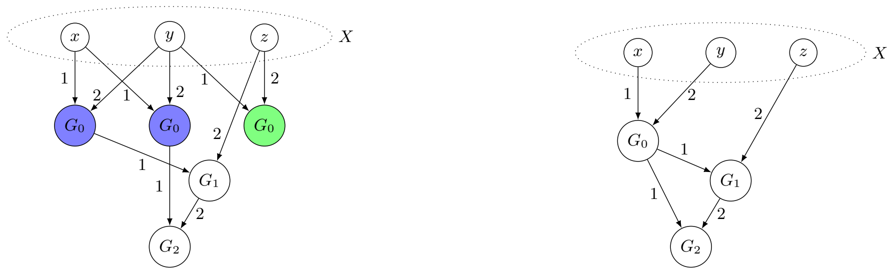
$$out(v) = \begin{cases} A(v) & \text{if } v \in X \\ v_G(out(v_L), out(v_R)) & \text{if } v \in V \setminus X. \end{cases}$$

The output of C is set to $out(v_{out})$. We denote the function described by C as C_f.

Of course, in our algorithm, we want to avoid checking the same function multiple times. Therefore, we want to establish a normal form for circuits next. To do so, we first define an equivalence relation in a straightforward way. Then we illustrate a non-normal circuit in Example 1 and formalize the concept of equivalent nodes, loose ends and ordered circuits. Finally, we define the circuit normal form in Definition 9.

Definition 4 (Equivalence of Circuits). *Two circuits C and C' are said to be equivalent if they correspond to the same function, i.e., if $C_f = C'_f$. We denote this as $C \sim C'$.*

Example 1. The circuit in Fig. 3a meets Definition 2 and by Definition 3 computes the function $f(x, y, z) = G_2(G_0(x, y), G_1(G_0(x, y), z))$. However, notice that there are two nodes (marked in blue) that compute the same intermediate value and another node (marked in green) that is never used. Hence, this is not a natural way of representing f. In contrast, the circuit in Fig. 3b, which also computes f, has no useless nodes and therefore, is a more sensible way to represent f.



(a) Circuit in non-normal form. (b) Circuit in normal form.

Fig. 3. A circuit in (non-)normal form.

Definition 5 (Equivalence of Nodes). *All nodes in a circuit C are equivalent to itself. For all $x \in X$ there is no other node than x that is equivalent to x. We define the equivalence of two nodes $u, v \in V \setminus X$, which we denote by $u \sim v$, inductively by stating that u and v are equivalent if both are assigned with the* *same gate function and $u_L \sim v_L$ and $u_R \sim v_R$. We call a circuit onefold if there are no two distinct nodes that are equivalent.*

Definition 6 (Loose End). *A node $v \in V \setminus X$ in a circuit C is called a loose end if $v \neq v_{out}$ and there is no edge $(v, u, l) \in E$ for some node u and a label l.*

Recall that a non-input node v in a circuit C is just a natural number that is labeled with a gate function, and that we require that v is greater than its predecessors. In other words, we assert that the order of the natural numbers $<$ is a topological order of the underlying graph of C. Thereby, starting with the empty circuit that only contains the input nodes X, $<$ describes an order in which we can add the gates to obtain the complete circuit C. But notice that this order is not always unique because C is not required to have a path that contains all nodes. Of course, we demand uniqueness so that we do not check the same circuit multiple times. Therefore, we commit to a specific topological order. As a first step, we formalize the depth of v, i.e., the length of the longest path from an input node to v.

Definition 7 (Depth of a Node). *For a node v in a circuit C we define the depth d of v as*

$$d(v) = \max\{d(v_L), d(v_R)\} + 1$$

where $d(x) = 0$ for all $x \in X$.

Now it only remains to sort nodes of the same depth. The input nodes $X = \{0, 1, \ldots, q-1\}$ are given, so we simply use $<$ for X. For non-input nodes, we use the associated gate function and the left and right predecessor. For the gate functions we assume that $\mathscr{G} = \{G_0, G_1, \ldots, G_{g-1}\}$ is ordered by $G_0 < G_1 < \cdots < G_{g-1}$. We formalize this in the following definition.

Definition 8 (Order of Nodes and Ordered Circuit). *For two nodes $u, v \in V$ in a onefold circuit C we have $u \prec v$ if $u \neq v$ and*

- *$d(u) < d(v)$ or*
- *$u, v \in X$ and $u < v$ or*
- *$d(u) = d(v)$ and $u_G < v_G$ or*
- *$d(u) = d(v)$ and $u_G = v_G$ and $u_L \prec v_L$ or*
- *$d(u) = d(v)$ and $u_G = v_G$ and $u_L = v_L$ and $u_R \prec v_R$.*

We call a circuit ordered if for all pairwise distinct nodes $u, v \in V$ it holds that $u < v \iff u \prec v$.

Definition 9 (Normal form for Circuits). *A circuit C is in normal form if C is onefold, without loose ends and ordered. We call such circuits normal.*

Lemma 1. *For every circuit C, there is an equivalent circuit C' s.t. C' is in normal form and has at most as many nodes as C.*

Proof. To bring C into normal form, we first make C onefold. To do so, for all equivalent nodes $u, v \in V$ we remove u and replace edges (u, w, l) by (v, w, l). After that, we remove all loose ends. Last, we make C ordered by permuting the underlying graph D and its labels accordingly. □

Now that we have established our definitions for circuits, we want to study the number of possible circuits. Recall that our basic approach is to fix X and $\mathscr{G}$ based on the construction we are interested in, and also bound the number of non-input nodes by some $k \in \mathbb{N}$. The essential part here is that there are gates in $\mathscr{G}$ that correspond to oracles an attacker would have access to in the corresponding security games. For now, we only restrict the size of X and $\mathscr{G}$, i.e., we consider the circuit class

$$\mathscr{C}(q, g, k) = \{C \mid |X| = q, |\mathscr{G}| = g, |V \setminus X| = k\}.$$

Strictly speaking, we should not only consider circuits of size exactly k but circuits of size up to k. But as we will see, for our application there are by far more circuits of size k then there are circuits of size $k - 1$. So for the sake of simplicity, in our analysis, we only consider circuits of size exactly k.

Of course, we are mainly interested in the number of normal circuits, i.e., in the size of

$$\mathscr{C}_{norm}(q, g, k) = \{C \in \mathscr{C}(q, g, k) \mid C \text{ is normal}\}$$

because from Lemma 1 it immediately follows that they cover all circuits up to equivalence. Nevertheless, we will first derive the number of all circuits of size k which we will then use to estimate the number of normal circuits.

To determine the number of all circuits, we consider a tree T with root X and depth k, where every edge corresponds to adding a new gate. The leaves of this tree are the circuits in $\mathscr{C}(q, g, k)$. We call T the circuit tree of $\mathscr{C}(q, g, k)$. T is diagrammed in Fig. 4. There are $g \cdot q^2$ possible ways to add the first gate. $g \cdot (q + 1)^2$ for the next and so on. In total, the number of all leaves is

$$|\mathscr{C}(q, g, k)| = \prod_{i=0}^{k-1} g \cdot (q + i)^2 = g^k \cdot \left(q^{\overline{k}}\right)^2 . \tag{10}$$

Therefore, the circuit size k has the highest impact on the number of all circuits and the number of input nodes q has a slightly higher impact than the number of gate functions g.

Now, to estimate the number of normal circuits, we wrote a `sage` script that essentially generates random circuits and then checks if they are normal. The results of this simulation, as well as the number of all circuits, are illustrated in Fig. 5 for the parameters $k \leq 9$ and $(q, g) \in \{(1, 3), (3, 3), (3, 5)\}$. The choice of these parameters is motivated by the idea of using $X = \{x\}$ or $X = \{x, \alpha_b, \alpha_{\bar{b}}\}$ and $\mathscr{G} = \{\oplus, ENC, DEC\}$ where ENC and DEC might be split into a left and a right half. We can draw two conclusions from Fig. 5. On the one hand, for small circuit classes, i.e., for circuit of size five or smaller, we might iterate all normal circuits with moderate computing power. On the other hand, we need

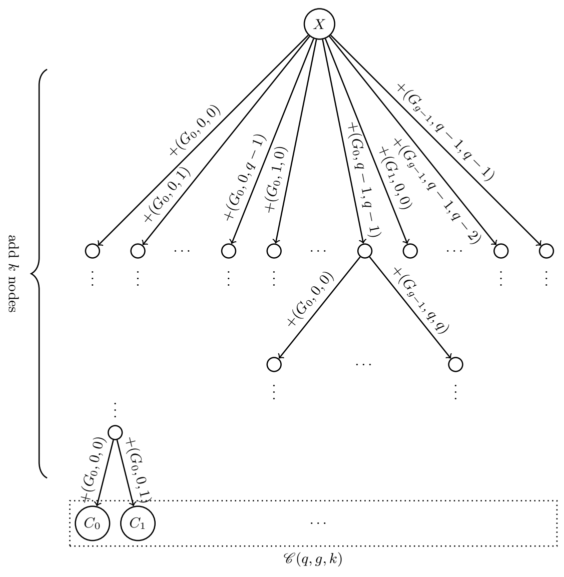


Fig. 4. Circuit tree T of $\mathscr{C}(q,g,k)$. By $+(G,l,r)$ we denote the adding of a node with gate function G, left predecessor l and right predecessor r.

to reduce the number of circuit further for larger classes. To do so, we establish a set of rules R that circuits have to comply with. These rules are basically an expansion of our already set up requirement that circuits must be normal. But they also include requirements that are based on the concrete choice of X and $\mathscr{G}$. E.g., one gate function that we always use is the XOR function, and it is natural to require that the inputs of the XOR are ordered since XOR is commutative. Thus, we consider the class of circuits

$$\begin{aligned}\mathscr{C}(X,\mathscr{G},k,R) = \{C \mid\ & C \text{ has input nodes } X, \text{ gate functions } \mathscr{G},\\ & |V \setminus X| = k,\ \forall r \in R : r(C) = 1\}.\end{aligned}$$

Notice that here we overload the notation because we are not interested in the pure size of X and $\mathscr{G}$ anymore. In the following, we use $R(C) = \wedge_{r\in R} r(C)$ to shorten the notation.

5.2 Enumerating Circuits

The most straightforward way of enumerating $\mathscr{C}(X,\mathscr{G},k,R)$ is surely to simply enumerate all circuits of size k with input nodes X and gate functions $\mathscr{G}$. Then, for each circuit, we can check whether it complies with R. But, as we have seen in the last section, there are many more circuits than circuits in normal form. So, this approach is rather inefficient.

