Quantum Linearization Attacks

Xavier Bonnetain[1,2], Gaëtan Leurent[2], María Naya-Plasencia[2], and André Schrottenloher[3]

[1] Institute for Quantum Computing, Department of Combinatorics and Optimization, University of Waterloo, Waterloo, ON, Canada
[2] Inria, Paris, France
[3] Cryptology Group, CWI, Amsterdam, The Netherlands

Abstract. Recent works have shown that quantum period-finding can be used to break many popular constructions (some block ciphers such as Even-Mansour, multiple MACs and AEs...) in the superposition query model. So far, all the constructions broken exhibited a strong algebraic structure, which enables to craft a periodic function of a single input block. Recovering the secret period allows to recover a key, distinguish, break the confidentiality or authenticity of these modes.

In this paper, we introduce the *quantum linearization attack*, a new way of using Simon's algorithm to target MACs in the superposition query model. Specifically, we use inputs of multiple blocks as an interface to a function hiding a linear structure. Recovering this structure allows to perform forgeries.

We also present some variants of this attack that use other quantum algorithms, which are much less common in quantum symmetric cryptanalysis: Deutsch's, Bernstein-Vazirani's, and Shor's. To the best of our knowledge, this is the first time these algorithms have been used in quantum forgery or key-recovery attacks.

Our attack breaks many parallelizable MACs such as LightMac, PMAC, and numerous variants with (classical) beyond-birthday-bound security (LightMAC+, PMAC+) or using tweakable block ciphers (ZMAC). More generally, it shows that constructing parallelizable quantum-secure PRFs might be a challenging task.

Keywords: Quantum cryptanalysis, MACs, superposition query model, Deutsch's algorithm, Bernstein-Vazirani algorithm, Simon's algorithm, Shor's algorithm.

1 Introduction

The possible emergence of large-scale quantum computing devices in a near future has prompted a wide move towards *post-quantum security*, which takes into account the new security threats that they pose. In particular, the most popular asymmetric cryptosystems currently in use, such as RSA, can be broken by an adversary capable of successfully implementing Shor's algorithm [59]. An ongoing standardization project led by the NIST [56] has structured the efforts of the (asymmetric) cryptographic community on this question.

As symmetric primitives do not rely on a trapdoor, they seemed for a long time to avoid the cases where quantum computers bring an exponential speedup over the best classical algorithms. In fact, most problems in symmetric cryptography, such as the search for the secret key of a black-box cipher, seem to admit a quadratic speedup at best, given by Grover's quantum search algorithm [29]. Although this speedup is significant, it could be countered by increasing the parameters of symmetric cryptosystems, e.g., doubling the size of secret keys.

However, in the past few years, a series of works have shown the insecurity of some symmetric cryptosystems against quantum adversaries entitled to *superposition queries*. That is, some primitives become broken if they can be queried inside a quantum algorithm. This started with the 3-round Feistel distinguisher proposed by Kuwakado and Morii [44]. Later, they found a polynomial-time key-recovery attack on the Even-Mansour cipher [45], which was the first quantum key-recovery on a classically secure symmetric construction. These results rely crucially on the fact that many popular designs in symmetric cryptography have a strong algebraic structure, as they are built by combining smaller primitives (such as permutations or block ciphers) using cheap operations such as XORs. Kaplan *et al.* [39] showed that many other constructions exhibited a structure exploitable by a quantum adversary, and designed the first forgery attacks on MACs (notably CBC-MAC [11], OMAC [35], PMAC [12]) and authenticated encryption schemes (e.g., OCB3 [43], GCM [49]).

In this paper, we will focus on idealized MAC constructions that authenticate messages of arbitrary size using smaller primitives such as permutations, block ciphers or tweakable block ciphers (TBCs) of block size n. These constructions have classical proofs of security showing either that the MAC behaves as a pseudo-random function, or that it is unforgeable, up to some exponential bound in n. We will exhibit polynomial-time quantum attacks on constructions that were not vulnerable to previous Simon's attacks (like those of [39,58]).

Previous Attacks. Although there have been many of them, all the quantum forgery attacks known so far follow the same paradigm. They query the MAC with a constant number of blocks, using usually a single block of message x in superposition. Inside the MAC, this block of message x is XORed to some unknown value α depending on other blocks: thus, the result is $\mathsf{MAC}(x \oplus \alpha)$. Having two different values α_0, α_1, we then have access to two functions $f(x) = \mathsf{MAC}(x \oplus \alpha_0)$ and $g(x) = \mathsf{MAC}(x \oplus \alpha_1)$, such that $f(x) = g(x \oplus \alpha_0 \oplus \alpha_1)$. From there, we can use Simon's Boolean hidden shift algorithm [60] as a black box. It recovers $\alpha_0 \oplus \alpha_1$ in quantum polynomial time, whereas any classical algorithm would require exponentially many queries to f and g (thus to the MAC). The recovery of the internal shift $\alpha_0 \oplus \alpha_1$ then enables the adversary to forge new messages, and in some cases to recover secret-key material.

Let us point out the following important remark:

> If the message blocks are not directly XORed to internal values (keys, offsets, encryption of other blocks. . .), then the previous attacks based on Simon's algorithm do not apply.

Contributions. In this paper, we present the *quantum linearization attack*, which is a new family of quantum attacks on classically unforgeable MACs when superposition queries are allowed. Thanks to the novelty of our approach, we are able to attack many MACs that resisted previous cryptanalysis, as they do not not exhibit the property recalled above (a message block XORed to an internal state value). In particular, our attack usually circumvents the use of TBCs instead of block ciphers. It is also the first case of a quantum polynomial-time attack on MACs with *beyond-birthday* security, where the internal state has a bigger size. As an example, we break LightMAC with a linear number of queries, and we can attack LightMAC+ with only twice as much.

Overview. Our attack starts with the following remark. Consider a function of ℓ blocks $x_1, \dots, x_\ell$ of the form: $G(x_1, \dots, x_\ell) = g_1(x_1) \oplus \dots \oplus g_\ell(x_\ell) \oplus C$, where C is an independent constant, and the g_i are independent random functions to which the adversary *does not have access*. Then classically, this function cannot be distinguished from random with a single query, though as little as four would be enough: we make $x_3, \dots, x_\ell$ constant, we query for every $x_1 \in \{0, 1\}$ and $x_2 \in \{0, 1\}$: the XOR of the four results is zero.

Our key idea is to *linearize* the function G by restricting the block inputs so that the output is an affine function. Similarly to the simple classical distinguisher, we make the blocks $x_1, \dots, x_\ell$ take only one-bit values and emulate a function of an ℓ-bit input: $F(x) = F(b_1\| \dots \|b_\ell) = G(0^{n-1}\|b_1, \dots, 0^{n-1}\|b_\ell)$. Now, we will remark that F is an *affine* function of $b_1, \dots, b_\ell$. As the g_i are XORed; flipping a bit b_i in the input XORs $g_i(0) \oplus g_i(1)$ to the output.

It is well known that the Bernstein-Vazirani algorithm allows to distinguish an affine function from a random one with a *single* quantum query. This shows that, thanks to a multi-block input, we can access new vulnerabilities of cryptographic constructions. But the power of our attack is clearly demonstrated when we make G go through a new random function:

$$G'(x_1, \dots, x_\ell) = g(G(x)) = g\left(g_1(x_1) \oplus \dots \oplus g_\ell(x_\ell) \oplus C\right) \quad . \tag{1}$$

All the functions $g_1, \dots, g_\ell, g$ are unknown to the adversary, so she cannot find the affine structure of the internal G. In fact, this function would be classically secure as a MAC. However, when linearizing, we obtain: $G'(x) = g(F(x))$ where F is an affine function of $x = b_1\| \dots \|b_\ell$. Thus, G' embeds a *hidden Boolean period*, and Simon's algorithm can recover it in polynomial time.

Applications. In Section 4 and Section 5, we detail the applications of our algorithm. We obtain the first polynomial-time attacks against the following MACs:

ΘCB3 [57,43], LightMAC [47], LightMAC+ [54], Deoxys [38],
ZMAC [37], PMAC_TBC3k [53], PolyMAC [36], GCM-SIV2 [36]

In addition, we provide attacks on the XOR-MACs of [4], on MACs based on universal hashing (e.g., NMH* [31] and BRW Hashing [7]) and, in Section 5.5, a new superposition forgery attack against Poly1305 [6]. A previous quantum attack was given in [18], using a hidden shift structure. Using Shor's algorithm instead, we reduce the number of superposition queries from 2^{38} to about 32.

On Parallelizable MACs. The quantum linearization attack leaves only little space for quantum-secure parallelizable PRFs. Indeed, we are able to break any PRF with extendable domain, where at least $\geq n$ independent input blocks of $\leq n$ bits are processed independently, then XORed. This works as well for any operation that is linear on $(\mathbb{F}_2)^n$. It is still possible to obtain an unforgeable IV- or nonce-based MAC of this form, as shown in [9], but the security then relies on the non-repetition of IVs. We do not know if an attack applies when we use a modular addition instead of a XOR in (1). If this was the case, then it would clearly mean that one has to rely on sequentiality or on nonlinear operations.

Organization. We start in Section 2 by reviewing some quantum computing notions, the quantum algorithms used in this paper (Deutsch's algorithm, Bernstein-Vazirani, Simon's algorithm, Shor's algorithm), the Q1 / Q2 attack scenarios and notions of quantum unforgeability. In Section 3, we detail our new algorithmic ideas. In Section 4, we apply our attack to many parallelizable MAC constructions. We dedicate Section 5 to MACs based on universal hashing. We discuss the implications of our attacks in Section 6 and conclude the paper in Section 7.

2 Preliminaries

In this section, we give some preliminaries about quantum computing, quantum attacker models and the well-known quantum algorithms that will be used throughout this paper. We elaborate about the Q2 attacker model and the notion of quantum unforgeability for MACs, with or without IVs. Note that some details of quantum computing will appear in this section. They are intended for the interested reader. In the rest of this paper (with the exception of Section 5.5), we will use the algorithms of this section as black boxes.

2.1 Notation

We consider n-bit string values, sometimes as elements of $\mathbb{F}_{2^n}$, sometimes as elements of $\mathbb{F}_2^n$. This shall be clear from context. We let $\oplus$ denote the XOR (addition in $\mathbb{F}_2^n$), $\odot$ denote multiplication in $\mathbb{F}_{2^n}$, and $+$ modular addition. We let $\cdot$ denote the scalar product of bit-strings seen as n-bit vectors.

2.2 On Quantum Computing

Although we choose to present in detail the quantum algorithms that we will use for our attacks, most of our results can be obtained by applying them as black boxes. Thus we stress that our results, similarly as other structural attacks on symmetric cryptosystems [45,39], can be understood from a high-level perspective, and our attacks do not require specific knowledge of quantum computing. Further details are only required to prove the correctness of the algorithms.

A general presentation of the *quantum circuit model* can be found in [55]. The basic computation units are qubits, two-level quantum systems whose state is represented by a *superposition* $\alpha |0\rangle + \beta |1\rangle$, with amplitudes α and β, which is a normalized vector in $\mathbb{C}^2$ (of norm $|\alpha^2| + |\beta^2| = 1$). The state of an n-qubit system belongs to $\mathbb{C}^{2^n}$, its 2^n basis vectors (in the computational basis) are the 2^n n-bit strings.

