Quantum Cryptanalysis of OTR and OPP: Attacks on Confidentiality, and Key-Recovery

Melanie Jauch and Varun Maram

Department of Computer Science, ETH Zurich, Switzerland
mjauch@student.ethz.ch, vmaram@inf.ethz.ch

Abstract. In this paper, we analyze the security of authenticated encryption modes OTR (Minematsu, Eurocrypt 2014) and OPP (Granger *et al.*, Eurocrypt 2016) in a setting where an adversary is allowed to make encryption queries in *quantum* superposition. Starting with OTR – or more technically, AES-OTR, a third-round CAESAR candidate – we extend prior quantum attacks on the mode's unforgeability in the literature to provide the first attacks breaking confidentiality, i.e., IND-qCPA security, of AES-OTR in different settings depending on how the associated data is processed. On a technical level, one of our IND-qCPA attacks involves querying the quantum encryption oracle on a superposition of data with *unequal* length; to the best of our knowledge, such an attack has never been modelled before in the (post-)quantum cryptographic literature, and we hence believe our technique is of independent interest. Coming to OPP, we present the first *key-recovery* attack against the scheme which uses only a *single* quantum encryption query.

Keywords: AES-OTR, OPP, Authenticated Encryption, IND-qCPA Security, Key-Recovery, Simon's Algorithm, Deutsch's Algorithm

1 Introduction

With the development of large-scale quantum computers on the horizon, the security of widely-deployed cryptographic systems faces new threats. Specifically, public-key cryptosystems that rely on the hardness of factorization or computing discrete-logs would suffer from devastating attacks based on Shor's algorithm [28]. This has led to the setting of so-called *post-quantum* secure public-key cryptography gaining a lot of attention over the last few years, which includes efforts by NIST [3] to standardize such quantum-resistant algorithms.

Coming to the *symmetric-key* cryptography setting however, the impact of quantum computers has been assumed to be significantly less severe for a long time. It was widely believed that quantum attacks on symmetric primitives such as block ciphers would only improve by a quadratic speed up due to Grover's algorithm [12], and that we cannot do any better. Naturally, it was hence assumed that it is enough to simply double the key size of the affected primitives to restore the same level of security as in the classical setting. However, the community

quickly realized that it is not sufficient to just consider the quantum security of standalone primitives such as block ciphers – since they are rarely used in isolation in practice – but to also consider their associated *modes of operation*. Such modes are typically designed to provide enhanced security guarantees such as confidentiality, integrity, and authenticity of encrypted messages. In this paper, we will be focusing on modes related to authenticated encryption (AE).

Starting with the work of Kaplan *et al.* [17] which showed how to break the authenticity guarantees of classical AE modes such as GCM and OCB in the quantum setting in *polynomial-time* (in contrast to the generic quadratic speed up offered by Grover's algorithm), follow-up works by Bhaumik *et al.* [4] and Bonnetain *et al.* [7] improved the quantum attacks against the latter OCB modes, albeit still targeting authenticity. At a high-level, the above attacks fundamentally rely on other well-known quantum algorithms such as Simon's period finding algorithm [29] and Deutsch's algorithm [8]. Subsequently, Maram *et al.* [22] extended the aforementioned authenticity attacks to also break confidentiality of the OCB modes in the quantum setting; more formally, the authors targeted a quantum security notion called "IND-qCPA security" [5], which differs from the classical IND-CPA security notion in that the adversary is allowed to make quantum encryption queries. It is worth pointing out that the practicality of this model (also generically called "Q2 security" in the literature) where the attacker has quantum access to the secret-keyed encryption functionality is still currently debated in the community (e.g., see [17,4,2,16,6]); however we consider this discussion beyond the scope of our work.

Now focusing on the OCB modes, they are one of the most well-studied and widely influential classical AE modes. OCB has three versions: OCB1 [27], OCB2 [26] and OCB3 [18]. While OCB1 and OCB3 are provably secure AE schemes in the classical setting, a breakthrough result by Inoue *et al.* [13] showed that OCB2 is classically broken as an AE mode. More specifically, Inoue and Minematsu [14] first came up with classical attacks on the authenticity guarantees of OCB2. After their attacks became public, Poettering [25] and Iwata [15] proceeded to extend them to also break confidentiality of OCB2 in the classical setting. In this context, the aforementioned work of Maram *et al.* [22] can be seen as translating this strategy to the quantum setting to break (IND-qCPA) confidentiality of the three OCB modes starting with the quantum forgery attacks in [17,4,7]. However at the same time, Inoue *et al.* [13] observed that their classical attacks on OCB2 do not extend to other popular AE designs based on OCB such as OTR [23] and OPP [11]; this is due to some subtle differences in the structures of these modes when compared to OCB. In this paper, we analyze if one would observe something similar regarding the effects of quantum insecurity of OCB on the OTR and OPP modes.