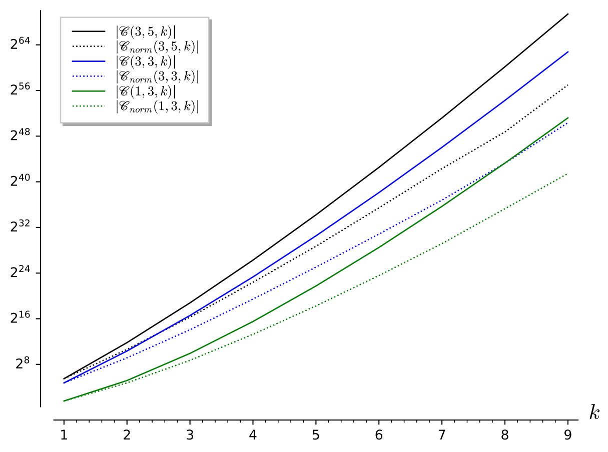


Fig. 5. Number of circuits for various parameters.

To find a better strategy, we assume that we cannot only test R on a complete circuit $C \in \mathscr{C}(X, \mathscr{G}, k, \emptyset)$ but also on every partial circuit

$$C' \in \bigcup_{1 \leq i < k} \mathscr{C}(X, \mathscr{G}, i, \emptyset)$$

in such a way that no partial circuit C' with $R(C') = 0$ can be extended to a circuit C s.t. $R(C) = 1$. E.g., again consider the XOR gate and observe that once we added a bad XOR gate, there is no way to get to a good circuit. The same applies to all other rules we use. Now recall the circuit tree T from Fig. 4. We are interested in enumerating all leaves C of T for which $R(C) = 1$ holds. Let us assume, that we have already identified the first, i.e., the leftmost, circuit C^* with this property. Then, we will first check all right siblings of C^*. After that, we test $R(C^\dagger)$ where $C^\dagger$ is the first right sibling of the parent of C^*. If $R(C^\dagger) = 1$ we will check all children of $C^\dagger$. Otherwise, we continue with the next sibling of $C^\dagger$. At some point, there are no more siblings of $C^\dagger$ left. Then, we continue with the same approach on the level of the parent of $C^\dagger$. As soon as we hit the root of T we are finished with our search.

Notice that, since the setting is rather vague, neither can we exclude that there are more efficient ways to enumerate circuits or even find periodic circuits nor can we give a general runtime analysis of our approach. Instead, we will analyze the runtime for concrete and practical examples in Sect. 5.4.

One rule that we always want to use is

$$r_{normal}(C) = \begin{cases} 1 & \text{if } C \text{ is in normal form} \\ 0 & \text{else.} \end{cases}$$

Therefore, we now describe how r_{normal} can be checked on partial circuits. First, recall that we call a circuit normal if it is onefold, without loose ends and ordered.

For loose ends, we observe that with each new node, we can get rid of at most one loose end. So if

$$|\{v \in V \setminus X \mid \nexists u \in V : v = u_L \vee v = u_R\}| > 1 + (k - (|V| - |X|))$$

holds for a partial circuit C', there is no way of adding $k - (|V| - |X|)$ nodes to C' to obtain a circuit without loose ends. To check whether C' is ordered, we only have to test $u \prec v$ as defined in Definition 8 where v is the last added and u is the second but last added node resp. This is because we will check $R(C')$ only if the predecessor of C' complied with R. For the same reason, if v is the last added node, it suffices to check whether the set

$$\{u \in V \mid u_G = v_G \text{ and } u_L = v_L \text{ and } u_R = v_R\}$$

is empty to check if C' is onefold. This is because if C' without v is onefold then

$$u_L \sim v_L \iff u_L = v_L$$

and the same holds for the right predecessor.

5.3 Testing a Circuit for Periodicity

Testing a circuit C, i.e., the function C_f, for periodicity is fairly simple because we can evaluate C_f on all inputs and then check whether there are any periods. When we enumerate circuits, we do not evaluate each circuit from the ground up. Instead, we store the evaluation of each node s.t. after adding a node and checking that it complies with all rules, we only have to evaluate the new node.

Aside from that, we want to filter out as much trivial periods as possible. By trivial periods, we mean periods that bear no useful information. For example, consider the function

$$f(x) = E(x) \oplus E(x \oplus \alpha)$$

where E is an encryption oracle and α is an arbitrary constant. Obviously, f has a period $s = \alpha$. But this has nothing to do with the structure of E and in fact, if we replace E with a truly random function E', the period persists. Since our enumeration will encounter functions of this kind, we want to check whether a period is trivial or not. To do so, we can replace all oracles by their random version and see whether the period persists. If so, we discard the period. Notice that the period may change since the period could depend on the oracle gates, e.g., consider f as above but replace α with $E(\alpha)$. We want to remark that this essentially matches the definition of a distinguisher, i.e., a non-trivial period is a property that is present in the real but not in the random case.

Notice, since we instantiate the constructions with a small input size, we might identify or miss a period accidentally. In practice, this was never a problem and in case of uncertainty, we can always repeat the search with fresh randomness or slightly increase the input size.

5.4 Setup of Our Search

Known Attacks Here, we rediscover known attacks with our automated search algorithm. Of course, the primary intention here is to identify sound choices for gates, inputs and rules that we can reuse when we search for novel attacks. Furthermore, we use the occasion to investigate the running time of our algorithm in a practical scenario.

Recall that the attack on the Even-Mansour cipher [17] is based on the function

$$f(x) = ENC(x) \oplus P(x) = P(x \oplus k_0) \oplus k_1 \oplus P(x) \tag{11}$$

with period $s = k_0$. Therefore, our attack requires input nodes $X = \{x\}$ and gates $\mathscr{G} = \{\oplus, ENC, P\}$. Notice that, both ENC and P are functions of one variable, but our definition requires that both are functions of two variables. Therefore, we set the rule

$$r_{SI}(C) = \begin{cases} 1 & \text{if } \forall v \in V : v_G \in \{ENC, P\} \Rightarrow v_L = v_R \\ 0 & \text{else.} \end{cases}$$

We can enforce r_{SI} simply by checking if $v_G \in \{ENC, P\} \Rightarrow v_L = v_R$ holds for each node v that we add to a circuit. Furthermore, since for Even-Mansour P is just a public permutation, only circuits that contain at least one ENC gate are of interest. Therefore, we add the rule

$$r_{1E}(C) = \begin{cases} 1 & \text{if } \exists v \in V : v_G = ENC \\ 0 & \text{else.} \end{cases}$$

This rule is of course only enforced for complete circuits, i.e., for circuits of size k. Equivalently, we can say that $r_{1E}(C') = 1$ for all partial circuits C'. Last, we add a rule that eliminates some circuits that are equivalent to another circuit because of the commutative or self inverse properties of the XOR function. That is, $r_\oplus$ excludes all circuits that contain a node v with $v_G = \oplus$ for which $v_L \prec v_R$ does not hold. Again, this can be checked each time we add a new gate. This also excludes circuits which compute the XOR of a node with itself. In addition, $r_\oplus$ shall exclude circuits for which there is a node that contributes twice to a XOR sum, and also circuits for which there are two nodes that compute the same XOR sum. To check these, for each node v we store a set Σ_v that is defined as follows

$$\Sigma_v = \begin{cases} \{v\} & \text{if } v \in X \text{ or } v_G \neq \oplus \\ \Sigma_{v_L} \cup \Sigma_{v_R} & \text{if } v_G = \oplus. \end{cases}$$

Thereby, Σ_v contains all nodes that contribute to the XOR sum if v is an XOR gate. So if we add a new node v we only have to check that the intersection $\Sigma_{v_L} \cap \Sigma_{v_R}$ is empty and that there is no other node u s.t. $\Sigma_u = \Sigma_v$.

As we saw in Sect. 3.1, the distinguishing attack by Kuwakado and Morii against 3-round Feistel cipher uses the function

$$f(x, b) = ENC_L(x, \alpha_b) \oplus \alpha_{\bar{b}} = F_1(x \oplus F_0(\alpha_b)) \oplus \alpha_b \oplus \alpha_{\bar{b}}$$

with period $s = (F_0(\alpha_0) \oplus F_0(\alpha_1))||1$ where $\alpha_0, \alpha_1 \in \mathbb{F}_2^n$ are arbitrary distinct constants. To find this f, we use $X = \{x, \alpha_b, \alpha_{\bar{b}}\}$ and set $\mathscr{G} = \{\oplus, ENC_L, ENC_R\}$. In terms of rules, we first notice that α_b and $\alpha_{\bar{b}}$ are equivalent for the attack. Therefore, we demand that α_b appears first. Further, the output must depend on x and at least on one of α_b and $\alpha_{\bar{b}}$ because otherwise there will be trivial periods. To check this, for complete circuits, we simply check whether there is a path from x to v_{out} and from α_b or $\alpha_{\bar{b}}$ to v_{out}. All the above rules do not exclude useful periodic circuits from the search. To decrease the search space even further, we limit the number of oracle queries to one. Therefore, ENC_L and ENC_R each are allowed only once in a circuit. If they both appear in a circuit, we require them to have the same inputs. Last, we enforce that ENC_L and ENC_R depend on x. This, again, can be checked with a reachability test in the underlying digraph, and excludes circuits that, e.g., compute $ENC_L(\alpha_b, \alpha_{\bar{b}})$. Although the last two rules might exclude useful periodic circuits, based on the periodic functions from the literature and our experience, we believe that this are reasonable additions to our set of rules.

Other attacks make also use of decryption queries, e.g., in [13, Section 5] the authors present a quantum chosen-ciphertext distinguisher against 4-round Feistel. To find such, we add decryption oracles to the set of gate functions. So, for Feistel, we have $\mathscr{G} = \{\oplus, ENC_L, ENC_R, DEC_L, DEC_R\}$. Naturally, we add a rule to exclude circuits that, e.g., decrypt an unaltered ciphertext. Further, we demand that DEC gates depend on ENC gates.

We now briefly discuss the complexity of the searches on Even-Mansour and Feistel. Consider Fig. 5 for k up to 6 and compare it with Fig. 6 which shows the complexity, i.e., the number of tests our algorithm needs. The solid lines are essentially the same in both plots. A trivial search algorithm would test all circuits (solid lines) for normality and then test all normal circuits (dotted lines from Fig. 5) for periodicity. Our algorithm does not only test for normality, but also for other rules defined by R. But these tests are done in a sophisticated way on (partial) circuits. Therefore, for our choices of R, a single test is not more complex than a test for normality. But the number of tests is clearly reduced (solid vs. dashed lines). And so is the number of tests for periodicity (dotted lines from Fig. 5 vs dotted lines from Fig. 6).