A quantum algorithm is a sequence of unitary operators of $\mathbb{C}^{2^n}$, partial measurements, and oracle calls. We say that a function f is queried *in superposition* if the following unitary operator O_f is made available: $|x\rangle |y\rangle \mapsto |x\rangle |y \oplus f(x)\rangle$. Indeed, this operator allows to query f on any quantum state, thus on any *superposition* of inputs x. This is the *standard* oracle, equivalent to the *phase oracle* $O_{f,\pm}$ which computes $|x\rangle \mapsto (-1)^{f(x)} |x\rangle$.

One of the basic unitary operations of the quantum circuit model (*quantum gates*), and actually the most important one in the algorithms of Section 2.3, is the Hadamard gate H which maps a single qubit $|b\rangle$ to $\frac{1}{\sqrt{2}}\left(|0\rangle + (-1)^b |1\rangle\right)$. By applying Hadamard gates to each individual qubit of an n-bit input, we compute the *Hadamard transform*, a particular example of Quantum Fourier Transform:

$$H^{\otimes n} : |x\rangle \mapsto \frac{1}{2^{n/2}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle \quad .$$

An important property is that the Hadamard transform is involutive. For better readability, we often omit global amplitude factors such as the $\frac{1}{2^{n/2}}$ above, as quantum states are always normalized.

Given a quantum state of the form $\sum_x \alpha_x |x\rangle$, the measurement operation destroys the state and yields an element x with probability $|\alpha_x|^2$. Partially measuring the state *projects* it on a smaller superposition of elements. For a quantum state of the form: $\sum_{x,y} \alpha_{xy} |x\rangle |y\rangle$, measuring the register $|x\rangle$ yields a value x_0 with probability $\sum_y |\alpha_{x_0 y}|^2$, and projects on the state $\frac{1}{\sqrt{\sum_y |\alpha_{x_0 y}|^2}} \sum_y \alpha_{x_0 y} |y\rangle$.

2.3 Quantum Algorithms

Our new attacks are based on well-known quantum algorithms: Deutsch's algorithm [26], which is a single-bit version of the Deutsch-Jozsa algorithm [27], the Bernstein-Vazirani algorithm [8], Simon's algorithm [60] and Shor's algorithm [59]. These algorithms have in common to be based on *Fourier sampling*, a process in which a quantum Fourier transform is applied before and after a single query to a superposition oracle. They are also amongst the earliest quantum algorithms proven to beat any classical algorithm, and as such are often presented in textbooks (see e.g. [55]). However, except for Shor's algorithm, their practical interest remained unclear for a long time.

Deutsch's Algorithm. Deutsch's algorithm [26] solves Problem 1 with probability 1 using a single query to O_f, whereas classically, two queries to f are needed for the same success probability. This constant speedup might seem anecdotal, but is crucial when the same function cannot be queried more than once.

Algorithm 1 Deutsch's algorithm

```
Start from $|0\rangle$                        ▷ $|0\rangle$
Apply a Hadamard gate                         ▷ $|0\rangle + |1\rangle$
Apply $O_{f,\pm}$    ▷ $(-1)^{f(0)} |0\rangle + (-1)^{f(1)} |1\rangle = (-1)^{f(0)} \left(|0\rangle + (-1)^{f(0)\oplus f(1)} |1\rangle\right)$
Apply a Hadamard gate                         ▷ $(-1)^{f(0)} |f(0) \oplus f(1)\rangle$
Measure the state
```

Algorithm 2 Bernstein-Vazirani algorithm

```
Start from $|0_n\rangle$                      ▷ $|0_n\rangle$
Apply a Hadamard transform                    ▷ $\sum_i |i\rangle$
Apply $O_{f,\pm}$                             ▷ $\sum_i (-1)^{(a\cdot i)\oplus b} |i\rangle$
Apply a Hadamard transform
Measure the state    ▷ $(-1)^b H^{\otimes n} \sum_i (-1)^{a\cdot i} |i\rangle = (-1)^b H^{\otimes n} \left(H^{\otimes n} |a\rangle\right) = (-1)^b |a\rangle$
```

Problem 1 (Deutsch's problem). Given access to a quantum oracle O_f for a function $f : \{0,1\} \to \{0,1\}$, decide whether f is *constant* ($f(0) = f(1)$) or *balanced* ($f(0) \neq f(1)$).

Deutsch's algorithm (Algorithm 1) is best presented with a phase oracle $O_{f,\pm} |b\rangle = (-1)^{f(b)} |b\rangle$. It can be seen that upon measurement, the algorithm actually yields the value $f(0) \oplus f(1)$ (although a single query has been made to f) whose knowledge solves Problem 1.

Bernstein-Vazirani Algorithm. The Bernstein-Vazirani algorithm [8] offers a polynomial speedup for finding the slope of an affine function over $\mathbb{F}_2^n$.

Problem 2 (Bernstein-Vazirani). Given access to an oracle O_f for an affine function $f : \{0,1\}^n \to \{0,1\}$, that is, $f(x) = a \cdot x \oplus b$ for a, b unknown, find a.

Upon measurement in Algorithm 2, we obtain the unknown a with certainty, using a single query to $O_{f,\pm}$, while n queries would be needed classically.

Remark 1. This algorithm can be seen as a generalization of Deutsch's algorithm. Indeed, in the case $n = 1$, there are only two types of affine functions: $f(x) = x \oplus b$ ($a = 1$) and $f(x) = b$ ($a = 0$), and Bernstein-Vazirani allows to distinguish them in one query.

Simon's Algorithm. Simon's algorithm [60] solves the problem of distinguishing an injective function from a periodic one. Note that it was the first example of an exponential quantum speedup relatively to an oracle.

Problem 3 (Simon). Given access to a function $f : \{0,1\}^n \to \{0,1\}^n$ for which there exists s such that: $\forall x, y, f(x) = f(y) \iff y \in \{x, x \oplus s\}$, find s.

In Algorithm 3, at Step 9 in the injective case, the value a obtained before has a single preimage x_a. Thus, the current state is $\sum_y ((-1)^{x_a \cdot y}) |y\rangle$ and we sample a uniformly random $y \in \{0,1\}^n$. After $n + r$ such samples, the family Y

Algorithm 3 Simon's algorithm

```
$Y = \emptyset$
Choose a number $r$ depending on the required probability of error
Repeat $n + r$ times
    Start from $|0_n 0_n\rangle$
    Apply a Hadamard transform to the first register    ▷ $\sum_x |x\rangle |0\rangle$
    Apply $O_f$ (standard)    ▷ $\sum_x |x\rangle |f(x)\rangle$
    Measure the second register, obtain $a$    ▷ $\sum_{x|f(x)=a} |x\rangle$
    Apply a Hadamard transform    ▷ $\sum_y (\sum_{x|f(x)=a} (-1)^{x \cdot y}) |y\rangle$
    Measure a $y$, $Y \leftarrow Y \cup \{y\}$
EndRepeat
if $Y$ is of full rank then
    return "injective case"
else if $Y$ is of rank $n - 1$ then
    return "periodic case" and the $s$ orthogonal to $Y$
else
    return "failure"
end if
```

will grow to a full-rank family. In the periodic case, the value a has exactly two preimages x_a and $x_a \oplus s$ which interfere with each other. The current state is

$$\sum_y ((-1)^{x_a \cdot y} + (-1)^{(x_a \oplus s) \cdot y}) |y\rangle = \sum_y (-1)^{x_a \cdot y} (1 + (-1)^{s \cdot y}) |y\rangle$$

and only the vectors y orthogonal to s have a non-zero amplitude. Thus, the family Y grows to span the euclidean subspace orthogonal to s. Computing the rank of Y allows to detect the period and solving the linear system $Ys = 0_n$ allows to recover it.

Generalizations. Although the original Simon's problem concerns functions without random collisions (that is, we cannot have $f(x) = f(y)$ if $x \oplus y \notin \{0, s\}$), it can be shown that the algorithm works as well for *random functions having a period*, which models the cryptographic problems that we are interested in.

The following simple condition was given in [39]. For Simon's algorithm to run as expected (i.e., with $\mathcal{O}(n)$ queries), it is sufficient for the periodic function f, of period s, to satisfy the following condition:

$$\max_{t \notin \{0,s\}} \Pr_x [f(x \oplus t) = f(x)] \leq \frac{1}{2} \ . \qquad (2)$$

That is, f should not admit another "unwanted partial period" t. In the examples studied in this paper, the condition (2) will be easy to check.

Note that if we had $f(x \oplus t) = f(x)$ for all x, then t would simply turn the set of periods of f into a vector space of dimension 2. In general, the space of periods could be a vector space of any dimension. An extended version of Simon's algorithm by Brassard and Høyer [20] allows to recover this whole space in polynomial time.

Finally, another important case is when the output set is smaller than the input set. This was studied in [16] for Simon's algorithm and [48] for period-finding in general. The results in [16] show that as long as the functions behave as random (but with the periodicity constraint), then for n input bits, the number of output bits required to run correctly without any cost increase is of order $\log_2 n$. The results in [48] show that the output can be hashed down to a single bit, and the algorithms still work up to a constant increase in queries.

Shor's Algorithm. We will use Shor's algorithm [59] to solve the *abelian hidden period* problem. It will appear in a "black-box" manner in Section 5.4, and in Section 5.5. We will analyse in detail the behavior of the algorithm on Poly1305.

Problem 4 (Abelian hidden period). Let $(G, +)$ be an abelian group, X a set. Given access to a function $f \ : G \to X$ which is either injective, or periodic $(\exists s \in G, f(s + \cdot) = f(\cdot))$, then determine the case and / or find the period.

In particular, we consider $G = \mathbb{Z}_{M_1} \times \ldots \times \mathbb{Z}_{M_k}$ the product of multiple cyclic groups of known order. For simplicity, and to prepare for Section 5.5, we present the algorithm in the case of $\mathbb{Z}_p^2$ for some prime p. Note that f is also periodic over $\mathbb{Z}_p$ in each of its parameters. This is the typical situation when Shor's algorithm is used to solve the Discrete Logarithm Problem. The periods of f form a two-dimensional integer lattice, which is generated by $(p, 0)$ and $(-1, s)$ for some s. In other words, the value of $f(x, y)$ depends only on the value of $xs + y \bmod p$. We may assume for simplicity that the function $xs + y \bmod p \mapsto f(x, y)$ is injective.

The algorithm only relies on an efficient implementation of the Quantum Fourier Transform over $\mathbb{Z}_p$:

$$|x\rangle \mapsto \sum_{y=0}^{p-1} \exp\left(2i\pi \frac{xy}{p}\right) |y\rangle \quad ,$$

which we assume exact. We represent the elements of X on m bits.

In Algorithm 4, at Step 4, we can only measure a vector $|z, t\rangle$ having a nonzero amplitude. This means that we need:

$$\sum_{x=0}^{p-1} \exp\left(2i\pi \frac{(z - st)x}{p}\right) \neq 0 \quad ,$$

which happens only when $(z - st) = 0$. In that case, the sum simply gives p. After renormalization, all vectors $|z, t\rangle$ with $(z - st) = 0$ have the same amplitude $\frac{1}{\sqrt{p}}$, and we will measure one of them taken uniformly at random. If $t \neq 0$, we compute s by $s = zt^{-1} \bmod p$. This occurs with probability $1 - \frac{1}{p}$.