Starting with OTR [23], it is technically a block cipher mode to realize a nonce-based authenticated encryption with associated data (AEAD) scheme. We specifically focus on an instantiation of the mode with AES as the underyling block cipher called AES-OTR [24], which also includes an additional way to process associated data when compared to the generic OTR. AES-OTR was also

a third-round candidate in the CAESAR competition [1] aimed at standardizing a portfolio of authenticated encryption schemes. In the classical setting, AES-OTR was shown to offer provable security guarantees as an AE scheme [24]. Coming to the quantum setting however, works by Kaplan *et al.* [17] and Chang *et al.* [20] proposed ways to attack the authenticity guarantees of AES-OTR in the quantum superposition model using Simon's algorithm. However the confidentiality of AES-OTR in the same quantum setting has not been addressed in the literature. Hence, following the work of Maram *et al.* [22] on the quantum (IND-qCPA) confidentiality of OCB2, and given the status of AES-OTR as a third-round CAESAR candidate, this leads us to pose the question:

Is the AES-OTR mode IND-qCPA secure?

Coming to OPP [11], it is a (public) permutation-based AE mode unlike OCB and OTR. OPP essentially generalizes OCB3 by replacing the underlying block cipher by an efficiently invertible public permutation and a different form of masking. This ensures fast encryption and full parallelization, hence making OPP an ideal candidate in applications requiring high efficiency of underlying primitives. However in contrast to the OCB and OTR modes, the quantum security of OPP – related to either authenticity or confidentiality – has not been analyzed in the literature at all. This motivates us to ask the following question:

Is the OPP mode quantum secure?

1.1 Our Contributions

In this paper, we answer the above questions concerning the quantum security of both AES-OTR and OPP modes by presenting tailor-made quantum attacks. Along the way, some of our attacks involve techniques that we believe will be of independent interest to the broader (post-)quantum cryptographic community. Our concrete results are listed below:

Attacks on IND-qCPA Security of AES-OTR. In Section 3, we present the first IND-qCPA attacks against AES-OTR. Specifically, our quantum attacks are tailored to three different settings depending on how the associated data (AD) is processed: namely, the settings with parallel and serial processing of AD, and the setting where no AD is used at all.

For the first two settings with non-empty AD, our attacks work in the weak adversarial setting where the nonces used by the challenger to answer encryption queries in the IND-qCPA security game are generated uniformly at random (instead of the nonces being chosen by the adversary). On a high level, the attack breaking IND-qCPA security w.r.t. parallel AD processing uses Simon's algorithm as well as Deutsch's algorithm to gain *raw block cipher access*, i.e., the ability to evaluate the underlying block cipher on arbitrary inputs. With this access, it is straightforward to break IND-qCPA security. This attack strategy is similar to that used in [22] to break confidentiality of the OCB modes. However,

we need to make an extra assumption for our attack to be efficient: namely that the authentication tags produced by the AES-OTR encryption oracle are not (significantly) truncated. It is worth noting that the specification of AES-OTR [24] recommends parameters with untruncated tags. But interestingly, in the process we were also able to point out a gap in the quantum cryptanalysis of OCB2's confidentiality (with non-empty AD) in [22], since the corresponding IND-qCPA attack there uses a similar assumption of untruncated tags *implicitly*. This was later confirmed by one of the authors [21].

Coming to the setting where no AD is used in AES-OTR, our attack assumes a stronger adversarial setting where the adversary is now allowed to adaptively choose the (classical) nonces for its quantum encryption queries in the IND-qCPA security game. This setting is the same as considered in [22, Section 4.4] with respect to their IND-qCPA attack against OCB2 as a "pure" AE (i.e., no AD) scheme. However, what makes our attack a non-trivial extension of the above attack on OCB2 is that for AES-OTR there is an additional formatting function applied to the nonce before it is AES-encrypted. We thus have to perform some additional steps that increase the overall complexity of our attack. The attack is described in detail in Appendix D.

Quantum Queries over *Unequal*-Length Data. Our IND-qCPA attack on AES-OTR w.r.t. serial AD processing involves a novel paradigm which, to the best of our knowledge, has never been considered in the (post-)quantum cryptanalytic literature. Note that in the IND-qCPA security definition, an adversary is allowed to make encryption queries on a quantum superposition of data. However, according to the laws of quantum physics, a superposition is defined only over states with the same number of qubits. Hence in our IND-qCPA scenario, this translates to the seemingly implicit restriction of an adversary only being able to make superposition queries over *equal*-length data.