MISTY Next, we describe how we set up our automated search algorithm to find periodic functions for 4-round MISTY R-FK and 5-round MISTY L-FK. Based on our insights from the previous section, we choose gates

$$\mathscr{G} = \{\oplus, ENC_L, ENC_R, F, F^{-1}\}.$$

So, we split the encryption gate in the same manner as for our searches on Feistel structures. In terms of rules, we use r_{normal} and $r_{\oplus}$ and of course also ensure that F and F^{-1} only take a single input and do not uncompute each other. If we search for large circuits of size $k > 5$ we also restrict the number of encryption queries to one, again in the same way as we did for Feistel. For the input nodes X we either choose only a single input x or x and a constant α. For 5-round

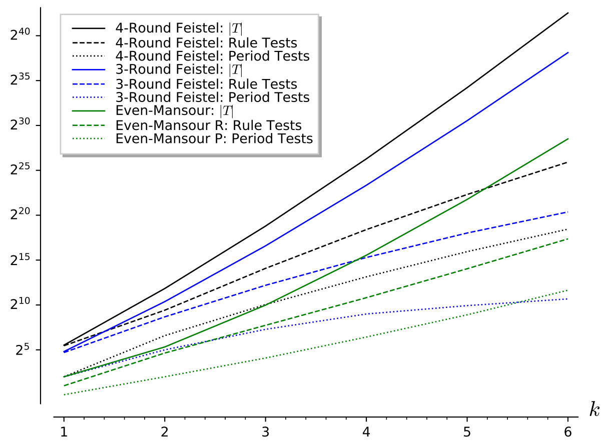


Fig. 6. Number of circuits and tests in practice.

MISTY L-FK, the search yields not only f_2 from Eq. (5) but also the following additional functions t_i with periods r_i.

$$\begin{aligned}
t_1(x) &= ENC_L(F^{-1}(x), \alpha) \oplus x && r_1 = F(\alpha) \oplus k_1\\
t_2(x) &= ENC_R(F^{-1}(F^{-1}(x)), F^{-1}(x)) \oplus x && r_2 = F(k_0) \oplus k_2\\
t_3(x) &= ENC_L(F^{-1}(F^{-1}(x)), F^{-1}(x)) \oplus F(x) &&\\
&\quad \oplus ENC_R(F^{-1}(F^{-1}(x)), F^{-1}(x)) && r_3 = k_0 \oplus k_1 \oplus k_2 \oplus F(k_0)\\
t_4(x) &= ENC_L(F^{-1}(F^{-1}(x)), F^{-1}(x)) \oplus F(x) \oplus x && r_4 = k_0 \oplus k_1
\end{aligned}$$

Here, $\alpha \in \mathbb{F}_2^n$ is an arbitrary constant. However, notice that these are of course our polished results. In reality, we might have encountered some variations first. For instance, if we choose the circuit size k for t_1 too large, we find circuits where α is permuted first. Furthermore, notice the close connection between t_2, t_3 and t_4 and their corresponding periods. We have $t_4(x) = t_2(x) \oplus t_3(x)$ and $r_4 = r_2 \oplus r_3$. This is of course due to the internal structure of MISTY and nothing we can exploit in general. E.g., consider t_1 and t_2. It is easy to verify that $t_1(x) \oplus t_2(x)$ is not periodic.

For MISTY, the search that yielded the circuit for t_3, for which we have $|X| = 1$, $|\mathscr{G}| = 5$ and $k = 7$, was the most expensive one. Our laptop took six minutes to traverse the circuit tree of size 2^{41} by doing 2^{25} rule tests and 2^{18} periodicity tests.

Feistel For our search on Feistel, we use essentially the same setup as before. Furthermore, for 4-round Feistel-FK, our search yields not only f_3 from Eq. (6) but also

$$t_5(x) = ENC_L(F(x), x) \oplus F(x)$$

with period $r_5 = k_1 \oplus F(k_0)$.

For the second search on 5-rounds Feistel-FK, we add a gate function F^{-1} and remove ENC_L or ENC_R to reduce the runtime. This search, for which we have $|X| = 2$, $|\mathscr{G}| = 4$ and $k = 7$, was the most expensive one. Our laptop took twelve minutes to traverse the circuit tree of size 2^{44} by doing 2^{26} rule tests and 2^{19} periodicity tests.

6 Proofs of the Results in Section 4

In this section we first show that equivalent Hadamard matrices give rise to essentially equivalent quantum algorithms (Proposition 1). Finally, we give a proof of Theorem 2, with the exception of a technical result that we prove in Appendix D.

6.1 Equivalence of Hadamard Transformations

Let us now consider a Hadamard matrix $H \in \mathbb{R}^{2^n \times 2^n}$ and consider it given as

$$(H)_{x,y} = (-1)^{g(x,y)},$$

for a suitable choice of $g : \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2$.

We define the Fourier-like transformation of H as follows

Definition 10. *For a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$, we define*

$$\widehat{f}^H(x) := (H\phi)_x = \sum_{y \in \mathbb{F}_2^n} (-1)^{f(y)+g(x,y)},$$

where $(H\phi)_x$ is the component x of the vector $H_0\phi$, and ϕ is the vector

$$\phi = \left((-1)^{f(0)}, \ldots, (-1)^{f(2^n-1)}\right)^T.$$

Hadamard matrices are an interesting research topic on their own, with fundamental questions still being unsolved. In particular, it is not even clear for which dimension N Hadamard matrices exist. A famous conjecture states that there exists at least one Hadamard matrix if the dimension is divisible by four. This is still an open problem and, at the time of writing, the smallest such integer for which no Hadamard matrix is known is $N = 668$ [15].

However, since we work on vector spaces of dimension 2^n for some $n \in \mathbb{N}$, we know that there is at least one Hadamard matrix for each of these dimensions, and that is indeed H_0.

In fact, a related question that is more interesting for our context is understanding how many *different* Hadamard matrices exist in each dimension that could possibly result in meaningfully different routines `Simon`(H). To this end, we consider the following – standard – notion of equivalence.

Definition 11. *Two Hadamard matrices H, H' are said to be equivalent if there exist diagonal matrices D_1, D_2 whose diagonals take entries in $\{-1, 1\}$ and permutation matrices P_1, P_2 such that*

$$H' = P_1 D_1 H D_2 P_2.$$

Similarly, we will say that two transformations $\widehat{f}^H$ and $\widehat{f}^{H'}$ are equivalent if H and H' are equivalent.

As we prove in Appendix C, two equivalent matrices H and H' will result in the circuits $\texttt{Simon}(H)$ and $\texttt{Simon}(H')$ being the same, up to some classical pre- and post-processing, as we state in the following result.

Proposition 1. *Let H and H' be two equivalent Hadamard matrices. There exist functions $\texttt{PRE} : \mathcal{B}_n \to \mathcal{B}_n$ and $\texttt{POST} : \mathbb{F}_2^n \to \mathbb{F}_2^n$ such that, for any $f : \mathbb{F}_2^n \to \mathbb{F}_2$*

$$\Pr(x \leftarrow \texttt{Simon}(H')(f)) = \Pr(x \leftarrow \texttt{POST}(\texttt{Simon}(H)(\texttt{PRE}(f)))).$$

Thus $\texttt{Simon}(H)$, with some classical pre- and post-processing, perfectly simulates the routine $\texttt{Simon}(H')$ for any H' equivalent to H. It follows that we can indeed reduce ourselves to consider only non-equivalent Hadamard matrices.

6.2 Proof of Theorem 2

In this section we provide a proof of Theorem 2. First, we prove that we can reduce the problem to the simpler case where the matrix B is not arbitrary and simply consider $B = A^T$ (Proposition 2), thanks to a result on the structure of the group $\mathrm{GL}(n, \mathbb{F}_2)$. After that, we prove that the possibility for H are then limited (Proposition 3) and conclude the proof.

Let us consider a Hadamard matrix H and let $\widehat{f}^H : \mathbb{F}_2^n \to \mathbb{Z}$ be its transform $\widehat{f}^H(i) = (H\phi)_i$ as in Sect. 4.1. Let $g : \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2$ be the (unique) Boolean function implicitly defined by $(-1)^{g(x,y)} = (H)_{x,y}$. We will make this relation explicit by indicating H as H_g. The idea of the proof of Theorem 2 is that, if $\widehat{f}^{H_g}$ satisfies the theorem, then the corresponding Boolean function g is (almost) the scalar product, and therefore H_g is equivalent to $H_0 = H_{\langle x,y \rangle}$.

Indeed, let for any Boolean function f we have:

$$\widehat{f \circ A}^{H_g}(x) = \sum_{y \in \mathbb{F}_2^n} (-1)^{(f \circ A)(y) + g(x,y)} = \sum_{y \in \mathbb{F}_2^n} (-1)^{f(y) + g(x, A^{-1}y)}$$

where we considered A^{-1} instead of A for ease of notation. On the other hand,

$$(\widehat{f}^{H_g} \circ B)(x) = \sum_{y \in \mathbb{F}_2^n} (-1)^{f(y) + g(Bx, y)}.$$

Given the arbitrary choice of f, the above quantities are equal for all $x \in \mathbb{F}_2^n$ if and only if

$$g(Bx, y) = g(x, A^{-1}y) \tag{12}$$

for all $x, y \in \mathbb{F}_2^n$. In fact, if there exist $\chi, \upsilon \in \mathbb{F}_2^n$ for which Eq. (13) does not hold, then if we consider the Boolean function $f_\chi(y) := g(\chi, A^{-1}y)$ for all $y \in \mathbb{F}_2^n$, it is clear that $\widehat{f_\chi \circ A}^{H_g}(\chi) \neq \widehat{f_\chi}^{H_g} \circ B(\chi)$. Furthermore, notice that for the arbitrary choice of A, Eq. (12) is equivalent to finding B such that

$$g(x, Ay) = g(B^{-1}x, y). \tag{13}$$

Therefore proving the theorem is equivalent to solving Eq. (13).

The main difficulty of the proof is the a priori freedom in choosing B for a given A. We therefore want to first understand what is the dependence of B on A. To do so, we define the function $F_g : \mathrm{GL}(n, \mathbb{F}_2) \to \mathrm{GL}(n, \mathbb{F}_2)$ such that

$$F_g(A) = B.$$

The key observation is that this mapping is actually an automorphism over $\mathrm{GL}(n, \mathbb{F}_2)$. Indeed, if g is a solution of Eq. (13), F_g is a well-defined function because if for $A \in \mathrm{GL}(n, \mathbb{F}_2)$ there existed two distinct $B, B' \in \mathrm{GL}(n, \mathbb{F}_2)$ such that Eq. (13) holds, then if we consider $z \in \mathbb{F}_2^n$ such that $B^{-1}z \neq (B')^{-1}z$, we would have that

$$g(B^{-1}z, x) = g(z, Ax) = g((B')^{-1}z, x)$$

for all $x \in \mathbb{F}_2^n$. But this contradicts the fact that $H_g H_g^T$ is the identity matrix.