2.4 Attack Scenarios

We consider different *attack scenarios* throughout this paper.

Algorithm 4 Shor's algorithm

```
Start from $|0, 0, 0_m\rangle$    ▷ $|0, 0, 0_m\rangle$
Apply a Quantum Fourier Transform on both input registers
    ▷ $\sum_{x,y=0}^{p-1} |x, y\rangle |0_m\rangle$
Apply $O_f$    ▷ $\sum_{x,y=0}^{p-1} |x, y\rangle |f(x, y)\rangle$
Measure the second register. The state collapses on a uniform superposition of all $(x, y)$ such that $xs + y = a \bmod p$ for some unknown $a$, meaning $y = a - xs \bmod p$:
    $\sum_{x=0}^{p-1} |x\rangle |a - xs\rangle$ .
Apply a Quantum Fourier Transform again. The state becomes:
    $\sum_{z,t=0}^{p-1} \left( \sum_{x=0}^{p-1} \exp\left(2i\pi \frac{zx + (a - xs)t}{p}\right)\right) |z, t\rangle$
    $= \sum_{z,t=0}^{p-1} \exp(2i\pi at/p) \left( \sum_{x=0}^{p-1} \exp\left(2i\pi \frac{(z - st)x}{p}\right)\right) |z, t\rangle$ .
Measure a $|z, t\rangle$ and return $s = zt^{-1} \bmod p$.
```

Q1 and Q2 setting. Following [40,33,17], we will adopt the Q1 / Q2 terminology to classify quantum attacks on symmetric schemes. Note that these models have alternative names, for example "quantum chosen-plaintext attack" (qCPA) is used for "Q2" in [34,22]. In the Q1 setting, the adversary is given only *classical* encryption or decryption query access to black-boxes. In the Q2 setting, the adversary is given *quantum* or *superposition* access, in the sense that a black-box E_K becomes a quantum oracle O_{E_K}. This is the case for all the attacks of this paper.

The study of quantum attacks on symmetric schemes in the Q2 setting was sparkled by seminal work of Kuwakado and Morii [44,45], who showed that the 3-round Luby-Rackoff construction and the Even-Mansour cipher became insecure if exposed to superposition queries. More precisely, they can use Simon's algorithm to respectively distinguish the construction and recover the key of the cipher in polynomial time, while classical proofs of security exist.

Attacks based on period-finding. Since these earlier results, many works have extended the reach of Q2 attacks [39,15,46,19,28,30]. However, the attack strategy has remained the same. A hidden structure is embedded in the construction to be attacked, so that $f(E_K(x), x)$ for some choice of combination f, is a periodic function of x; or that a shift exists between $f(E_K(x), x)$ and $g(E_K(x), x)$. The recovery of this hidden period or shift, which is secret material, then leads to a break. We can cite some examples:

Against the Even-Mansour construction [45]: $E_K(x) = K_2 \oplus \Pi(x \oplus K_1)$ for a random public permutation Π and two keys K_1, K_2. One has:

$$E_K(x) \oplus \Pi(x) = E_K(x \oplus K_1) \oplus \Pi(x \oplus K_1)$$

which leads to a recovery of K_1 in $\mathcal{O}(n)$ queries and $\mathcal{O}\left(n^3\right)$ computations.

Against $\mathsf{CBC\text{-}MAC}$ *with two blocks [39]:* It can be defined as:

$$\mathsf{CBC\text{-}MAC}(y, x) = E_{K'} \circ E_K\Big(x \oplus E_K(y)\Big) \ ,$$

where K and K' are two keys that will remain unknown to the adversary. Due to the structure of $\mathsf{CBC\text{-}MAC}$, one can take two arbitrary values α_0, α_1, and define the function:

$$F \ : \begin{cases} \{0,1\} \times \{0,1\}^n \mapsto & \{0,1\}^n \\ \qquad (b, x) & \to \mathsf{CBC\text{-}MAC}(\alpha_b, x) \end{cases} \qquad (3)$$

We have then that $F(b, x) = F(b \oplus 1, x \oplus E_K(\alpha_0) \oplus E_K(\alpha_1))$. Thus F has a hidden boolean period $1 \| E_K(\alpha_0) \oplus E_K(\alpha_1)$. Having obtained the internal value $E_K(\alpha_0) \oplus E_K(\alpha_1)$, we can query the tag of any message starting with block α_0, and then forge a message starting with α_1 with the same tag.

Constructions based on IVs. We consider two types of constructions with quantum access: some make use of an *initialization value* (IV, sometimes also named a *nonce*) and some do not. In the IV case, we consider that the IV is a classical value, chosen randomly before each oracle query, and not repeated. This model follows from the idea that the IV is not controlled by the adversary, and it can serve as an intermediate between the classical setting and a (much) stronger model in which the adversary would completely (and quantumly) control the IVs.

In fact, the latter case does not seem to have been studied so far in quantum security. Well-known notions such as IND-qCPA [14] rely on classical randomness, and many modes of operation have been proven secure in this model [3,9].

In the classical setting, many MAC constructions have a security that relies on the non-repetition of IVs, for example the MAC of OCB [43]. The same happens in the quantum setting, since the MAC of QCB [9] has been proven secure under quantum queries with classical non-repeated IVs.

Unforgeability. The first notion of quantum unforgeability for MACs was defined by Boneh and Zhandry [13]. We will name it *plus-one unforgeability* (PO), following [1]. The idea is that an adversary making q quantum queries to the construction, where q is polynomial, should not be able to produce $q + 1$ valid $\{\text{message}, \text{tag}\}$ pairs. A more recent definition is *blind-unforgeability* (BU), proposed in [1]. It is strictly stronger than PO-unforgeability. In this paper, we will give several quantum forgery attacks that break the PO notion, thereby also breaking BU.

Quantum PRFs. A *quantum pseudorandom function* (qPRF) is a family of functions F_K, indexed by a key space $\mathcal{K}$, such that no quantum adversary making queries to an oracle O_f can distinguish efficiently between a function F_K, with

K drawn uniformly at random, and a truly random function. It is shown respectively in [13] and [1] that a qPRF is also a quantum-secure deterministic MAC by the PO and BU definitions. Therefore, any function that is not PO-unforgeable is also not a secure qPRF. To the best of our knowledge, the only classical symmetric construction that has been proven quantum-secure as a deterministic MAC, the Cascade / NMAC / HMAC construction [61], is also a qPRF.

2.5 A Quantum Attack on OCB3

We detail the Q2 attack on the MAC of OCB3 from [39]. As the other previous works recalled above, this attack relies on a Boolean period-finding problem.

Specification. OCB3 is an IV-based mode of authenticated encryption with associated data (AEAD), based on a block cipher [43]. As OCB stands for *offset codebook*, the scheme relies on the definition of *offsets* that are dependent on the key and change between each block. We will focus on the authentication tag of OCB3 (see Figure 1). Our considerations are independent on the exact value of the offsets, and apply to all versions of OCB, but we use OCB3 as a concrete example.

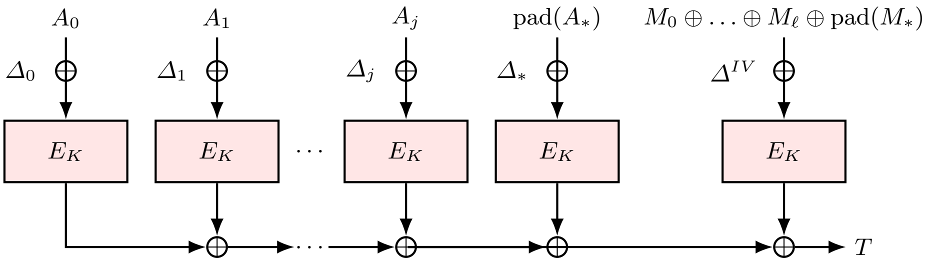

Fig. 1. Computation of the tag in OCB3. Only the offset Δ^{IV} depends on the IV.

Forgery Attack with Simon's Algorithm. Kaplan *et al.* showed in [39] how to forge authentication tags using Simon's algorithm. The idea is to query the tag of an empty message with two AD blocks $A_0, A_1 = x$:

$$x \rightarrow E_K(\Delta_{IV}) \oplus E_K(\Delta_0 \oplus x) \oplus E_K(\Delta_1 \oplus x) \ .$$

One can then remark that this function of x is periodic, of period $\Delta_0 \oplus \Delta_1$, independent of the IV, and only on the secret key K. Although the function changes at each query (since the IV changes), the period is always the same and Simon's algorithm allows to recover it with $\mathcal{O}(n)$ superposition queries. (For the same reason, we could use a non-empty message, and even different messages between the queries.)

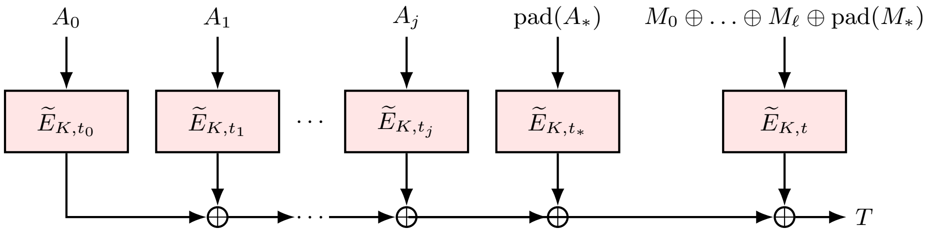

Fig. 2. Computation of the tag in ΘCB3. Only the final tweak t depends on the IV.

Once $\Delta_0 \oplus \Delta_1$ has been obtained, one can then query the tag of any pair of AD blocks A_0, A_1 and forge the tag of A_1, A_0.

Remark 2. It is easy to check that Equation (2) is satisfied in practice. If it wasn't, then the existence of an unwanted partial period t:

$$\Pr_x\left[f(x \oplus t) = f(x)\right] \geq \frac{1}{2} \ ,$$

would imply a higher-order differential of probability greater than $\frac{1}{2}$ for E_K, which is impossible if E_K is a pseudorandom permutation (in other words, E_K would suffer from a classical break).

3 The Quantum Linearization Attack: Algorithmic Ideas

In this section, we present the algorithmic ideas underlying our new *quantum linearization attack*. To that end, we keep the example of OCB3 [43] introduced in Section 2.5. We explain a new way to forge with Q2 queries. The extensions and applications of this new idea will be explored in the next sections.

Note that to the best of our knowledge, this is the first application of the Deutsch and Bernstein-Vazirani algorithms for forgery attacks.

3.1 Attack on ΘCB with Deutsch's Algorithm

The attack of Section 2.5 works only because of the offsets. In fact, the existence of a controlled value (here x) XORed to a secret (here the offsets) has been so far a prerequisite of all Q2 attacks.

Here we present a forgery attack against the mode ΘCB3 [57,43], which is a more abstract version of OCB3 in which the block cipher E_K is replaced by a *tweakable* block cipher (a family of independent block ciphers $\widetilde{E}_{K,t}$ indexed by a *tweak* t). This is shown in Figure 2.