In our work, we show how to overcome this restriction by modelling the quantum encryption oracle in the IND-qCPA security game in a way which allows an adversary to also make superposition queries over *unequal*-length data (see Section 3.3 for more details). Furthermore, to give evidence of the power of this new quantum cryptanalytic paradigm, we show how an adversary can immediately gain raw block cipher access in our IND-qCPA attack on AES-OTR with serial AD processing using *only* Simon's algorithm; this is in contrast to the IND-qCPA attacks against OCB in [22] and against AES-OTR with parallel AD processing discussed above which also require Deutsch's algorithm to obtain this raw access. It's also worth pointing out that using this novel paradigm, we no longer have to rely on the above extra assumption of the AES-OTR authentication tags being untruncated in the serial AD case. Finally, our paradigm can also be extended to cryptanalysis in the more realistic *post*-quantum setting, i.e., where the adversary has quantum access only to *public* cryptographic oracles (also called "Q1 security" in the literature) such as hash functions in the so-called Quantum Random Oracle Model (QROM).

A Quantum Key-Recovery Attack on OPP. We present the first quantum key-recovery attack on OPP in Section 4. Our attack is conducted in the weak adversarial setting similar to our IND-qCPA attacks on AES-OTR, where the nonces are chosen uniformly at random by the challenger. In contrast to AES-OTR being based on a block cipher which may only be inverted knowing the key, OPP is built upon an efficiently invertible public permutation P. We exploit this specific property to formulate our key recovery attack. On a high level, we are able to recover a value $\Omega = P(\ldots||K)$ using only a *single* quantum encryption query via an application of Simon's algorithm, where K is the key used in OPP; hence, applying P^{-1} to Ω allows us to recover the key K.

2 Preliminaries

Notation. Denote by $\{0,1\}^*$ the set of all finite-length bit strings and $\{0,1\}^{8*}$ the set of all finite-length byte strings. We let the parameters n, k, τ, $\kappa \geq 0$ define the block length, the size of the key, tag, and nonce respectively. For $b \in \mathbb{N}$ we let $[b] := \{1, ..., b\}$. Given $x, y \in \{0,1\}^*$, the concatenation of x and y is denoted as $x||y$. We let the length of x in bits be denoted as $|x|$ and we define $|x|_b := \max\{1, \lceil X/b \rceil\}$. We use the symbols $\oplus, \ll, \gg, \lll, \ggg$ to denote bit-wise XOR, left-shift, right-shift, left-rotation and right-rotation, respectively.

Further, we define the following padding function that, for a given input X, extends it to a desired length m

$$\mathrm{pad}_m^0 : \{0,1\}^{\leq m} \to \{0,1\}^m, X \mapsto X||0^{m-|X|}$$

and for $0 \leq |X| < m$, we write $\underline{X} = X||10^{m-|X|-1}$ as the 10^* padding.

By $\mathrm{msb}_l(x)$ we mean the sequence of first l left-most bits of the bit sting x and for any non-negative integer q, let $\mathrm{bin}(q, m)$ denote the standard m-bit encoding of q.

We want to highlight the difference in notation for the encryption functions used for AES-OTR in Section 3 and for OPP in Section 4. By OTR-$\mathcal{E}_{K,\cdot}(\cdot)$ we indicate the encryption algorithm of AES-OTR with an underlying block cipher (AES to be precise) encryption function E_K with key K. On the other hand we use OPP-$\mathcal{E}(K, \cdot)$ to denote the encryption algorithm of OPP with key K that uses a public permutation instead of a block cipher.

In the context of AES-OTR we use notations such as $2X, 3X$ or $7X$ for an n-bit string X. Following [24], we here interpret X as a coefficient vector of the polynomial in $\mathrm{GF}(2^n)$. So by $2X$ we essentially mean multiplying the generator of the field $\mathrm{GF}(2^n)$, which is the polynomial x, and X over $\mathrm{GF}(2^n)$. This process is referred to as *doubling*. Similarly, 2^iX denotes i-times doubling X and we denote $3X = X \oplus 2X$ as well as $7X = 2^2X \oplus 2X \oplus X$. Field multiplication over $\mathrm{GF}(2^n)$ for $n = 128$ can be implemented as

$$2X = \begin{cases} X \ll 1 & \text{if } \mathrm{msb}_1 X = 0. \\ (X \ll 1) \oplus 0^{120}10000111 & \text{if } \mathrm{msb}_1 X = 1. \end{cases}$$

We omit the details here and refer to [24] for further details.

Simon's Algorithm. Simon's algorithm is a quantum algorithm that is able to solve the following problem referred to as *Simon's problem*. This algorithm is the key element of most of our quantum attacks against AES-OTR and OPP.