The same reasoning shows that F_g is injective, and therefore bijective. Furthermore, it also holds that $F_g(A \cdot A') = F_g(A) \cdot F_g(A')$, from which it follows that F_g is an automorphism over $\mathrm{GL}(n, \mathbb{F}_2)$. Luckily for us, it is possible to characterize such automorphisms in a nice way, thanks to a result due to Dieudonné [8], which can be found in a more convenient formulation for our purposes in [18].

Lemma 2 ([8,18]). *For every automorphism F of $\mathrm{GL}(n, \mathbb{F}_2)$, there exists $G \in \mathrm{GL}(n, \mathbb{F}_2)$ such that either*

$$F(A) = G^{-1}AG \quad \text{or} \quad F(A) = G^{-1}(A^T)^{-1}G \tag{14}$$

for all $A \in \mathrm{GL}(n, \mathbb{F}_2)$.

This characterization, together with the considerations above, will allow us to reduce the proof to the case of Theorem 2 with $B = (A^T)^{-1}$, as is stated by the following proposition.

Proposition 2. *For any g such that H_g is Hadamard and that fulfils Eq. (13), there exists a matrix $G \in \mathrm{GL}(n, \mathbb{F}_2)$ such that $g(x, y) = \overline{g}(Gx, y)$, where $\overline{g}$ fulfils*

$$\overline{g}(x, Ay) = \overline{g}(A^T x, y). \tag{15}$$

Proof. The result is trivial for $n = 1, 2$ and thus we assume $n \geq 3$ from now on.

Let us suppose that F_g is of the first form, i.e. it exists $G \in \mathrm{GL}(n, \mathbb{F}_2)$ such that $F_g(A) = G^{-1}AG$ for all $A \in \mathrm{GL}(n, \mathbb{F}_2)$. Let us prove that this is not possible. Let $x \neq y \in \mathbb{F}_2^n \setminus \{0\}$. Then, since H_g is Hadamard, we have that $\sum_{z\in\mathbb{F}_2^n}(-1)^{g(x,z)+g(z,y)} = 0$, that is there are exactly 2^{n-1} values of z for which $g(x,z) \neq g(z,y)$. This means that for any $n \geq 3$ we can find z_1, z_2 such that $z_1 \notin \{0, Gx\}$ and $z_2 \notin \{0, G^{-1}y\}$ and $g(x,z_1) \neq g(x,z_2)$. Then, there exits $A \in \mathrm{GL}(n,\mathbb{F}_2)$ such that $y = Az_1$ and $x = G^{-1}A^{-1}Gz_2$ (that is, A is such that $z_1 \mapsto^A y$ and $Gx \mapsto^A Gz_2$), so that

$$g(z_2,y) = g(z_2,Az_1) = g(F_g(A)^{-1}z_2,z_1) = g(G^{-1}A^{-1}Gz_2,z_1) = g(x,z_1)$$

which leads to a contradiction.

Let us now suppose that F_g is of the second form, i.e. $F_g(A) = G^{-1}(A^T)^{-1}G$ for some $G \in \mathrm{GL}(n,\mathbb{F}_2)$. Consider $\overline{g}$ such that $\overline{g}(x,y) = g(x,(G^T)^{-1}y)$. Since $(G^T)^{-1}$ is a permutation, we have that H_g is Hadamard if and only if $H_{\overline{g}}$ is Hadamard and, furthermore, g satisfies Eq. (13) with $B = F_g(A)^{-1} = G^{-1}A^TG$ if and only if for all $x, y \in \mathbb{F}_2^n$ and $z = G^Ty$

$$\begin{aligned}\overline{g}(x,Az) = \overline{g}(x,AG^Ty) = g(x,(G^T)^{-1}AG^Ty) = g(F_g((G^T)^{-1}AG^T)^{-1}x,y)\\ = g(A^Tx,y) = \overline{g}(A^Tx,G^Ty) = \overline{g}(A^Tx,z),\end{aligned}$$

i.e. $\overline{g}$ satisfies Eq. (13) with $B = A^T$. This is true for any $A \in \mathrm{GL}(n,\mathbb{F}_2)$ and concludes the proof.

□

In other words, for the proof of Theorem 2 we can consider without loss of generality the case $B = F_g(A) = (A^T)^{-1}$.

What is now left to conclude is that if $F_g(A)^{-1} = A^T$, then g is *almost* the scalar product $\langle x,y\rangle$. This is exactly what the following proposition states. The rather technical proof of it can be found in Appendix D.

Proposition 3. *Let $g : \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2^n$ such that for any $A \in \mathrm{GL}(n,\mathbb{F}_2)$*

$$g(x,Ay) = g(A^Tx,y) \tag{16}$$

for all $x, y \in \mathbb{F}_2^n$. Then

$$g(x,y) = \varepsilon_0 + \varepsilon_1\delta_{\{0\}}(x) + \varepsilon_2\delta_{\{0\}}(y) + \varepsilon_3\langle x,y\rangle + \varepsilon_4\delta_{\{(0,0)\}}(x,y)$$

where δ_M is the indicator function of a set M and $\varepsilon_0, \ldots, \varepsilon_4 \in \mathbb{F}_2$.

Remark 1. The family of functions g of the proposition are the very natural solutions that are found by observing that we have

1. if g, h are solutions then $g + h$ is also a solution;
2. the scalar product $\langle x,y\rangle$ is a solution;
3. if $g(x,y) = \delta_M(x,y)$, where M is either $\{(0,0)\}$ or $\{0\} \times \mathbb{F}_2^n$ or $\mathbb{F}_2^n \times \{0\}$ or $\mathbb{F}_2^n \times \mathbb{F}_2^n$, then g is a solution because for any $A \in \mathrm{GL}(n,\mathbb{F}_2)$ and *any* $B \in \mathrm{GL}(n,\mathbb{F}_2)$, $g(x,Ay) = g(x,y) = g(Bx,y)$.

Now we can conclude the proof of Theorem 2.

Proof of Theorem 2. We restrict to the case of g such that H_g is Hadamard and fulfils Eq. (16). If g represents a Hadamard matrix H_g, then we know that

$$(H_g H_g^T)_{\alpha,\beta} = \sum_{x\in\mathbb{F}_2^n} (-1)^{g(x,\alpha)+g(x,\beta)}$$

is zero whenever $\alpha \neq \beta$. We can assume the form given in Proposition 3, that is

$$g(x,y) = \varepsilon_0 + \varepsilon_1\delta_{\{0\}}(x) + \varepsilon_2\delta_{\{0\}}(y) + \varepsilon_3\langle x,y\rangle + \varepsilon_4\delta_{\{(0,0)\}}(x,y).$$

Then we have

$$\begin{aligned} g(x,\alpha)+g(x,\beta) = \varepsilon_2(\delta_{\{0\}}(\alpha)+\delta_{\{0\}}(\beta)) + \varepsilon_3\langle x,\alpha+\beta\rangle + \\ \varepsilon_4(\delta_{\{(0,0)\}}(x,\alpha)+\delta_{\{(0,0)\}}(x,\beta)) \end{aligned}$$

so that

$$\sum_{x\in\mathbb{F}_2^n}(-1)^{g(x,\alpha)+g(x,\beta)} = \sum_{x\in\mathbb{F}_2^n}(-1)^{\varepsilon_3\langle x,\alpha+\beta\rangle+\varepsilon_4(\delta_{\{(0,0)\}}(x,\alpha)+\delta_{\{(0,0)\}}(x,\beta))}.$$

But if $\alpha = 0$ and $\beta \neq 0$, then we must have that

$$(H_g H_g^T)_{0,\beta} = \sum_{x\in\mathbb{F}_2^n}(-1)^{g(x,0)+g(x,\beta)} = \sum_{x\in\mathbb{F}_2^n}(-1)^{\varepsilon_3\langle x,\beta\rangle+\varepsilon_4(\delta_0(x))} = 0$$

where we have used the fact that H_g is Hadamard. But $\sum_{x\in\mathbb{F}_2^n}(-1)^{\langle x,\beta\rangle} = 0$, as $\beta \neq 0$. Then, the above equality holds if and only if $\varepsilon_3 = 1$ and $\varepsilon_4 = 0$. It follows that if g is Hadamard and satisfies Eq. (16) then

$$g(x,y) = \varepsilon_0 + \varepsilon_1\delta_{\{0\}}(x) + \varepsilon_2\delta_{\{0\}}(y) + \langle Gx,y\rangle$$

which means that $H_g = (-1)^{\varepsilon_0}(D_{\delta_{\{0\}}})^{\varepsilon_1}H_0(D_{\delta_{\{0\}}})^{\varepsilon_2}$, where D_h is the diagonal matrix such that $(D_{\delta_{\{0\}}})_{z,z} = \delta_{\{0\}}(z)$ for all $z \in \mathbb{F}_2^n$. Therefore, we conclude that H_g is indeed equivalent to H_0. Thus concludes the proof of our main result. $\square$

7 Conclusion

Motivated by the search for alternative ways to attack symmetric primitives, our goal was to find new applications of Simon's algorithm, as well as study possible generalizations.

On the one hand, we have seen that the standard Hadamard transformation is the only one that preserves linear equivalence. This result suggests that using alternative Hadamard transformations will not help in improving known attack vectors using the well studied linear-invariant cryptographic criteria. However, it does not exclude the possibility of new attacks based on non linear-invariant properties, that could profit from $\texttt{Simon}(H)$. Moreover, for exploiting the standard criteria, our result shows that it is most promising to focus on the standard Hadamard-transform when searching for new quantum algorithms.

On the other hand, we have successfully automatized the search of periodic functions, resulting in new key-recovery attacks. However, our algorithm can be applied to other targets and can be still improved in order to search for larger circuits. We leave a more optimized and parallelized implementation or the use of a different representation of the functions, possibly allowing for the use of symbolic computation, as interesting future work.

Acknowledgments

This work was funded by the DFG (German Research Foundation), under Germany´s Excellence Strategy - EXC 2092 CASA – 390781972. The authors would further like to thank Alessandro Ghigi and Philippe Langevin for fruitful discussions, particularly for pointing out the results of [8] and [18].