Here, the tweaks $t_0, \ldots, t_j, t_*$ form an arbitrary sequence of distinct values, that depend only on the block index; the tweak t is the only one dependent on the IV. Again, we consider an empty message, but this time a single AD block that is either 0 or 1. We define i functions which truncate the output of such a call to the ith bit:

$$F_i : \begin{array}{rcl} \{0,1\} & \to & \{0,1\} \\ b & \mapsto & \mathrm{Trunc}_i(\widetilde{E}_{K,t_0}(b) \oplus \widetilde{E}_{K,t}(0)) \end{array} \quad .$$

The functions F_i change at each new superposition query (because the IV intervenes in $\widetilde{E}_{K,t}(0)$). Thus we need the ability to compute a query to F_i using a *single* query to the untruncated mode itself. This is fortunately easy to do so using the truncation technique of [33].

With this single query, Deutsch's algorithm allows to recover the value:

$$\begin{aligned} &\mathrm{Trunc}_i(\widetilde{E}_{K,t_0}(0) \oplus \widetilde{E}_{K,t}(0)) \oplus \mathrm{Trunc}_i(\widetilde{E}_{K,t_0}(1) \oplus \widetilde{E}_{K,t}(0)) \\ &\qquad\qquad\qquad\qquad = \mathrm{Trunc}_i(\widetilde{E}_{K,t_0}(0) \oplus \widetilde{E}_{K,t_0}(1)) \ , \end{aligned}$$

and within n queries and uses of the algorithm, we can obtain the full value $\widetilde{E}_{K,t_0}(0) \oplus \widetilde{E}_{K,t_0}(1)$.

We can now forge valid messages as follows: we query a message with 0 as the first block, we XOR $\widetilde{E}_{K,t_0}(0) \oplus \widetilde{E}_{K,t_0}(1)$ to the tag, and we have obtained the tag of the same message with 1 replacing the first block. This works for any block and for any pair of messages.

This attack shows that XORing with an IV-dependent value, although it provides sufficient protection against forgeries in the classical setting (since ΘCB has a security proof), does not in the quantum setting.

Interestingly, it is possible to protect against this attack by using the IV in the TBC calls, as done by Bhaumik et al. in [9]. While this simple modification has practically no incidence on the classical security of the mode, it is crucial to obtain unforgeability in the quantum setting.

Another Example: XOR-MACs. In [4], two *XOR-MAC* constructions are defined, which can be attacked with Deutsch's algorithm. They are both based on a pseudorandom function F_K and an IV. The first one, XMACR (*randomized XOR scheme*), considers that the IV is drawn uniformly at random, and the second one, XMACC (*counter-based XOR scheme*) that it is maintained as a counter. Both compute:

$$\mathsf{MAC}(m_1, \dots, m_\ell; IV) = F_K(0 \| IV) \oplus \bigoplus_{1 \le i \le \ell} F_K(1 \| i \| m_i) \ .$$

Then, since the contribution of the IV is only XORed, forgeries can be made.

3.2 Using the Bernstein-Vazirani algorithm

We propose here a generalization of the previous attack, with longer queries. We now consider functions of the form

$$g_1(x_1) \oplus g_2(x_2) \oplus \cdots \oplus g_\ell(x_\ell) \oplus C$$

with, as before, a C that is independent from all x_i. Now, we can choose some arbitrary α_i^0 and α_i^1, and consider the function

$$F_j : \begin{array}{rcl} \{0,1\}^\ell & \to & \{0,1\} \\ (b_1, \ldots, b_\ell) & \mapsto & \mathrm{Trunc}_j \left(\bigoplus_{i=1}^{\ell} g_i(\alpha_i^{b_i}) \oplus C \right) \end{array} ,$$

It is easy to see that this function is affine: indeed, if we change the value of b_i, then we add $\mathrm{Trunc}_j \left(g_i(\alpha_i^0) \oplus g_i(\alpha_i^1) \right)$ to the output.

Hence, if we apply the Berstein-Vazirani algorithm, in one query, we recover the values of the $\mathrm{Trunc}_j \left(g_i(\alpha_i^0) \oplus g_i(\alpha_i^1) \right)$, for all i. Next, it suffices to repeat the algorithm for each bit of the output to obtain the value of all the $g_i(\alpha_i^0) \oplus g_i(\alpha_i^1)$.

This technique can be applied to OCB3 / ΘCB3, as the tag is a function of the form

$$\bigoplus_i g_i^k(AD_i) \oplus f_k(IV, M)$$

Hence, we can attack multiple blocks of associated data at once.

3.3 Attacking any XOR of permutations

The main limitation of the previous attacks is that they need a direct access to the linear combination of independent blocks. In this section, we overcome this limitation with an attack that leverages linear combinations of permutations in a more intrinsic way, using Simon's algorithm in a novel fashion.

We consider a MAC construction that processes $m > n$ message blocks $x_1, \ldots, x_m$ by pushing the x_i through independent TBC calls $\widetilde{E}_{K,i}$, XORing the result and applying an IV-dependent function afterwards.

$$IV, (x_1, \ldots, x_m) \mapsto f_K \left(IV, \left(\bigoplus_{1 \le i \le m} \widetilde{E}_{K,i}(x_i) \right) \right) .$$

Remark 3. We write the attack with a TBC, i.e., a family of independent block ciphers $\widetilde{E}_{K,T}$ indexed by a secret key K *and* a public tweak T. This is to emphasize the application of our attack to parallelizable MACs; however, the attack works in the same way if we replace the independent block ciphers by independent functions.

In the case of ΘCB, the definition of f_K is simple, since it only XORs the IV- and the AD-dependent parts. But the attack of Section 3.2 does not apply anymore if f_K is a pseudorandom function. This will be the case of our new attack, which is why it will apply to many MAC constructions.

Quantum Attack. First of all, it is easy to see that if the $\widetilde{E}_{K,i}$ are independent block ciphers, and if f_K is a pseudorandom function family, then this construction is a classically unforgeable MAC: this is the security of ΘCB3.

Our attack in the quantum setting starts from the same idea as above (Section 3.2): we query the MAC with arbitrary blocks taking two values: $x_1 =$

$b_1||0_{n-1}$, …, $x_m = b_m||0_{n-1}$, where $x = b_1 \dots b_m$ forms an m-bit input (in the remaining of this paper, we will write the $n-1$ zeroes used for completion as a single 0). We will put x in superposition, and so, there will be only "one superposed bit" in each individual block input.

One then observes that $\widetilde{E}_{K,1}(x_1) \oplus \dots \oplus \widetilde{E}_{K,m}(x_m)$ is an affine function of x:

$$\begin{aligned} F(x) &:= \widetilde{E}_{K,1}(x_1) \oplus \dots \oplus \widetilde{E}_{K,m}(x_m) \\ &= \bigoplus_i \left(b_i \odot \left(\widetilde{E}_{K,i}(0) \oplus \widetilde{E}_{K,i}(1) \right) \oplus \widetilde{E}_{K,i}(0) \right) \ . \end{aligned}$$

More precisely, if we identify bit-strings with boolean column vectors, $F(b_1 \dots b_m)$ is equal to:

$$\underbrace{\left((\widetilde{E}_{K,1}(0) \oplus \widetilde{E}_{K,1}(1)) \ \cdots \ (\widetilde{E}_{K,m}(0) \oplus \widetilde{E}_{K,m}(1)) \right)}_{M_m} \times \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix} \oplus \bigoplus_i \widetilde{E}_{K,i}(0) \ .$$

The matrix M_m has n rows and m columns, so when $m \geq n+1$, its kernel is nontrivial. This means there will exist a non-zero m-bit boolean vector α such that:

$$\left((\widetilde{E}_{K,1}(0) \oplus \widetilde{E}_{K,1}(1)) \ \cdots \ (\widetilde{E}_{K,m}(0) \oplus \widetilde{E}_{K,m}(1)) \right) \times \alpha = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \ ,$$

and for all such vectors α, seen as m-bit strings, we have:

$$F(x \oplus \alpha) = M_m \times (x \oplus \alpha) \oplus \bigoplus_i \widetilde{E}_{K,i}(0) = F(x) \ .$$

In other words, this function F hides a subgroup of $(\mathbb{F}_2)^m$ generated by all the vectors α satisfying the condition above (it is easy to see that they indeed form a group). Thus, F satisfies the promise of Simon's algorithm: by making a single superposition query, we can find y such that $y \cdot \alpha = 0$ for such an α, and furthermore, as Brassard and Høyer showed [20], we can even recover the full subspace of periods with a polynomial number of quantum queries to F.

However, in our model, we cannot query F directly and we have instead access to: $f_K(IV, F(x))$, where IV changes at each query. The key remark is that the hidden subgroup is unchanged, since F is independent of the IV. This assumption is enough to allow Simon's algorithm and its extensions to work.

Remark 4 (Smaller m). Some period might still arise if $m \leq n$. Indeed, if $m = n$, there will be a non-trivial period with probability around $1 - 1/e$. This quickly decays for smaller m.

Remark 5 (Unwanted collisions). Since the "inner" function F is affine, it does not contain any unwanted collisions. If $F(x \oplus \alpha) = F(x)$ for some α and x, then this holds as well for all x. However, unwanted collisions might occur in $f_K(IV, \cdot)$.

Assuming that M_m is full rank, we can express the probability of unwanted partial periods for $f_K(IV, F(\cdot))$ as the probability of such unwanted collisions for $f_K(IV, \cdot)$:

$$\begin{aligned} p &= \max_{t, M_m \times t \neq 0} \Pr_{x \in \{0,1\}^\ell} \left[f_K(IV, F(x \oplus t)) = f_K(IV, F(x)) \right] \\ &= \max_{t, M_m \times t \neq 0} \Pr_{x \in \{0,1\}^\ell} \left[f_K(IV, F(x) \oplus M_m \times t) = f_K(IV, F(x)) \right] \\ &= \max_{u \neq 0} \Pr_{x \in \{0,1\}^\ell} \left[f_K(IV, F(x) \oplus u) = f_K(IV, F(x)) \right] \\ &= \max_{u \neq 0} \Pr_{y \in \{0,1\}^n} \left[f_K(IV, y \oplus u) = f_K(IV, y) \right] \ . \end{aligned}$$

Even if the output is truncated to less than n bits, $p \leq \frac{1}{2}$ follows trivially from the fact that $f_K(IV, \cdot)$ should not admit a differential of such high probability. To conclude, it is precisely the fact that the termination function $f_K(IV, \cdot)$ is a good PRF, and does *not* admit an interfering period, that allows to apply easily Simon's algorithm in our case.

Thus, by making a polynomial number of Q2 queries to the MAC construction, we can obtain such an α. This allows to create forgeries as follows.

Forgeries without IVs. We first make n queries to find a valid α with Simon's algorithm (with constant probability of success). Then, the knowledge of this α allows us, for each tag x queried, to output a forged tag $x \oplus \alpha$. Thus we can double the number of tags that we produce compared to the number of queries we make. This breaks the PO notion as soon as, making $r + n$ queries, we have $2r \geq r + 1 + n$ tags, thus with $2n + 2$ queries in total. Note that by breaking PO, we are actually showing that the MAC construction is not a qPRF (if it were, it would be PO-secure).

Forgeries with IVs. As long as the IV (or nonce) is used only in the keyed post-processing, we can recover a value α and run the attack as above. We will indeed output more triples $\{IV, \text{message}, \text{tag}\}$ than the number of queries made, although some IVs are repeated in the outputs.

Universal Forgeries. Instead of taking the arbitrary values $b_i||0$ in message blocks, we can take any pair of values for each of them. That way, we can even start from any m-block message $y_1, \ldots, y_m$, and then define a function of $x = b_1 \ldots b_m$ that inputs y_i in block i if $b_i = 1$ and an arbitrary value 0 otherwise. Using Simon's algorithm, we will find a subset of the y_i such that the $\widetilde{E}_{K,i}(y_i)$ have the same XOR as the $\widetilde{E}_{K,i}(0)$. Hence, we can produce a new message having the same tag as $y_1, \ldots, y_m$. This works as soon as $m \geq n$ (there just needs to be enough message blocks for our attack).

4 Applications to Parallelizable MACs

In this section, we apply the quantum linearization attack to many parallelizable MACs of the literature. In particular, we show that the attack can be extended to parallelizable beyond birthday-bound (BBB) MACs, although they have a larger internal state. Here is a summary of MACs attacked in this section (usually in time quadratic in the internal state size n), whose previous best quantum attack was exponential:

LightMAC [47], LightMAC+ [54], Deoxys [38], ZMAC [37],
PMAC_TBC3k [53]

On the contrary, here are some MACs on which, to the best of our knowledge, our attack does not apply: SUM-ECBC [62], 2K-ECBC-Plus [24], 3kf9 [63]. The best Q2 attacks on these remain exponential-time (usually $\widetilde{\mathcal{O}}\left(2^{n/2}\right)$ or $\mathcal{O}\left(2^{k/2}\right)$ where n is the internal block size, and k the key size).

4.1 First Examples

We will consider MAC designs with or without IVs or nonces. When there is no IV, then the attack of Section 3.3 breaks them in the PO notion. This also shows that even though they usually yield classical PRFs, these constructions are not quantum-secure PRFs. When there is an IV, the MAC may be insecure as a PRF but still secure as a MAC (since the IV is changed at each query, and not repeated). Despite that, our attack may still yield a break, as we showed in the example of ΘCB above. In that case, the period that is recovered with Simon's algorithm is independent of the IV, and can be reused to forge a new valid (message, tag) pair under any previously queried IV.

LightMAC. LightMAC [47] is based on an n-bit block cipher and separates the message in blocks of $n-s$ bits, where $s \leq n/2$ is some parameter that limits the maximal message size. The function is the following:

$$\mathsf{LightMAC}(m_1, \ldots, m_\ell) = \mathrm{Trunc}_t\left(E_{K_2}\left((m_\ell 10*) \oplus \bigoplus_{i=1}^{\ell-1} E_{K_1}(i_s m_i)\right)\right) \quad ,$$

where the i_s are s-bit constants. Calling LightMAC with single-bit blocks and using Simon's algorithm, we immediately obtain a sequence of indices $j_1, \ldots, j_v$ such that $E_{K_1}(i_{j_1}1) \oplus \ldots \oplus E_{K_1}(i_{j_v}1) = E_{K_1}(i_{j_1}0) \oplus \ldots \oplus E_{K_1}(i_{j_v}0)$ and thus, we can produce existential forgeries, and universal forgeries of messages with a linear number of blocks.

Deoxys. Due to the similarity of its MAC with ΘCB, our attack applies to all versions of Deoxys-II [38], one of the finalists of the CAESAR competition (it also applies to Deoxys-I).

Protected Counter Sums. In [5], Bernstein defined the *protected counter sum* construction, which uses a pseudorandom function $f \ : \{0,1\}^{d+c} \rightarrow \{0,1\}^c$ to build a pseudorandom function with message space of at most $2^c - 1$ blocks of length d:

$$f'(m_1, \ldots, m_\ell) = f\left(0 \| f(1\|m_1) \oplus \ldots \oplus f(\ell\|m_\ell)\right) \quad .$$

The quantum linearization attack essentially shows that this construction, while classically sound, does not yield a quantum-secure pseudorandom function (even if f itself is a qPRF).

4.2 Attacks on BBB MACs

We consider here a variant of the previous construction typically used to design Beyond Birthday MACs. We focus on deterministic MACs, but as before, the same forgery attacks apply if IVs are used in the final processing of the tag.

In the most generic setting, the input $x_1, \ldots, x_m$ is processed with a TBC $\widetilde{E}_{K,i}$, then combined in two different ways:

$$(x_1, \ldots, x_m) \mapsto f_K\left(\bigoplus_i \widetilde{E}_{K,i}(x_i), \bigoplus_i 2^i \widetilde{E}_{K,i}(x_i) \right) \enspace .$$

Here f_K is a function whose details are insignificant for our attack.

A similar observation as above applies. By calling the MAC in superposition with messages of the form $x = b_1||0, \ldots, b_m||0$, we will obtain a periodic function. Indeed, there are now two matrices M_m and M'_m with n rows and m columns, and two column vectors C, C' such that:

$$F(x) = F(b_1, \ldots, b_m) := f_K\left(M_m \times \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix} \oplus C, M'_m \times \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix} \oplus C' \right) \enspace ,$$

where the columns of M_m correspond to $\widetilde{E}_{K,i}(0) \oplus \widetilde{E}_{K,i}(1)$ and the columns of M'_m correspond to $2^i(\widetilde{E}_{K,i}(0) \oplus \widetilde{E}_{K,i}(1))$. Then, as soon as $m \geq 2n+1$, the matrix: $\begin{pmatrix} M_m \\ M'_m \end{pmatrix}$ has $2n$ rows and at least $2n+1$ columns, and so, it has a non-trivial kernel. There exists a non-zero vector α such that

$$M_m \alpha = M'_m \alpha = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \enspace .$$

This α is a boolean period of F, for which $MAC(x \oplus \alpha) = MAC(x)$. Again, the further we increase m, the bigger the subspace of periods will become. This whole space can be recovered using Brassard and Høyer's extension of Simon's algorithm [20] in polynomial time.

Related works. In [30], Guo, Wang, Hu and Ye used combinations of Simon's algorithm and Grover's algorithm to design forgery attacks on many BBB MACs, in the Q2 setting. With this technique, they found two things. First, state-recovery attacks of complexity $\widetilde{\mathcal{O}}\left(2^{n/2}\right)$ where n is the block size of the underlying block cipher, and the internal state is $2n$ bits in total. This comes from the fact that the same input blocks are processed in two branches separately. The standard use of Simon's algorithm, where a controlled message block x is XORed to an uncontrolled value, allows only to recover this value in one of the branches. The n bits on the other branch have to be guessed with a Grover search, and so, the attack is a Grover-meets-Simon [46] instance. And next, partial key-recovery attacks for parallelizable MACs, of complexity $\mathcal{O}\left(2^{k/2}\right)$, where k is the partial

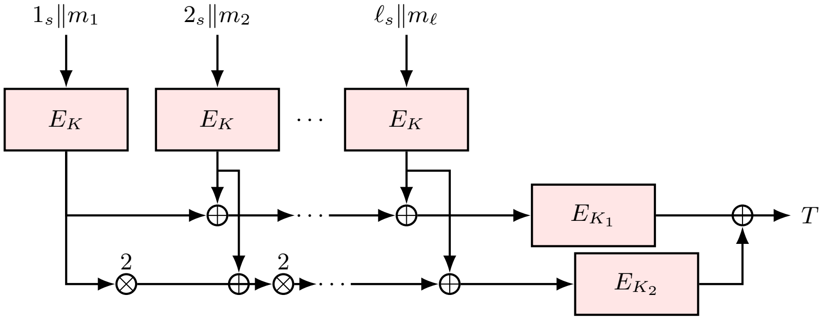

Fig. 3. LightMAC+ with three keys K, K_1, K_2.

key size (the total key size ranges from $3k$ to $5k$). They consist in guessing part of the key and breaking the MAC by using a symmetry of the branches. To these attacks correspond classical partial key-recoveries of complexity $\mathcal{O}\left(2^k\right)$.

Our attack has completely different requirements and offers different results. We need longer messages (of roughly $2n$ blocks in this setting), but when it applies, the complexity is always polynomial. Note that there are constructions targeted by Guo *et al.*, such as SUM-ECBC, that we cannot attack since the blocks are processed sequentially and not linearly in parallel as we require.

LightMAC+. LightMAC+ [54], as its name suggests, is a BBB extension of LightMAC.

As shown in Figure 3, it processes ℓ message blocks $m_1, \ldots, m_\ell$ as follows:

$$\begin{aligned}\mathsf{LightMAC+}(m_1, \ldots, m_\ell) = &\, E_{K_1}(E_K(1_s\|m_1) \oplus \ldots \oplus E_K(\ell_s\|m_\ell)) \\ &\oplus E_{K_2}(2^{\ell-1} \odot E_K(1_s\|m_1) \oplus \ldots \oplus 2^0 \odot E_K(\ell_s\|m_\ell) \ ,\end{aligned}$$

where the multiplications are done in the finite field $\mathbb{F}_{2^n}$. This falls into our framework and is thus forgeable in quadratic time (about $2n$ blocks are required to embed a vector space in both branches, and this can then be recovered in a linear number of queries).

PMAC+. It is a double-block hash-then-sum construction similar to LightMAC+, which also falls into our framework. In full generality, there are three keys K_1, K_2, K_3. The message blocks $m_1, \ldots, m_\ell$ are processed as follows: $y_i = m_i \oplus 2^i \odot E_{K_1}(0) \oplus 2^{2i} \odot E_{K_1}(1)$ and then:

$$\begin{aligned}\mathsf{PMAC+}(m_1, \ldots, m_\ell) = &\, E_{K_2}\left(E_{K_1}(y_1) \oplus \ldots E_{K_1}(y_\ell)\right) \\ &\oplus E_{K_3}\left(2 \odot E_{K_1}(y_1) \oplus \ldots 2^\ell \odot E_{K_1}(y_\ell)\right) \ .\end{aligned}$$

The masking by $2^i \odot E_{K_1}(0) \oplus 2^{2i} \odot E_{K_1}(1)$ simply makes the processing of each block different, but this is insignificant for our attack. By recovering a period, we can create forgeries and break PMAC+ as a qPRF.

Note that both LightMAC+ and PMAC+ were classically proven secure up to $2^{3n/4}$ queries [41]. Besides, increasing the number of parallel branches may have consequences on the bound, but only increases the complexity of our attack by a constant factor. We considered here three-key versions, but of course, the two- and one-key versions [24,25] are similarly broken.

4.3 Other MACs

LAPMAC. LAPMAC was defined in [51]. The definition depends on some parallelization parameter μ. Successive chunks of μ message blocks will be processed in parallel through the block cipher E_K (except the last one), then some tweak function depending on their index in the chunk. The results are XORed and encrypted again, before being XORed to the next chunk of μ message blocks, etc. When $\mu \geq n$, LAPMAC applied to n message blocks becomes similar to LightMAC or PMAC, and there is sufficient parallelization to perform our attack. Whether a variant of the attack applies for smaller values of μ is an interesting question.

ZMAC. ZMAC [37] is a MAC that uses a TBC. It is based on the ZHASH double-block hash construction followed by a finalization function. We can simply focus on the abstraction $\mathbb{ZHASH}$ (see Fig. 5 in [37]):

$$\mathbb{ZHASH}(X_\ell^1, X_r^1, \dots, X_\ell^l, X_r^l) = \bigoplus_i 2^{l+1-i}\widetilde{E}_K^{i,X_r^i}(X_\ell^i), \bigoplus_i X_r^i \oplus \bigoplus_i \widetilde{E}_K^{i,X_r^i}(X_\ell^i) \ ,$$

where $\widetilde{E}_K^t$ is $\widetilde{E}_K$ called with a tweak t. If we make the tweak inputs constant, then the construction is similar to PMAC+ with a TBC, and different random keyed permutations for each block. Our attack applies as well.