References

1. Bernstein, E., Vazirani, U.V.: Quantum complexity theory. SIAM J. Comput. **26**(5), 1411–1473 (1997)
2. Bonnetain, X., Hosoyamada, A., Naya-Plasencia, M., Sasaki, Y., Schrottenloher, A.: Quantum attacks without superposition queries: The offline Simon's algorithm. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology - ASIACRYPT 2019, Proceedings, Part I. LNCS, vol. 11921, pp. 552–583. Springer (2019)
3. Bonnetain, X., Jaques, S.: Quantum period finding against symmetric primitives in practice. IACR Cryptol. ePrint Arch. **2020**, 1418 (2020)
4. Bonnetain, X., Leurent, G., Naya-Plasencia, M., Schrottenloher, A.: Quantum linearization attacks. In: Tibouchi, M., Wang, H. (eds.) Advances in Cryptology - ASIACRYPT 2021, Proceedings, Part I. LNCS, vol. 13090, pp. 422–452. Springer (2021)
5. Carlet, C.: Boolean Functions for Cryptography and Coding Theory. Cambridge University Press (2021)
6. Chen, Y.L., Lambooij, E., Mennink, B.: How to build pseudorandom functions from public random permutations. In: Boldyreva, A., Micciancio, D. (eds.) Advances in Cryptology - CRYPTO 2019, Proceedings, Part I. LNCS, vol. 11692, pp. 266–293. Springer (2019)
7. Cui, J., Guo, J., Ding, S.: Applications of Simon's algorithm in quantum attacks on Feistel variants. Quantum Inf. Process. **20**(3), 117 (2021)
8. Dieudonné, J., Hua, L.: On the Automorphisms of the Classical Groups. Memoirs of the American Mathematical Society, American Mathematical Society (1951)
9. Dutta, A., Nandi, M., Talnikar, S.: Permutation based EDM: an inverse free BBB secure PRF. IACR Trans. Symmetric Cryptol. **2021**(2), 31–70 (2021)
10. Gouget, A., Patarin, J., Toulemonde, A.: (Quantum) cryptanalysis of Misty schemes. In: Hong, D. (ed.) Information Security and Cryptology - ICISC 2020, Proceedings. LNCS, vol. 12593, pp. 43–57. Springer (2020)
11. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Miller, G.L. (ed.) Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing. pp. 212–219. ACM (1996)

12. Guo, T., Wang, P., Hu, L., Ye, D.: Attacks on beyond-birthday-bound MACs in the quantum setting. In: Cheon, J.H., Tillich, J. (eds.) Post-Quantum Cryptography - 12th International Workshop, PQCrypto 2021, Proceedings. LNCS, vol. 12841, pp. 421–441. Springer (2021)
13. Ito, G., Hosoyamada, A., Matsumoto, R., Sasaki, Y., Iwata, T.: Quantum chosen-ciphertext attacks against Feistel ciphers. In: Matsui, M. (ed.) Topics in Cryptology - CT-RSA 2019 - The Cryptographers' Track at the RSA Conference 2019, Proceedings. LNCS, vol. 11405, pp. 391–411. Springer (2019)
14. Kaplan, M., Leurent, G., Leverrier, A., Naya-Plasencia, M.: Breaking symmetric cryptosystems using quantum period finding. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology - CRYPTO 2016, Proceedings, Part II. LNCS, vol. 9815, pp. 207–237. Springer (2016)
15. Kharaghani, H., Tayfeh-Rezaie, B.: A Hadamard matrix of order 428. Journal of Combinatorial Designs **13**(6), 435–440 (2005)
16. Kuwakado, H., Morii, M.: Quantum distinguisher between the 3-round Feistel cipher and the random permutation. In: IEEE International Symposium on Information Theory, ISIT 2010, Proceedings. pp. 2682–2685. IEEE (2010)
17. Kuwakado, H., Morii, M.: Security on the quantum-type Even-Mansour cipher. In: Proceedings of the International Symposium on Information Theory and its Applications, ISITA 2012. pp. 312–316. IEEE (2012)
18. Landin, J., Reiner, I.: Automorphisms of the general linear group over a principal ideal domain. Annals of Mathematics **65**(3), 519–526 (1957)
19. Leander, G., May, A.: Grover meets Simon - quantumly attacking the FX-construction. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology - ASIACRYPT 2017, Proceedings, Part II. LNCS, vol. 10625, pp. 161–178. Springer (2017)
20. Matsui, M.: New structure of block ciphers with provable security against differential and linear cryptanalysis. In: Gollmann, D. (ed.) Fast Software Encryption 1996, Proceedings. LNCS, vol. 1039, pp. 205–218. Springer (1996)
21. May, A., Schlieper, L.: Quantum period finding with a single output qubit - factoring n-bit RSA with n/2 qubits. CoRR **abs/1905.10074** (2019)
22. May, A., Schlieper, L., Schwinger, J.: Practical period finding on IBM Q - quantum speedups in the presence of errors. CoRR **abs/1910.00802** (2019)
23. Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information (10th Anniversary edition). Cambridge University Press (2016)
24. Rahman, M., Paul, G.: Quantum attacks on HCTR and its variants. IACR Cryptol. ePrint Arch. **2020**, 802 (2020)
25. Santoli, T., Schaffner, C.: Using Simon's algorithm to attack symmetric-key cryptographic primitives. Quantum Inf. Comput. **17**(1&2), 65–78 (2017)
26. Xie, H., Yang, L.: Using Bernstein-Vazirani algorithm to attack block ciphers. CoRR **abs/1711.00853** (2017)
27. Xie, H., Yang, L.: Using Bernstein-Vazirani algorithm to attack block ciphers. Des. Codes Cryptogr. **87**(5), 1161–1182 (2019)

A Simon's Algorithm

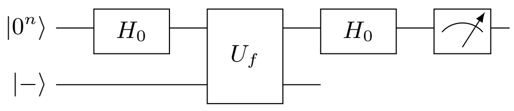


Fig. 7. Simon's Algorithm

Let $f : \mathbb{F}_2^n \to \mathbb{F}_2^m$ be a 2-to-1 periodic function, i.e. $f(a) = f(b)$ if and only if $b = a$ or $b = a + s$. Simon's algorithm consists in iterating the circuit in Fig. 7, which prepares and then measures the state

$$|0^n\rangle |0^m\rangle \xrightarrow{H_0 \otimes I_n} \sum_{x \in \mathbb{F}_2^n} |x\rangle |0^m\rangle \xrightarrow{U_f} \left(\sum_{x \in \mathbb{F}_2^n} |x\rangle f(x) \right)$$
$$\xrightarrow{H_0 \otimes I_n} \sum_{x \in \mathbb{F}_2^n} \left(\sum_{y \in \mathbb{F}_2^n} (-1)^{\langle x,y \rangle} |y\rangle \right) |f(x)\rangle$$

which is the same as

$$\sum_{y \in \mathbb{F}_2^n} |y\rangle \left(\sum_{x \in \mathbb{F}_2^n} (-1)^{\langle x,y \rangle} |f(x)\rangle \right) . \tag{17}$$

Let $X \subset \mathbb{F}_2^n$ such that $X \cup (s + X) = \mathbb{F}_2^n$ (where $s + X = \{y = s + x, x \in X\}$), we can rewrite Eq. (17)

$$\sum_{y \in \mathbb{F}_2^n} |y\rangle \left(\sum_{x \in X} (-1)^{\langle x,y \rangle} |f(x)\rangle + (-1)^{\langle x+s,y \rangle} |f(x+s)\rangle \right)$$
$$= \sum_{y \in \mathbb{F}_2^n} |y\rangle \left(\sum_{x \in X} (-1)^{\langle x,y \rangle} \left(|f(x)\rangle + (-1)^{\langle s,y \rangle} |f(x)\rangle \right) \right) ,$$

from which it follows that, upon measurement, y is yielded with probability 0 if $\langle s, y \rangle = 1$. In particular, the measurement results are distributed uniformly over the space of the orthogonal space of s. Collecting $O(n)$ such y then allows to compute the space orthogonal to s and, ultimately, s. A more detailed analysis of the algorithm, as well as a generalization to an f that is not necessarily 2-to-1, can be found in [14].

Notice that, despite Eq. (17) being very similar to the state computed by the Bernstein-Vazirani algorithm for the case $m = 1$ (Eq. (3)), there is an important difference: in the latter, the dependence on f of the state is in the amplitude sign, which determines the correspondence with the Walsh-Hadamard transform. Notice that this is a consequence of the fact that the Bernstein-Vazirani routine (Fig. 8) starts with the state $|0^n\rangle\,|-\rangle$ instead of $|0^n\rangle\,|0\rangle$ as in the case of Simon's algorithm, and which results in obtaining the following state after applying $(H_0 \otimes I_1) \circ U_f$

$$\sum_{x\in\mathbb{F}_2^n} (-1)^{\langle x,y\rangle}\,|x\rangle\,(|f(x)\rangle - |f(x)\oplus 1\rangle) = \sum_{x\in\mathbb{F}_2^n} (-1)^{\langle x,y\rangle}(-1)^{f(x)}\,|x\rangle\,|-\rangle$$

since $|f(x)\rangle - |f(x)\oplus 1\rangle = (-1)^{f(x)}\,|-\rangle$.

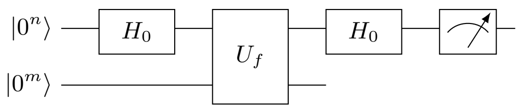


Fig. 8. Bernstein-Vazirani Algorithm

B Periodicity of Functions from Sect. 3.2

In this section, we verify the periods of the functions presented in Sect. 3.2. We also give corresponding circuits in Fig. 9.