PMAC with a TBC. Naito [53] showed that PMAC+ used with a TBC could achieve full PRF security (up to $\mathcal{O}(2^n)$ queries). In this variant, the message blocks are processed independently with different tweaks. This has no consequence on our attack, which requires only $\mathcal{O}(n)$ queries of about $2n$ blocks each.

5 Attacks on MACs based on Universal Hashing

In this section, we focus on some attacks on MACs based on universal hashing. In particular, we give polynomial-time attacks on PolyMAC [36], GCM-SIV2, and we give a superposition attack on Poly1305 requiring about 32 queries.

5.1 Overview

Universal hash functions were introduced by Carter and Wegman in 1977 [21] in order to build secure MACs, and are now used in many MAC constructions and security proofs. The first proposal by Wegman and Carter was to hash the message and to encrypt the result with a one-time-pad. This defines a MAC with information-theoretic security, but the use of a one-time-pad is impractical, and it was soon suggested to replace it with the output of a PRF, i.e., to replace the one-time-pad by counter-mode encryption. This results in the Wegman-Carter construction used in GCM and Poly1305-AES: $M \mapsto H_{K_1}(M) \oplus F_{K_2}(N)$ where F is a secure pseudorandom function family, and H an almost-XOR-universal hash function family.

5.2 Universal hash functions and MAC constructions

An almost-XOR-universal hash function family is a family of function H from $\{0,1\}^*$ to $\{0,1\}^n$ indexed by a key $K \in \mathcal{K}$ such that:

$$\forall m \neq m',\, \forall d \in \{0,1\}^n,\, \#\{K \in \mathcal{K} : H_K(m) \oplus H_K(m') = d\} \leq \varepsilon \# \mathcal{K}$$

The most widely used universal hash function construction is polynomial hashing. The input message is interpreted as the coefficients of a polynomial in a field $\mathbb{F}$, and the polynomial is evaluated on the hash key:

$$\mathsf{PolyHash}_K : \mathbb{F}^\ell \to \mathbb{F} \qquad\qquad m_1, m_2, \dots m_\ell \mapsto \sum_{i=1}^{\ell} K^i \odot m_i$$

Block cipher-based constructions such as the OCB3 MAC can also be analysed as universal hashing-based, using $\bigoplus_i E_K(A_i \oplus \Delta_i)$ as a universal hash function.

There are many different ways to turn a universal hash function into a MAC:

One-time-MAC: $\boldsymbol{H_K(M)}$. If the universal hash function satisfies extra properties (it must be strongly universal), it can be used directly as a MAC, if a new key is used for each message. This construction is used in ChaCha20-Poly1305, Grain128A and Grain128AEAD [32].

Wegman-Carter: $\boldsymbol{H_{K_1}(M) \oplus F_{K_2}(N)}$. The Wegman-Carter construction is a nonce-based MAC using a universal hash function H and a PRF F. It authenticates several messages using the same key, as long as the nonce N is not repeated (the security is lost as soon as two different messages are authenticated with the same key). This construction is used in GMAC.
More generally, the construction $H(M) \star F(N)$ with $\star$ a group operation and F almost-$\star$-universal is a secure MAC. This construction is used in Poly1305-AES.

Hash-then-PRF: $\boldsymbol{F_{K_2}(H_{K_1}(M))}$. The hash-then-PRF construction builds a deterministic MAC from a universal hash function H and a PRF F.
The PolyMAC construction discussed below follows this design. More generally, security proofs for several block cipher-based MACs consider the MAC as following the hash-then-PRF construction; in particular this is the case of double-block hash-then-sum constructions [24].

WMAC: $\boldsymbol{F_{K_2}(H_{K_1}(M) \| N)}$. WMAC [10] is a generalization of the hash-then-PRF construction using an additional nonce input N to the PRF. This requires a PRF with a larger input, but provides higher security when nonces are unique, without breaking down when they are repeated.

EWCDM: $\boldsymbol{E_{K_3}\left(E_{K_2}(N) \oplus N \oplus H_{K_1}(M)\right)}$. The *Encrypted Wegman-Carter with Davies-Meyer* construction [23] is an alternative construction offering high security with a nonce with graceful degradation when nonces are repeated. Instead of using a $2n$-bit PRF as in WMAC, it uses two calls to an n-bit block cipher.

5.3 Attacking Wegman-Carter MACs

All MACs following the Wegman-Carter construction are exposed to the attack using Deutsch's algorithm that we presented in Section 3.1. More precisely, an IV-respecting quantum adversary can retrieve the value of $H_{K_1}(M_1) \oplus H_{K_1}(M_2)$ for an arbitrary pair of messages M_1, M_2. He can then repeatedly query the tag of M_1 under new nonces, and produce corresponding valid tags for M_2.

When using the generalization with a group operation $\star$ instead of $\oplus$, this simple attack does not apply. In particular, Poly1305-AES uses a modular addition and cannot be broken with Deutsch's algorithm, but we will show a dedicated attack in Section 5.5, using the fact that it is based on polynomial hashing.

5.4 Attacking Algebraic Universal Hash Functions

We can apply our linearization attack to MACs that reuse the same hash key for several messages, whether deterministic (like hash-then-PRF), or nonce-based (like Wegman-Carter, WMAC, and EWCDM). Indeed, it is enough for us to linearize the function H, and the attack applies regardless of the security of the operations that are computed afterwards, even if they involve a nonce.

Many Universal Hash Functions based on algebraic operations have a strong linear structure. In particular, polynomial hashing is a linear function of the message, making it a natural target for Simon's algorithm (in characteristic 2) or Shor's algorithm (in general). We describe concrete attacks against a few constructions.

PolyMAC. PolyMAC [24] is a double block hash-then-sum construction based on polynomial hashing. The generic construction uses two hashing keys K_1, K_2 and two encryption keys K_3, K_4. For an ℓ-block message $m_1, \dots, m_\ell$, this gives:

$$\begin{aligned} \text{PolyMAC}(m_1, \dots, m_\ell) = E_{K_3}\left(K_1 \odot m_\ell \oplus K_1^2 \odot m_{\ell-1} \oplus \dots \oplus K_1^\ell \odot m_1\right) \\ \oplus E_{K_4}\left(K_2 \odot m_\ell \oplus K_2^2 \odot m_{\ell-1} \oplus \dots \oplus K_2^\ell \odot m_1\right) \ . \end{aligned}$$

If a single branch is used, then this looks like the GMAC construction [50] (but without a nonce), using polynomial hashing. GMAC was already attacked in [39] due to its similarities with CBC-MAC, and the fact that the nonce did not influence the embedded hidden shift. However, we can use our attack here. By taking ℓ-block message inputs with blocks 0 or 1, we will recover with Simon's algorithm a period $b_1 \cdots b_\ell$ such that:

$$\bigoplus_i b_i K_1^i = 0 \text{ and } \bigoplus_i b_i K_2^i = 0 \ .$$

This immediately allows a forgery attack, but also, we can recover multiple such periods and solve the corresponding equations to recover K_1 and K_2.

PolyMAC with Modular Additions. Interestingly, our attack applies as well when the polynomial hashing does not use XORs, but modular additions (modulo some value M). However, Simon's algorithm has to be replaced by Shor's algorithm. Note that this is specific to polynomial hashing, and does not apply to LightMAC or PMAC-style constructions in general.

We can define:

$$\begin{aligned}\mathsf{PolyMAC+}(m_1,\dots,m_\ell) = & E_{K_3}\left(K_1 \odot m_\ell + K_1^2 \odot m_{\ell-1} + \dots + K_1^\ell \odot m_1 \bmod M\right) \\ & \oplus E_{K_4}\left(K_2 \odot m_\ell + K_2^2 \odot m_{\ell-1} + \dots + K_2^\ell \odot m_1 \bmod M\right) \enspace .\end{aligned}$$

In that case, we can remark that there exists periods $a_1,\dots,a_\ell$ such that:

$$K_1 a_1 + \dots + K_1^\ell a_\ell \bmod M = 0 \text{ and } K_2 a_1 + \dots + K_2^\ell a_\ell \bmod M = 0 \enspace .$$

More precisely, these periods form a lattice in $\mathbb{Z}_M^\ell$, and for all of them, we have:

$$\forall m_1,\dots,m_\ell, \mathsf{PolyMAC+}(m_1+a_1,\dots,m_\ell+a_\ell) = \mathsf{PolyMAC+}(m_1,\dots,m_\ell) \enspace .$$

Thus, the generalization by Mosca and Ekert [52] of Shor's algorithm allows to retrieve the full lattice of these periods: we can not only forge, but also recover the internal hashing keys.

GCM-SIV2. This is a double-block variant of GCM-SIV defined in [36]. The tag generation combines two independent polynomial hashes (with two keys K_1, K_2) with a keyed-dependent combination function F_K, of which we shall not study the details. This mode is nonce-based. With an empty associated data, the tag is computed as follows:

$$\begin{aligned}&\mathsf{GCM\text{-}SIV2} - \mathsf{MAC}(N, m_1,\dots,m_\ell) = \\ &\qquad F_K\left(N \oplus H_{K_1}(m_1,\dots,m_\ell), N \oplus H_{K_2}(m_1,\dots,m_\ell)\right) \enspace ,\end{aligned}$$

where H_{K_1} and H_{K_2} are polynomial hashes (this would be similar for the tag of an empty message, replacing M by the associated data). Thus, although the MAC is nonce-dependent, it falls into our framework since the periods of the polynomial hashes remain independent of N.

Other algebraic hashing constructions. There are many alternatives to polynomial hashing based on field operations. Several constructions are linear, such as the dot product construction, and Toeplitz hashing [42].[4]

Some other constructions can be linearized using specially crafted messages.

NMH [31].* The NMH* universal hash function is defined as:

$$\mathrm{NMH}^*(M) = \sum (m_{2i} + K_{2i})(m_{2i+1} + K_{2i+1}) \bmod p$$

If we consider messages with blocks with an even index set to arbitrary constants, we obtain a linear function of the odd message blocks. Therefore, Shor's algorithm can break MACs based on this hash function that reuse the hash key.

[4] Grain128A and Grain128AEAD [32] use Toeplitz hashing, but we can only attack them in the nonce-misuse setting because they use the one-time-MAC construction.

BRW Hashing [7]. The BRW universal hash function is based on a class of polynomials that can be evaluated with $\ell/2$ multiplications with ℓ inputs, using a single key. The construction is defined recursively, depending on the input length:

- $BRW_K() = 0$
- $BRW_K(m_1) = m_1$
- $BRW_K(m_1, m_2) = m_1 \odot K + m_2$
- $BRW_K(m_1, m_2, m_3) = (K + m_1) \odot (K^2 + m_2) + m_3$
- $BRW_K(m_1, m_2, \dots m_\ell) = BRW_K(m_1, m_2, \dots m_{t-1}) \odot (K^t + m_t) +$ $BRW_K(m_{t+1}, m_{t+2}, \dots m_\ell)$ with t a power of 2, and $4 \le t \le n < 2t$.