B.1 4-Round MISTY R-FK

Recall f_1 from Eq. (4)

$$f_1(x) = ENC_L(x, x\oplus F(x)) \oplus ENC_R(x, x\oplus F(x)).$$

To prove the correctness of the attack on 4-round MISTY R-FK, we only have to show that f_1 indeed has period k_0. To do so, we first notice that, due to the construction of MISTY R-FK, f_1 computes R_3. Therefore, we provide the intermediate values of MISTY R-FK up to R_3 as a function of the input $(x, x\oplus F(x))$.

$$\begin{aligned}
R_1 &= k_0 \oplus F(x)\\
L_1 &= k_0 \oplus F(x) \oplus F(x) \oplus x = x \oplus k_0\\
R_2 &= k_1 \oplus F(x\oplus k_0)\\
L_2 &= k_0 \oplus k_1 \oplus F(x) \oplus F(x\oplus k_0)\\
R_3 &= k_2 \oplus F(k_0 \oplus k_1 \oplus F(x) \oplus F(x\oplus k_0))\\
&= f(x) = f(x\oplus k_0)
\end{aligned}$$

B.2 5-Round MISTY L-FK

Recall f_2 from Eq. (5)

$$f_2(x) = ENC_L(F^{-1}(F^{-1}(x) \oplus x), F^{-1}(x)) \oplus F(x).$$

We now prove that f_2 indeed has period k_0. To begin with, we provide the intermediate values and the output of MISTY L-FK depending on the input (L_0, R_0) and the keys. We omit the values for L_i because $L_{i+1} = R_i$ holds.

$$\begin{aligned}
R_1 &= k_0 \oplus R_0 \oplus F(L_0) \\
R_2 &= k_0 \oplus k_1 \oplus R_0 \oplus F(L_0) \oplus F(R_0) \\
R_3 &= k_0 \oplus k_1 \oplus k_2 \oplus R_0 \oplus F(L_0) \oplus F(R_0) \oplus F(k_0 \oplus R_0 \oplus F(L_0)) \\
R_4 &= k_0 \oplus k_1 \oplus k_2 \oplus k_3 \oplus R_0 \oplus F(L_0) \oplus F(R_0) \\
&\quad \oplus F(k_0 \oplus R_0 \oplus F(L_0)) \oplus F(k_0 \oplus k_1 \oplus R_0 \oplus F(L_0) \oplus F(R_0)) \\
&= ENC_L(L_0, R_0) \\
R_5 &= k_0 \oplus k_1 \oplus k_2 \oplus k_3 \oplus k_4 \oplus R_0 \oplus F(L_0) \oplus F(R_0) \\
&\quad \oplus F(k_0 \oplus R_0 \oplus F(L_0)) \oplus F(k_0 \oplus k_1 \oplus R_0 \oplus F(L_0) \oplus F(R_0)) \\
&\quad \oplus F(k_0 \oplus k_1 \oplus k_2 \oplus R_0 \oplus F(L_0) \oplus F(R_0) \oplus F(k_0 \oplus R_0 \oplus F(L_0))) \\
&= ENC_R(L_0, R_0).
\end{aligned}$$

Now we have

$$\begin{aligned}
f_2(x) &= ENC_L(F^{-1}(F^{-1}(x) \oplus x), F^{-1}(x)) \oplus F(x) \\
&= k_0 \oplus k_1 \oplus k_2 \oplus k_3 \oplus F(k_0 \oplus x) \oplus F(k_0 \oplus k_1) \oplus F(x) \\
&= f_2(x \oplus k_0).
\end{aligned}$$

B.3 4-Round Feistel-FK

Recall f_3 from Eq. (6)

$$f_3(x) = ENC_R(x, \alpha) \oplus F(ENC_L(x, \alpha)) \oplus F(x).$$

To prove the correctness of the attack on 4-round Feistel-FK, we only have to show that f_3 indeed has period $k_0 \oplus F(\alpha)$. To do so, we first notice that, due to the construction of Feistel-FK, f_3 computes

$$R_2 \oplus k_3 \oplus F(x).$$

Next, we express R_2 as a function of the input (x, α):

$$R_2 = k_1 \oplus \alpha \oplus F(x \oplus k_0 \oplus F(\alpha)).$$

Therefore, we have

$$\begin{aligned}
f_3(x) &= k_1 \oplus k_3 \oplus \alpha \oplus F(x \oplus k_0 \oplus F(\alpha)) \oplus F(x) \\
&= f_3(x \oplus k_0 \oplus F(\alpha)).
\end{aligned}$$

B.4 5-Round Feistel-FK with Internal Permutation

Recall f_4 and f_5 from Eqs. (7) and (8)

$$\begin{aligned} f_4(x) &= ENC_R(F(x),x) \oplus F(ENC_L(F(x),x)) \oplus F(x) \\ f_5(x) &= ENC_L(F(F^{-1}(x) \oplus s_4), F^{-1}(x) \oplus s_4) \oplus F(x) \oplus F^{-1}(x) \end{aligned}$$

as well as the auxiliary value

$$h = ENC_R(F(s_4), s_4) \oplus F(ENC_L(F(s_4), s_4)) \oplus F(0).$$

To prove the correctness of the attack, we show that f_4 and f_5 have period s_4 and s_5 resp. and we show that $h = k_0 \oplus k_2 \oplus k_4$ holds. For that purpose, we first provide the intermediate values and the output of Feistel-FK depending on the input (L_0, R_0). We omit the values for L_i because $L_{i+1} = R_i$ holds.

$$\begin{aligned} R_1 &= L_0 \oplus k_0 \oplus F(R_0) \\ R_2 &= R_0 \oplus k_1 \oplus F(L_0 \oplus k_0 \oplus F(R_0)) \\ R_3 &= L_0 \oplus k_0 \oplus F(R_0) \oplus k_2 \oplus F(R_0 \oplus k_1 \oplus F(L_0 \oplus k_0 \oplus F(R_0))) \\ R_4 &= R_0 \oplus k_1 \oplus F(L_0 \oplus k_0 \oplus F(R_0)) \oplus k_3 \\ &\quad \oplus F(L_0 \oplus k_0 \oplus F(R_0) \oplus k_2 \oplus F(R_0 \oplus k_1 \oplus F(L_0 \oplus k_0 \oplus F(R_0)))) \\ &= ENC_L(L_0, R_0) \\ R_5 &= L_0 \oplus k_0 \oplus F(R_0) \oplus k_2 \oplus F(R_0 \oplus k_1 \oplus F(L_0 \oplus k_0 \oplus F(R_0))) \oplus k_4 \\ &\quad \oplus F(R_0 \oplus k_1 \oplus F(L_0 \oplus k_0 \oplus F(R_0)) \oplus k_3 \oplus F(L_0 \oplus k_0 \oplus F(R_0) \oplus k_2 \\ &\qquad \oplus F(R_0 \oplus k_1 \oplus F(L_0 \oplus k_0 \oplus F(R_0))))) \\ &= ENC_R(L_0, R_0) \end{aligned}$$

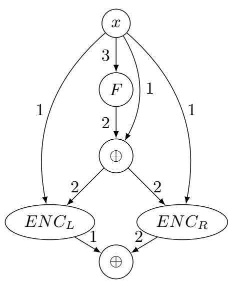
We notice that, due to the structure of Feistel networks, f_4 computes $R_3 \oplus k_4 \oplus F(x)$. For this reason,

$$\begin{aligned} f_4(x) &= ENC_R(F(x),x) \oplus F(ENC_L(F(x),x)) \oplus F(x) \\ &= F(x) \oplus k_0 \oplus F(x) \oplus k_2 \oplus F(x \oplus k_1 \oplus F(F(x) \oplus k_0 \oplus F(x))) \oplus k_4 \oplus F(x) \\ &= k_0 \oplus k_2 \oplus k_4 \oplus F(x \oplus k_1 \oplus F(k_0)) \oplus F(x) \end{aligned}$$

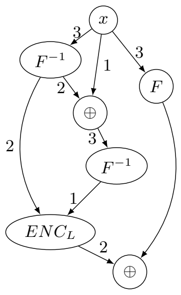
has indeed period $s_4 = F(k_0) \oplus k_1$. Notice that h is similar to $f_4(s_4)$, but we replace the last $F(x)$ by $F(0)$ s.t. both F terms cancel out and only $h = k_0 \oplus k_2 \oplus k_4$ remains. For f_5 we have

$$\begin{aligned} f_5(x) &= ENC_L(F(F^{-1}(x) \oplus s_4), F^{-1}(x) \oplus s_4) \oplus F(x) \oplus F^{-1}(x) \\ &= s_4 \oplus k_1 \oplus F(k_0) \oplus k_3 \oplus F(k_0 \oplus k_2 \oplus F(F^{-1}(x) \oplus s_4 \oplus k_1 \oplus F(k_0))) \oplus F(x) \\ &= k_3 \oplus F(x \oplus k_0 \oplus k_2) \oplus F(x) \end{aligned}$$

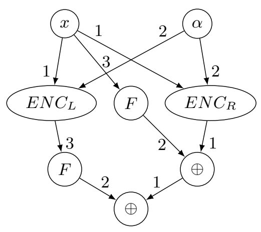
and thus f_5 has period $s_5 = k_0 \oplus k_2$. This concludes the proof of the correctness of our attack on 5-round Feistel-FK with a bijective round function.



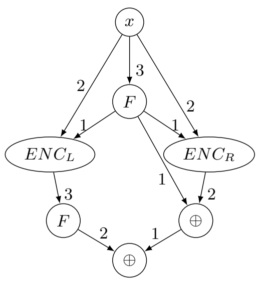
(a) Circuit C_{f_1}.



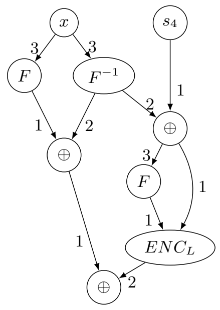
(b) Circuit C_{f_2}.



(c) Circuit C_{f_3}.



(d) Circuit C_{f_4}.



(e) Circuit C_{f_5}.

Fig. 9. Circuits for attacks on MISTY and Feistel.

C Proof of Proposition 1

Proposition 1. *Let H and H' be two equivalent Hadamard matrices. There exist functions* $\texttt{PRE} : \mathcal{B}_n \to \mathcal{B}_n$ *and* $\texttt{POST} : \mathbb{F}_2^n \to \mathbb{F}_2^n$ *such that, for any* $f : \mathbb{F}_2^n \to \mathbb{F}_2$

$$\Pr(x \leftarrow \texttt{Simon}(H')(f)) = \Pr(x \leftarrow \texttt{POST}(\texttt{Simon}(H)(\texttt{PRE}(f))).$$

Proof. Let D be a diagonal matrix of inputs $-1, 1$ and let d be the Boolean function associated to D, i.e. d is such that $(D)_{x,x} = (-1)^{d(x)}$. Let P be a permutation matrix and $\pi : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be the permutation associated to P such that $\pi(x) = y$ if and only if $Pe_x = e_y$. We have that

i) if $H' = DH$, then
$$\widehat{f}^{H'}(x) = (H'\phi)_x = (DH\phi)_x = (D)_{x,x}\widehat{f}^H(x);$$

ii) if $H' = HD$, then
$$\widehat{f}^{H'}(x) = (H'\phi)_x = (HD\phi)_x = \widehat{f+d}^H(x)$$
where $(D\phi)_i \coloneqq (-1)^{f(x)+d(x)}$;

iii) if $H' = HP$, then
$$\widehat{f}^{H'}(x) = (H'\phi)_x = (HP\phi)_x = \widehat{f \circ \pi^{-1}}^H(x)$$
since $(P\phi)_x = (-1)^{f\circ\pi^{-1}(x)}$;

Let us now consider the diagonal matrices D_1, D_2 and permutation matrices P_1, P_2 such that
$$H' = P_1 D_1 H D_2 P_2,$$
together with the associated functions, respectively $d_1, d_2 : \mathbb{F}_2^n \to \mathbb{F}_2$ and $\pi_1, \pi_2 : \mathbb{F}_2^n \to \mathbb{F}_2^n$, as defined above. Then from *i)*-*iii)* it follows that for any Boolean function f, we can define $g \coloneqq f \circ \pi_2^{-1} + d_2$ such that
$$\widehat{g}^H(\pi_1^{-1}(x)) = (-1)^{d_1(x)}\widehat{f}^{H'}(x)$$

This shows that for two fixed equivalent Hadamard matrices H and H', it is possible to easily retrieve the same output of $\texttt{Simon}(H')$ with access to only $\texttt{Simon}(H)$ on any f by (classical) pre- and post-processing.