For instance, with 8 inputs, we obtain

$$\Big(\big((K+m_1)\odot(K^2+m_2)+m_3\big)\odot(K^4+m_4)+(K+m_5)\odot(K^2+m_6)+m_7\Big)\odot(K^8+m_8)$$

This construction can also be linearized by setting message blocks with an even index set to arbitrary constants.

5.5 Period-Finding against Poly1305

Poly1305 [6] is a polynomial MAC with some specific constraints that force a dedicated analysis. It has already been cryptanalysed in [18], where the authors proposed an attack in 2^{38} time and queries. The authors managed to overcome the specific constraints by leveraging a *hidden shift* structure. The attack we propose here is drastically more efficient, and uses a *hidden period* instead.

Poly1305 uses a hashing key r of 124 bits with at most 106 non-zero bits and a 128-bit cipher key K. The MAC of a message $m_1, \dots, m_\ell$ with the nonce N is computed as:

$$\begin{aligned}\mathsf{Poly1305}(m_1, \dots, m_\ell) = (c_1 r^\ell + c_2 r^{\ell-1} + \dots + c_\ell r^1) \bmod 2^{130} - 5 \\ + AES_K(N) \bmod 2^{128} \quad ,\end{aligned}$$

where $c_1, \dots, c_\ell$ are the padded message blocks obtained from the message blocks $m_1, \dots, m_\ell$. When message blocks are full 128-bit blocks, the c_i are simply obtained from the m_i by adding 2^{128}.

Let us assume that we query with two message blocks. We have:

$$\begin{aligned}\mathsf{Poly1305}(m_1, m_2) &= \big((m_1 + 2^{128}) \cdot r^2 + (m_2 + 2^{128}) \cdot r\big) \bmod 2^{130} - 5 \\ &\qquad + AES_K(N) \bmod 2^{128} \\ &= \big(((m_1 \cdot r + m_2) \cdot r + C_1) \bmod 2^{130} - 5\big) + C_2 \bmod 2^{128} \quad ,\end{aligned}$$

where C_1, C_2 are constants of our query that depend on r, K, N. Since the computation ends with a reduction modulo 2^{128}, which is smaller than $2^{130} - 5$, we must actually use a *compressed* instance of Shor's algorithm [48]. This increases mildly the number of queries, by less than a factor 2.

Two inputs (m_1, m_2) and (m'_1, m'_2) lead to the same tag if

$$m_1 r + m_2 = m'_1 r + m'_2 \bmod 2^{130} - 5$$
$$\Leftrightarrow (m_1 - m'_1)r + (m_2 - m'_2) = 0 \bmod 2^{130} - 5 \ .$$

Hence, the periods of the function $\mathsf{Poly1305}(m_1, m_2)$ are solutions of $m_1 r + m_2 = 0 \bmod 2^{130} - 5$.

As the period is modulo $2^{130} - 5$ but the input is 128-bit long, we cannot do the query expected by Shor's algorithm. Still, the fraction of inputs we can actually query is large enough so that we can still apply Shor's algorithm with a *partial* query, and recover efficiently the period.

The initial query is:

$$\frac{1}{2^{128}} \sum_{m_1, m_2 = 0}^{2^{128}-1} |m_1\rangle |m_2\rangle |\mathsf{Poly1305(m_1, m_2)}\rangle$$
$$= \frac{1}{2^{128}} \sum_{m_1, m_2 = 0}^{2^{128}-1} |m_1\rangle |m_2\rangle |f(m_1 r + m_2)\rangle \ .$$

Here, f is a function that depends on r, K, N. The only relevant point is that it does not depend on m_1, m_2 directly, but only on $m_1 r + m_2$. For simplicity, in the following we assume f is a permutation. We will now apply the QFT over $\mathbb{Z}/(2^{130} - 5)$ on the input registers. We note $p = 2^{130} - 5$. We obtain

$$\frac{1}{p} \frac{1}{2^{128}} \sum_{m_1, m_2 = 0}^{2^{128}-1} \sum_{x, y = 0}^{p-1} \exp\left(2i\pi \frac{x m_1 + y m_2}{p}\right) |x\rangle |y\rangle |f(m_1 r + m_2)\rangle \ .$$

We can rewrite the state by regrouping components with identical $m_1 r + m_2$:

$$\frac{1}{p} \frac{1}{2^{128}} \sum_{x, y = 0}^{p-1} \sum_{c = 0}^{p-1} \sum_{\substack{m_1, m_2 = 0 \\ m_1 r + m_2 = c}}^{2^{128-1}} \exp\left(2i\pi \frac{x m_1 + y m_2}{p}\right) |x\rangle |y\rangle |f(c)\rangle$$
$$= \frac{1}{p} \frac{1}{2^{128}} \sum_{x, y = 0}^{p-1} \sum_{c = 0}^{p-1} \sum_{\substack{m_1, m_2 = 0 \\ m_1 r + m_2 = c}}^{2^{128-1}} \exp\left(2i\pi \frac{x m_1 + y(c - m_1 r)}{p}\right) |x\rangle |y\rangle |f(c)\rangle$$
$$= \frac{1}{p} \frac{1}{2^{128}} \sum_{x, y = 0}^{p-1} \sum_{c = 0}^{p-1} \exp\left(2i\pi \frac{yc}{p}\right) \sum_{\substack{m_1, m_2 = 0 \\ m_1 r + m_2 = c}}^{2^{128-1}} \exp\left(2i\pi \frac{m_1(x - yr)}{p}\right) |x\rangle |y\rangle |f(c)\rangle$$

Now, we can compute the probability to measure a nonzero tuple (x, y) with $x = yr$.

As there are $p-1$ such tuples, the overall probability is

$$\left(\frac{1}{p}\frac{1}{2^{128}}\right)^2 (p-1)\sum_{c=0}^{p-1}\left(\sum_{\substack{m_1,m_2=0\\ m_1r+m_2=c}}^{2^{128-1}} 1\right)^2$$

$$= \frac{p-1}{p^2 2^{256}}\sum_{c=0}^{p-1}\left(\#\{0 \le m_1, m_2 < 2^{128} : m_1 r + m_2 = c\}\right)^2$$

Now, as $x \mapsto x^2$ is a convex function, we can use Jensen's inequality:

$$\sum_{i=1}^{n}\frac{1}{n}\alpha_i^2 \ge \left(\sum_{i=1}^{n}\frac{1}{n}\alpha_i\right)^2 \quad .$$

This allows us to lower bound the previous probability by

$$\frac{p-1}{p^2 2^{256}} p \left(\sum_{c=0}^{p-1}\frac{1}{p}\#\{0 \le m_1, m_2 < 2^{128} : m_1 r + m_2 = c\}\right)^2$$

$$= \frac{p-1}{p 2^{256}}\left(\frac{1}{p}\#\{0 \le m_1, m_2 < 2^{128}\}\right)^2 = \frac{p-1}{p 2^{256}}\left(\frac{2^{256}}{p}\right)^2 = \frac{(p-1)2^{256}}{p^3} > \frac{1}{16} \quad .$$

Thus, we measure a tuple $(x, y) \neq (0, 0)$ with $x = yr$ with probability at least 1/16. As $2^{130} - 5$ is prime, one such tuple is enough to recover r. Hence, we need at most 16 queries on average to recover r, assuming f is a permutation. Here, as f is a function, we rely on [48] to bound the increase by a factor 2. Note that as we are only a few bits of output short of having a permutation, this is a very loose bound. Overall, the attack will require no more than 32 queries.

6 On Parallelizable Quantum PRFs

Let us take a broader point of view. The deterministic MACs that we attacked in this paper all have common points. Besides allowing inputs of any length (as should be expected of any MAC construction), they • process their input blocks independently; • compute one or more linear functions, *with XORs*, of these processed input blocks; • process the authentication tag from the outputs of these linear functions.

These characteristics are to be expected from any MAC that is: • of average rate one, meaning that there are as many primitive calls as there are blocks; • parallelizable; • having an internal state of size $\mathcal{O}(n)$, independent of the query length. Our attack is easily defeated if the blocks are processed sequentially by calling a compression function, as in the NMAC construction. However, the construction becomes unparallelizable.

It may be possible to obtain a quantum-secure parallelizable qPRF using a tree hashing, where the blocks are placed at the leaves of a binary tree, and each node is computed by calling a (keyed) compression function on its two children. However, such a construction requires an internal state greater than $\mathcal{O}(n)$, and that increases with the amount of data. Typically to traverse the binary tree, we will need to remember $\mathcal{O}(\log m)$ nodes, where m is the number of leaves.

Open Question. If we stand by the characteristics listed above (efficient, parallelizable, constant internal state size), then it seems that the only solution is to use modular additions in place of XORs in the constructions that we attacked. In that case, our attack does not seem to work anymore, due to the fact that modular additions, contrary to XORs, are not involutive. Thus changing one of the blocks in our n-block queries does not modify involutively the result, which breaks the periodicity property that we used with Simon's algorithm.

This makes this option worth investigating, both from a provable security and a cryptanalysis perspective. Note that the situation is different from most attacks with Simon's algorithm, where the replacement of XORs by + changes the attack complexity from polynomial to subexponential (see [2,18]). In our case, it is possible that using + allows an exponential security level.

7 Conclusion

In this paper, we introduced a novel way of using quantum period-finding to break parallelizable MAC constructions in the superposition query model, breaking most of them in this setting. In full generality, our attack makes use of multiple blocks to embed a hidden period, a surprisingly simple idea that might have other applications. We gave new polynomial-time forgery or partial key-recovery attacks on LightMAC, LightMAC+, PolyMAC, Poly1305, GCM-SIV2, Deoxys, ZMAC, PMAC_TBC3k. Our attack is not mitigated by the use of multiple parallel branches (as in double-block hash-then-sum MACs). It can be prevented for IV-based MACs if the non-reused IV intervenes in the processing of all message blocks (as done in [9]).

These results show that we cannot obtain a parallelizable quantum-secure PRF by processing independently the message blocks, XORing the results, and then hashing the output. If modular additions are used instead of XORs, our attack does not apply anymore (except on polynomial hashing, which has a simpler structure). Overcoming this limitation, or on the contrary, proving the security of such a PRF, is an interesting open question.

Acknowledgments This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. 714294 - acronym QUASYModo). A.S. is supported by ERC-ADG-ALGSTRONGCRYPTO (project 740972).