Indeed, an output x of this routine is such that $\widehat{g}^H(\pi_1^{-1}(x)) = \pm\widehat{f}^{H'}(x) \neq 0$ and would be returned with a probability $\left(\widehat{g}^H(\pi_1^{-1}(x))\right)^2$. This is exactly the same as $\texttt{Simon}(H')$, which indeed returns such an x with a probability
$$2^{-n/2}(\widehat{f}^{H'}(x))^2 = 2^{-n/2}\left(\widehat{g}^H(\pi_1^{-1}(x))\right)^2.$$

This concludes the proof with $\texttt{POST}(x) = \pi_1(x)$ and $\texttt{PRE}(f(x)) = f(\pi_2^{-1}(x)) + d_2(x)$. □

D Proof of Proposition 3

In this section we prove the following result, that is equivalent to Proposition 3.

Proposition 4. *Let $g : \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2^n$ such that for any $A \in \mathrm{GL}(n, \mathbb{F}_2)$*

$$g(Ax, y) = g(x, A^T y) \tag{18}$$

for all $x, y \in \mathbb{F}_2^n$. Then

$$g = \varepsilon_0 + \varepsilon_1 g_1 + \varepsilon_2 g_2 + \varepsilon_3 g_3 + \varepsilon_4 g_4$$

where $\varepsilon_0, \dots, \varepsilon_4 \in \mathbb{F}_2$ and

$$\begin{aligned}
g_1(X, Y) &= \sum_{u \in \mathbb{F}_2^n \setminus \{0\}} X^u \\
g_2(X, Y) &= \sum_{v \in \mathbb{F}_2^n \setminus \{0\}} Y^v \\
g_3(X, Y) &= \langle X, Y \rangle \\
g_4(X, Y) &= \sum_{(u,v) \in \mathbb{F}_2^n \setminus \{(0,0)\}} X^u Y^v
\end{aligned}$$

In order to do so, we will work on the ANF of g. For any boolean function $g : \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2^n$, let $\lambda_{u,v}$ for $u, v \in \mathbb{F}_2^n$ represent its ANF, i.e. $g(X, Y) = \sum_{u,v} \lambda_{u,v} X^u Y^v$. For more clarity, we define the function $\lambda : \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2$ as

$$\lambda(u, v) = \lambda_{u,v}.$$

Notice that the left multiplication of $X = (X_1, \dots, X_n)$ by A defines a function $\mathbb{F}_2^n[X] \to \mathbb{F}_2^n[X]$ such that if $A = (a_{i,j})_{1 \le i,j \le n}$, then

$$(AX)_i = \sum_{1 \le j \le n} a_{i,j} X_j.$$

A difficulty of the proof of this result is arguably not being able to say much about A^T as an isomorphism for a generic $A \in \mathrm{GL}(n, \mathbb{F}_2)$.

However, if we consider a subspace $U, W \subset \mathbb{F}_2^n$ generated by a subset of the standard basis of $\mathbb{F}_2^n$ and such that $\mathbb{F}_2^n = W + U$, then for any $A \in \mathrm{GL}(n, \mathbb{F}_2)$ for which $A|_U$ is the identity, we have that $A^T|_U$ is also the identity. Therefore, since U is generated by elements of the standard basis, we will have that $(AX)^u = (A^T X)^u = X^u$ for any $u \in U$. This, together with the fact that matrices of this form generate $\mathrm{GL}(n, \mathbb{F}_2)$, will lead to a restriction of the problem to the case of subspaces of lower dimension, thus allowing to conclude with an inductive reasoning.

Indeed, in order to show the proposition, we will only consider A as being an element of the following set of generators for $\mathrm{GL}(n, \mathbb{F}_2)$:

– A is such that $Ae_i = e_i + e_j$ for some $i \neq j$ and $Ae_l = e_l$ for $l \neq i$;
– A is such that it permutes two vectors of the basis, i.e. $Ae_i = e_j$, $Ae_j = e_i$ for some $i \neq j$ and $Ae_l = e_l$, $l \neq i, j$.

Let us now fix one of these A and let now U be the space where A acts as the identity and W its dual.

Consider first, for some fixed $\tilde{u}, \tilde{v} \in U$, the polynomial

$$h(X, Y) = X^{\tilde{u}} Y^{\tilde{v}} \sum_{w,w' \in W} \lambda_{w,w'} X^w Y^{w'} \tag{19}$$

we have that $h(AX, Y) = h(X, A^T Y)$ if and only if

$$\sum_{w,w' \in W} \lambda_{w,w'} (AX^w) X^{\tilde{u}} Y^{w'} = \sum_{w,w' \in W} \lambda_{w,w'} X^w (A^T Y^{w'}). \tag{20}$$

Intuitively, since g is a sum of polynomials of the form of h, in order for g to satisfy Eq. (18) for a fixed A, each of this h must also satisfy it because they behave independently. We will see that we can study Eq. (18) by focusing on the polynomials of the form given by Eq. (19) that compose the ANF of g, that need to satisfy the somewhat simpler Eq. (20). With the aim of making this reduction formal, we introduce the following notation that allows us to isolate such components of the ANF of g.

For any $\tilde{u}, \tilde{v}$ in the vector space U that is fixed by A, we introduce the Boolean functions $g_{\widetilde{u},\widetilde{v}}(X, Y)$ as

$$g_{\widetilde{u},\widetilde{v}}(X, Y) := \sum_{w,w' \in W} \lambda(\widetilde{u} + w, \widetilde{v} + w') X^w Y^{w'}$$

The function $g_{\widetilde{u},\widetilde{v}}$ is the *unique* (as a consequence of the uniqueness of the ANF of boolean functions) polynomial that is obtained when grouping by $X^{\widetilde{u}} Y^{\widetilde{v}}$ the terms of the ANF of g of the form $X^u Y^v$, where $u \in \tilde{u} + W, v \in \tilde{v} + W$. In fact:

$$\begin{aligned}
g(X, Y) &= \sum_{u,v \in \mathbb{F}_2^n} \lambda(u, v) X^u Y^v = \sum_{u,v \in (\widetilde{u}+W, \widetilde{v}+W)} \lambda(\widetilde{u} + w, \widetilde{v} + w') X^{\widetilde{u}+w} Y^{\widetilde{v}+w'} \\
&= \sum_{\widetilde{u},\widetilde{v} \in U} X^{\widetilde{u}} Y^{\widetilde{v}} \left(\sum_{w,w' \in W} \lambda(\widetilde{u} + w, \widetilde{v} + w') X^w Y^{w'} \right) \\
&= \sum_{\widetilde{u},\widetilde{v} \in U} g_{\widetilde{u},\widetilde{v}}(X, Y) X^{\widetilde{u}} Y^{\widetilde{v}}.
\end{aligned}$$

As a concrete example, if we consider the case where

$$g(X, Y) = X_1 X_2 Y_1 Y_2 + X_1 X_2 + Y_1 Y_2 + 1$$

and let $W = \langle e_1 \rangle$ and $U = \langle e_2 \rangle$, we have that $g_{0,0}(X, Y) = 1$, $g_{e_2,0}(X, Y) = X_1$, $g_{0,e_2}(X, Y) = Y_1$, $g_{e_2,e_2}(X, Y) = X_1 Y_1$.

Notice that if g satisfies Eq. (18) for a fixed $A \in \mathrm{GL}(n, \mathbb{F}_2)$, then $g_{\tilde{u},\tilde{v}}$ must satisfy it too for any $\tilde{u}, \tilde{v} \in U$, because the factor $X^{\widetilde{u}}Y^{\widetilde{v}}$ is fixed by A and A^T.

To show this, we introduce the following notation that will also be useful later: for any two given functions h and g with ANF, respectively, κ_{uv} and λ_{uv} we will say that $h \preceq g$ if and only if $\kappa_{uv} \leq \lambda_{uv}$ for all $u, v \in \mathbb{F}_2^n$.

Consider then, for a g that solves Eq. (18), the function $g^A : (x, y) \mapsto g(Ax, y) = g(x, A^T y)$. Then, since $(AX)^{\widetilde{u}}Y^{\widetilde{v}} = X^{\widetilde{u}}(A^T Y)^{\widetilde{v}} = X^{\widetilde{u}}Y^{\widetilde{u}}$, from the first equality we have that

$$g^A_{\widetilde{u},\widetilde{v}}(X, Y) = g_{\widetilde{u},\widetilde{v}}(AX, Y),$$

while from the second equality we get

$$g^A_{\widetilde{u},\widetilde{v}}(X, Y) = g_{\widetilde{u},\widetilde{v}}(X, A^T Y).$$

But the uniqueness of $g^A_{\widetilde{u},\widetilde{v}}$ implies then that

$$g_{\widetilde{u},\widetilde{v}}(AX, Y) = g_{\widetilde{u},\widetilde{v}}(X, A^T Y).$$

This means that we can effectively solve Eq. (18) in lower dimension by looking at $g_{\tilde{u},\tilde{v}}$. Indeed, we will see that understanding the behaviour of such $g_{\widetilde{u},\widetilde{v}}$ for $\tilde{u}, \tilde{v} \in U$ such that $\dim U \geq n - 3$ (i.e. W is of dimension up to 3) is enough to conclude for the entire g. Let us therefore solve Eq. (18) in this hypothesis.