References

1. Alagic, G., Majenz, C., Russell, A., Song, F.: Quantum-access-secure message authentication via blind-unforgeability. In: EUROCRYPT (3). Lecture Notes in Computer Science, vol. 12107, pp. 788–817. Springer (2020)

2. Alagic, G., Russell, A.: Quantum-secure symmetric-key cryptography based on hidden shifts. In: EUROCRYPT (3). Lecture Notes in Computer Science, vol. 10212, pp. 65–93 (2017)
3. Anand, M.V., Targhi, E.E., Tabia, G.N., Unruh, D.: Post-quantum security of the CBC, CFB, OFB, CTR, and XTS modes of operation. In: PQCrypto. Lecture Notes in Computer Science, vol. 9606, pp. 44–63. Springer (2016)
4. Bellare, M., Guérin, R., Rogaway, P.: XOR MACs: New methods for message authentication using finite pseudorandom functions. In: CRYPTO. Lecture Notes in Computer Science, vol. 963, pp. 15–28. Springer (1995)
5. Bernstein, D.J.: How to stretch random functions: The security of protected counter sums. J. Cryptol. 12(3), 185–192 (1999)
6. Bernstein, D.J.: The Poly1305-AES message-authentication code. In: FSE. Lecture Notes in Computer Science, vol. 3557, pp. 32–49. Springer (2005)
7. Bernstein, D.J.: Polynomial evaluation and message authentication (2007), http://cr.yp.to/papers.html#pema
8. Bernstein, E., Vazirani, U.V.: Quantum complexity theory. SIAM J. Comput. 26(5), 1411–1473 (1997)
9. Bhaumik, R., Bonnetain, X., Chailloux, A., Leurent, G., Naya-Plasencia, M., Schrottenloher, A., Seurin, Y.: QCB: efficient quantum-secure authenticated encryption. IACR Cryptol. ePrint Arch. 2020, 1304 (2020)
10. Black, J., Cochran, M.: MAC reforgeability. In: FSE. Lecture Notes in Computer Science, vol. 5665, pp. 345–362. Springer (2009)
11. Black, J., Rogaway, P.: CBC MACs for arbitrary-length messages: The three-key constructions. In: CRYPTO. LNCS, vol. 1880, pp. 197–215. Springer (2000)
12. Black, J., Rogaway, P.: A block-cipher mode of operation for parallelizable message authentication. In: Knudsen, L.R. (ed.) EUROCRYPT. LNCS, vol. 2332, pp. 384–397. Springer (2002)
13. Boneh, D., Zhandry, M.: Quantum-secure message authentication codes. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 7881, pp. 592–608. Springer (2013)
14. Boneh, D., Zhandry, M.: Secure signatures and chosen ciphertext security in a quantum computing world. In: CRYPTO (2). Lecture Notes in Computer Science, vol. 8043, pp. 361–379. Springer (2013)
15. Bonnetain, X.: Quantum key-recovery on full AEZ. In: SAC. Lecture Notes in Computer Science, vol. 10719, pp. 394–406. Springer (2017)
16. Bonnetain, X.: Tight bounds for Simon's algorithm. IACR Cryptol. ePrint Arch. 2020, 919 (2020)
17. Bonnetain, X., Hosoyamada, A., Naya-Plasencia, M., Sasaki, Y., Schrottenloher, A.: Quantum attacks without superposition queries: The offline Simon's algorithm. In: ASIACRYPT (1). Lecture Notes in Computer Science, vol. 11921, pp. 552–583. Springer (2019)
18. Bonnetain, X., Naya-Plasencia, M.: Hidden shift quantum cryptanalysis and implications. In: ASIACRYPT (1). Lecture Notes in Computer Science, vol. 11272, pp. 560–592. Springer (2018)
19. Bonnetain, X., Naya-Plasencia, M., Schrottenloher, A.: On quantum slide attacks. In: SAC. Lecture Notes in Computer Science, vol. 11959, pp. 492–519. Springer (2019)
20. Brassard, G., Høyer, P.: An exact quantum polynomial-time algorithm for Simon's problem. In: ISTCS. pp. 12–23. IEEE Computer Society (1997)
21. Carter, L., Wegman, M.N.: Universal classes of hash functions (extended abstract). In: STOC. pp. 106–112. ACM (1977)

22. Cid, C., Hosoyamada, A., Liu, Y., Sim, S.M.: Quantum cryptanalysis on contracting feistel structures and observation on related-key settings. In: INDOCRYPT. Lecture Notes in Computer Science, vol. 12578, pp. 373–394. Springer (2020)
23. Cogliati, B., Seurin, Y.: EWCDM: an efficient, beyond-birthday secure, nonce-misuse resistant MAC. In: CRYPTO (1). Lecture Notes in Computer Science, vol. 9814, pp. 121–149. Springer (2016)
24. Datta, N., Dutta, A., Nandi, M., Paul, G.: Double-block hash-then-sum: A paradigm for constructing BBB secure PRF. IACR Trans. Symmetric Cryptol. 2018(3), 36–92 (2018)
25. Datta, N., Dutta, A., Nandi, M., Paul, G., Zhang, L.: Single key variant of PMAC_Plus. IACR Trans. Symmetric Cryptol. 2017(4), 268–305 (2017)
26. Deutsch, D.: Quantum theory, the church–turing principle and the universal quantum computer. In: Proceedings of the Royal Society London A. vol. 400, pp. 97—-117. Springer (1985)
27. Deutsch, D., Jozsa, R.: Rapid solution of problems by quantum computation. Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences 439(1907), 553–558 (1992)
28. Dong, X., Dong, B., Wang, X.: Quantum attacks on some feistel block ciphers. Des. Codes Cryptogr. 88(6), 1179–1203 (2020)
29. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: STOC. pp. 212–219. ACM (1996)
30. Guo, T., Wang, P., Hu, L., Ye, D.: Attacks on beyond-birthday-bound macs in the quantum setting. In: PQCrypto. Lecture Notes in Computer Science, vol. 12841, pp. 421–441. Springer (2021)
31. Halevi, S., Krawczyk, H.: MMH: software message authentication in the gbit/second rates. In: FSE. Lecture Notes in Computer Science, vol. 1267, pp. 172–189. Springer (1997)
32. Hell, M., Johansson, T., Meier, W., Sönnerup, J., Yoshida, H.: Grain-128 AEAD a lightweight AEAD streamcipher. Submission to NIST-LWC (2nd Round) (2019)
33. Hosoyamada, A., Sasaki, Y.: Quantum demiric-selçuk meet-in-the-middle attacks: Applications to 6-round generic feistel constructions. In: SCN. Lecture Notes in Computer Science, vol. 11035, pp. 386–403. Springer (2018)
34. Ito, G., Hosoyamada, A., Matsumoto, R., Sasaki, Y., Iwata, T.: Quantum chosen-ciphertext attacks against feistel ciphers. In: CT-RSA. Lecture Notes in Computer Science, vol. 11405, pp. 391–411. Springer (2019)
35. Iwata, T., Kurosawa, K.: OMAC: one-key CBC MAC. In: FSE. LNCS, vol. 2887, pp. 129–153. Springer (2003)
36. Iwata, T., Minematsu, K.: Stronger security variants of GCM-SIV. IACR Trans. Symmetric Cryptol. 2016(1), 134–157 (2016)
37. Iwata, T., Minematsu, K., Peyrin, T., Seurin, Y.: ZMAC: A fast tweakable block cipher mode for highly secure message authentication. In: CRYPTO (3). Lecture Notes in Computer Science, vol. 10403, pp. 34–65. Springer (2017)
38. Jean, J., Nikolic, I., Peyrin, T., Seurin, Y.: Deoxys v1. 41. Submitted to CAESAR (2016)
39. Kaplan, M., Leurent, G., Leverrier, A., Naya-Plasencia, M.: Breaking symmetric cryptosystems using quantum period finding. In: CRYPTO (2). Lecture Notes in Computer Science, vol. 9815, pp. 207–237. Springer (2016)
40. Kaplan, M., Leurent, G., Leverrier, A., Naya-Plasencia, M.: Quantum differential and linear cryptanalysis. IACR Trans. Symmetric Cryptol. 2016(1), 71–94 (2016)

41. Kim, S., Lee, B., Lee, J.: Tight security bounds for double-block hash-then-sum MACs. In: EUROCRYPT (1). Lecture Notes in Computer Science, vol. 12105, pp. 435–465. Springer (2020)
42. Krawczyk, H.: New hash functions for message authentication. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 921, pp. 301–310. Springer (1995)
43. Krovetz, T., Rogaway, P.: The software performance of authenticated-encryption modes. In: FSE. Lecture Notes in Computer Science, vol. 6733, pp. 306–327. Springer (2011)
44. Kuwakado, H., Morii, M.: Quantum distinguisher between the 3-round feistel cipher and the random permutation. In: ISIT. pp. 2682–2685. IEEE (2010)
45. Kuwakado, H., Morii, M.: Security on the quantum-type even-mansour cipher. In: ISITA. pp. 312–316. IEEE (2012)
46. Leander, G., May, A.: Grover meets Simon - quantumly attacking the FX-construction. In: ASIACRYPT (2). Lecture Notes in Computer Science, vol. 10625, pp. 161–178. Springer (2017)
47. Luykx, A., Preneel, B., Tischhauser, E., Yasuda, K.: A MAC mode for lightweight block ciphers. In: FSE. Lecture Notes in Computer Science, vol. 9783, pp. 43–59. Springer (2016)
48. May, A., Schlieper, L.: Quantum period finding is compression robust. CoRR abs/1905.10074 (2019)
49. McGrew, D.A., Viega, J.: The security and performance of the Galois/Counter Mode (GCM) of operation. In: INDOCRYPT. LNCS, vol. 3348, pp. 343–355. Springer (2004)
50. McGrew, D.A., Viega, J.: The security and performance of the galois/counter mode (GCM) of operation. In: INDOCRYPT. Lecture Notes in Computer Science, vol. 3348, pp. 343–355. Springer (2004)
51. Minematsu, K.: A lightweight alternative to PMAC. In: SAC. Lecture Notes in Computer Science, vol. 11959, pp. 393–417. Springer (2019)
52. Mosca, M., Ekert, A.: The hidden subgroup problem and eigenvalue estimation on a quantum computer. In: QCQC. Lecture Notes in Computer Science, vol. 1509, pp. 174–188. Springer (1998)
53. Naito, Y.: Full PRF-secure message authentication code based on tweakable block cipher. In: ProvSec. Lecture Notes in Computer Science, vol. 9451, pp. 167–182. Springer (2015)
54. Naito, Y.: Blockcipher-based MACs: beyond the birthday bound without message length. In: ASIACRYPT (3). Lecture Notes in Computer Science, vol. 10626, pp. 446–470. Springer (2017)
55. Nielsen, M.A., Chuang, I.: Quantum computation and quantum information (2002)
56. NIST: Submission requirements and evaluation criteria for the post-quantum cryptography standardization process (2016), https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf
57. Rogaway, P.: Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In: ASIACRYPT. Lecture Notes in Computer Science, vol. 3329, pp. 16–31. Springer (2004)
58. Santoli, T., Schaffner, C.: Using Simon's algorithm to attack symmetric-key cryptographic primitives. Quantum Inf. Comput. 17(1&2), 65–78 (2017)
59. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: FOCS. pp. 124–134. IEEE Computer Society (1994)
60. Simon, D.R.: On the power of quantum computation. SIAM J. Comput. 26(5), 1474–1483 (1997)

61. Song, F., Yun, A.: Quantum security of NMAC and related constructions - PRF domain extension against quantum attacks. In: CRYPTO (2). Lecture Notes in Computer Science, vol. 10402, pp. 283–309. Springer (2017)
62. Yasuda, K.: The sum of CBC MACs is a secure PRF. In: CT-RSA. Lecture Notes in Computer Science, vol. 5985, pp. 366–381. Springer (2010)
63. Zhang, L., Wu, W., Sui, H., Wang, P.: 3kf9: Enhancing 3GPP-MAC beyond the birthday bound. In: ASIACRYPT. Lecture Notes in Computer Science, vol. 7658, pp. 296–312. Springer (2012)