Lemma 3. *Let $1 \leq i, j, k \leq n$. Consider U the complement space of $W := \langle e_i, e_j, e_k \rangle$. Let $g : \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2^n$ be a function that satisfies Eq.* (18)*. Then, for all $\widetilde{u}, \widetilde{v} \in U$ we have that*

$$g_{\widetilde{u},\widetilde{v}}(X, Y) = \varepsilon_0 + \varepsilon_1 h_1 + \varepsilon_2 h_2 + \varepsilon_3 h_3 + \varepsilon_4 h_4$$

where $\varepsilon_0, \ldots, \varepsilon_4 \in \mathbb{F}_2$ and

$$\begin{aligned}
h_1(X, Y) &= \sum_{w \in W \setminus \{0\}} X^w \\
h_2(X, Y) &= \sum_{w \in W \setminus \{0\}} Y^w \\
h_3(X, Y) &= \sum_{1 \leq l \leq n} X_l Y_l \\
h_4(X, Y) &= \sum_{(w,w') \in W \times W \setminus \mathcal{W}} X^w Y^{w'} \\
&= \sum_{(w,w') \in W \times W \setminus \{(0,0)\}} X^w Y^{w'} + h_1(X, Y) + h_2(X, Y) + h_3(X, Y)
\end{aligned}$$

where

$$\mathcal{W} := \{(w, w') \in W \times W \setminus \{(0, 0)\} : w = 0 \text{ or } w' = 0\} \cup \{(e_l, e_l) : 1 \leq l \leq n\}.$$

Proof. We have seen that if g satisfies Eq. (18), then $g_{\tilde{u},\tilde{v}}$ also does. Therefore, we can effectively reduce the problem on $g : W \times W \to \mathbb{F}_2$, i.e. to $\mathbb{F}_2^m \simeq W$, for $m = \dim W \leq 3$. For these low dimensions, the solution to the equation can be found with the help of a computer, finally yielding exactly the functions of the thesis.

□

We are now ready to prove the final result. The main idea of the proof is to choose appropriately the space U (or, equivalently, its dual W) and the $\tilde{u}, \tilde{v} \in U$ so that, by looking at the simpler $g_{\tilde{u},\tilde{v}}$ of which we know the form thanks to the previous lemma, we can prove that a certain monomial must appear in the ANF of a g that solves Eq. (18). We remark that imposing that $g_{\tilde{u},\tilde{v}}$ satisfy the above lemma for a choice of U (or W) is equivalent to imposing that g solves Eq. (18) for any A such that $A|_U$ is the identity.

Proof of Proposition 4. Let us first remark that $g = \varepsilon_0 + \varepsilon_1 g_1 + \varepsilon_2 g_2 + \varepsilon_3 g_3 + \varepsilon_4 g_4$ for some $\varepsilon_i \in \mathbb{F}_2$ if and only if there exist $\varepsilon_0', \dots, \varepsilon_4' \in \mathbb{F}_2$ such that

$$g = \varepsilon_0' + \varepsilon_1' g_1 + \varepsilon_2' g_2 + \varepsilon_3' g_3 + \varepsilon_4' g_4' \tag{21}$$

where $g_4'(X,Y) = g_4(X,Y) + g_1(X,Y) + g_2(X,Y) + g_3(X,Y)$. For the rest of the proof, it will be more convenient to work on this other set of generators $\{1, g_1, g_2, g_3, g_4'\}$ because they have complementary ANF coefficients and we can divide the proof in four distinct cases.

Indeed, as noticed in Remark 1, if g is of the form of Eq. (21), then it indeed satisfies the hypothesis. The converse holds if we show that

1) if there exists $\mu \neq 0$ such that $X^\mu \preceq g$, then $g_1(X,Y) = \sum_{u \in \mathbb{F}_2^n \setminus \{0\}} X^u \preceq g$
2) if there exists $\nu \neq 0$ such that $Y^\nu \preceq g$, then $g_2(X,Y) = \sum_{v \in \mathbb{F}_2^n \setminus \{0\}} Y^v \preceq g$
3) if there exists $1 \leq k \leq n$ such that $X^k Y^k \preceq g$, then $g_3(X,Y) = \sum_{1 \leq i \leq n} X_i Y_i \preceq g$
4) if there exist $\mu, \nu \in \mathbb{F}_2^n$ such that for some $1 \leq i < j \leq n$ we have $\mu_i = \nu_j = 1$ and $X^\mu Y^\nu \preceq g$, then $g_4'(X,Y) \preceq g$.

1-2) Consider $h(X) \coloneqq g(X,0)$. It is clear that for any $A \in \mathrm{GL}(n, \mathbb{F}_2)$

$$h(AX) = h(X)$$

and $X^u \preceq g(X,Y) \iff X^u \preceq h(X)$. Therefore, let us prove that if $X^\mu \preceq h(X)$, then $\sum_{u \in \mathbb{F}_2^n \setminus \{0\}} X^u \preceq h$. Since h is not the constant function, let $u_1 \neq 0$ such that $h(u_1) = 1 + h(0)$. Then for any $u \in \mathbb{F}_2^n \setminus \{0\}$, we may consider the matrix $A_u \in \mathrm{GL}(n, \mathbb{F}_2)$ such that $Au_1 = u$ and we have

$$h(u) = h(A_u u_1) = h(u_1) = 1 + h(0)$$

for all $u \neq 0$, i.e. $h(X) = \sum_{u \in \mathbb{F}_2^n \setminus \{0\}} X^u + h(0)$. The same reasoning can be done for *ii*), considering $h(Y) \coloneqq g(0,Y)$.

3) Let $j \neq k$ an integer between 1 and n. Then Lemma 3 with $W = \langle e_j, e_k \rangle$ and $\tilde{u} = \tilde{v} = 0$ implies that, since $X_k Y_k \preceq g$, then $X_k Y_k \preceq g_{0,0}(X, Y)$ and $g_{0,0}$ is of the form h_3, i.e. $g_{0,0}(X, Y) = X_j Y_j + X_k Y_k$. Therefore $X_j Y_j \preceq g$ for any $j \neq k$.

4) We want to show that for all $u, v \in \mathbb{F}_2^n$ such that $u_i = v_j = 1$ for some $i \neq j$, then $X^u Y^v \preceq g$. In particular, we are going to show that

a) if $\mu = e_i$ and $\nu = e_j$ for some $i \neq j$, then $X_k Y_j + X_i X_k Y_j \preceq g$ for all $k \neq j$;
b) if $\mu = e_i$ and $\nu = e_j$ for some $i \neq j$, then $X_k Y_h \preceq g$ for any $1 \leq k < h \leq n$, $k \neq h$.
c) Let $\mathrm{wt}(u)$ be the number of non-zero components of u. If $2 \leq \mathrm{wt}(\mu) \leq n-1$, then $X^u Y^\nu \preceq g$ for any $u \in \mathbb{F}_2^n$ such that $\mathrm{wt}(u) = \mathrm{wt}(\mu)$.
d) Additionally, if $2 \leq \mathrm{wt}(\mu) \leq n-1$, there exist $u_1, u_2 \in \mathbb{F}_2^n \setminus \{0\}$ such that $X^{u_1} Y^\nu + X^{u_2} Y^\nu \preceq g$ with $\mathrm{wt}(u_1) + 1 = \mathrm{wt}(u) = \mathrm{wt}(u_2) - 1$.

From these four facts, it will follow for induction on the weight of u that

– if $\nu = e_j$ then $X^u Y^\nu \preceq g$ for any $u \notin \{0, e_j\}$ and
– if $\nu \neq e_j$, then $X^u Y^\nu$ for any $u \neq 0$.

The same reasoning applied to $X^\mu Y^v$ for $v \in \mathbb{F}_2^n \setminus \{0\}$ will conclude the proof.

Therefore, let $1 \leq k \leq n$ (possibly $k = i$ or $k = j$) and consider $W := \langle e_i, e_j, e_k \rangle$ and the projections of μ, ν onto its dual U, i.e. $\widetilde{\mu} = \mu + e_i + \mu_j e_j + \mu_k e_k \in U$ and $\widetilde{\nu} = \nu + \nu_i e_i + e_j + \nu_k e_k \in U$. Then we have that $X_i Y_j \preceq g_{\widetilde{\mu}\widetilde{\nu}}$ so that Lemma 3 implies that $g_{\widetilde{\mu}\widetilde{\nu}}(X, Y) = h_4(X, Y)$ and, in particular,

$$h_4(X, Y) X^{\widetilde{\mu}} Y^{\widetilde{\nu}} \preceq g(X, Y) \tag{22}$$

From Eq. (22), if $k \neq i, j$ and $\mu = e_i, \nu = e_j$ (and therefore $\tilde{\mu} = \tilde{\nu} = 0$) we have:

$$X_k Y_j + X_i X_k Y_j \preceq g_{0,0}(X, Y) \preceq g,$$

for any $k \neq i, j$, which proves *a*).

Furthermore, if we consider, for any $h \neq j$, $W = \langle e_j, e_h \rangle$, since $X_i Y_j \preceq g$, we have that $Y_j \preceq g_{e_i,0}$, and therefore $g_{e_i,0} = Y_j + Y_h + Y_j Y_h$ from the lemma, which finally implies that $X_i Y_h \preceq g$. Applying *a*) to each $X_i Y_h$ for every h, proves that for any h, $X_k Y_h \preceq g$ for all $k \neq h$, which proves *b*).

To prove *c*) notice that from Eq. (22) it follows that, for any $k \neq i$, then $X_i X_k Y_j \preceq h_4(X, Y)$, and therefore $X_i X_k X^{\widetilde{\mu}} Y^\nu \preceq g$ for any $k \neq i$. Furthermore, for any $k \neq i$, since $X_i X_k X^{\widetilde{\mu}} Y^\nu \preceq g$, we can prove that for any $h \neq k$

$$X_h X_k X^{\widetilde{\mu}} Y^\nu \preceq g$$

with the same reasoning applied to $W = \langle e_i, e_k, e_h \rangle$. Therefore, this proves that $X^{e_k + e_h + \tilde{\mu}} Y^\nu \preceq g$ for any $1 \leq h < k \leq n$. If $\tilde{\mu} = 0$, then we would be finished. Otherwise, apply the same reasoning for the other $i \neq j$ such that $\mu_i = 1$ to get that $X^u Y^\nu \preceq g$ for any u such that $\mathrm{wt}(u) = \mathrm{wt}(\nu)$.

For *d*), we use Eq. (22) and find that, when $k \neq i, j$, since $X_i X_k Y_j \preceq h_4(X, Y)$, then $X_i X_k X^{\widetilde{\mu}} Y^\nu = \preceq g$. Therefore, if we choose k such that $\mu_k = 0$, this means that $u_2 = e_k + \mu$ is of greater weight thatn μ and such that $X^{u_2}Y^{\nu} \preceq g$. On the other hand, if we let $l \neq i$ such that $e_l = 1$ and consider $W = \langle e_i, e_l, e_j \rangle$ and $\widetilde{\mu}, \widetilde{\nu}$ the projection of $\widetilde{\mu}, \widetilde{\nu}$ onto W. Then, since $X_i X_l Y_j Y_l^{\nu_l} \preceq g_{\widetilde{\mu},\widetilde{\nu}}$, then also $X_i Y_j Y_l^{\nu_l} \preceq g_{\widetilde{\mu},\widetilde{\nu}}$ and conclude because $X^{\mu - \mu_l e_l} Y^{\nu} \preceq g$, so that we have found u_1.

$\square$