Definition 1. *(Simon's Problem) Given quantum access to a Boolean function* $f : \{0,1\}^n \to \{0,1\}^n$ *(called Simon's function) for which it holds:* $\exists s \in \{0,1\}^n : \forall x, y \in \{0,1\}^n$

$$f(x) = f(y) \iff y \in \{x, x \oplus s\},$$

the goal is to find the period s *of* f*.*

This problem of course can be solved in a classical setting by searching for collisions in $\Theta(2^{n/2})$, when we are given classical access to the function f. However, when we are able to query the function f quantum-mechanically, and we are thus allowed to make queries of arbitrary quantum superpositions of the form $|x\rangle|0\rangle \mapsto |x\rangle|f(x)\rangle$, Simon's algorithm can solve this problem with query complexity $\mathcal{O}(n)$. On a high level, Simon's algorithm is able to recover a random vector $y \in \{0,1\}^n$ in a *single* quantum query to f that is orthogonal to the period s, i.e. $y \cdot s = 0$. This subroutine is repeated $\mathcal{O}(n)$ times such that one obtains $n-1$ independent vectors where each is orthogonal to s with high probability. Therefore s can be recovered by solving the corresponding system of linear equations. For more details on the subroutine, we refer to [17].

Also [17] showed that Simon's algorithm recovers the hidden period s with $\mathcal{O}(n)$ quantum queries even if f has some "unwanted periods" – i.e., values $t \neq s$ such that $f(x) = f(x \oplus t)$ holds with probability $\leq 1/2$ over a random choice of x. As we will show, this condition is always satisfied in our attacks.

Deutsch's Algorithm. Deutsch's algorithm solves the following problem.

Definition 2. *Given quantum access to a Boolean function* $f : \{0,1\} \to \{0,1\}$*, the goal is to decide whether* f *is constant, i.e.* $f(0) = f(1)$*, or* f *is balanced, i.e.* $f(0) \neq f(1)$*.*

The algorithm can solve this problem with a *single* quantum query to f with success probability 1; note that any algorithm with classical access to f would need two queries for the same. To be precise, Deutsch's algorithm solves the above problem by computing the value $f(0) \oplus f(1)$ using a single quantum query to f.

IND-qCPA security of AEAD schemes. Below we define IND-qCPA security for nonce-based authenticated encryption with associated data (AEAD) schemes; the formal definitions for such nonce-based AEAD schemes are provided in Appendix A.

Definition 3. *(IND-qCPA with random nonces) A nonce-based AEAD scheme* Π = (*Enc, Dec*) *is indistinguishable under quantum chosen-plaintext attack (IND-qCPA secure) with random nonces, if there is no efficient quantum adversary* $\mathcal{A}$ *that is able to win the following security game, except with probability at most* $\frac{1}{2} + \epsilon$ *where* $\epsilon > 0$ *is negligible.*

Key generation: *A random key $K \leftarrow \mathcal{K}$ and a random bit $b \leftarrow \{0,1\}$ are chosen by the challenger.*

Queries: *In any order the adversary $\mathcal{A}$ is allowed to make two types of queries:*

- **Encryption queries:** *The challenger first randomly chooses a nonce $N \leftarrow \{0,1\}^\kappa$ and forwards it to $\mathcal{A}$. The adversary now can choose a message-AD pair (M, A), possibly in superposition, and the challenger encrypts (N, A, M) with the classical nonce N and returns the output (C, T) to $\mathcal{A}$.*
- **Challenge query:** *The challenger picks a random nonce $N \leftarrow \{0,1\}^\kappa$ once more and gives it to the adversary. Afterwards, $\mathcal{A}$ chooses two same sized classical message-AD pairs $(M_0, A), (M_1, A)$ and forwards them to the challenger which in turn encrypts (N, A, M_b) with the previously chosen classical nonce N. The output (C^*, T^*) is again given to $\mathcal{A}$.*

Guess: *The adversary outputs a bit b' and wins if $b = b'$.*

Let p be the probability that $\mathcal{A}$ wins the above game. Then its IND-qCPA advantage with respect to the AEAD scheme Π is given by $Adv_{\Pi}^{IND\text{-}qCPA}(\mathcal{A}) = \left|p - \frac{1}{2}\right|$. So Π is said to be IND-qCPA secure under randomly chosen nonces if $Adv_{\Pi}^{IND\text{-}qCPA}(\mathcal{A})$ of any polynomial-time quantum adversary $\mathcal{A}$ is negligible.

3 Quantum Attacks on Confidentiality of OTR

The *AES-OTR* block cipher mode emerged from the *Offset Two-Round* (OTR) mode [23] as a part of the CAESAR competition [1] and is based on the AES block cipher as proposed in [24]. It is a nonce based authenticated encryption with associated data (AEAD) scheme and provides two methods for associated data processing. AES-OTR has a provable security in the classical setting under the assumption that AES is a pseudorandom function as argued in [24].

However, in this section we will show that we can exploit the way AD is processed in both cases, namely in parallel and serial, to break IND-qCPA security. In this case, we assume a setting where the adversary has quantum access to an encryption oracle and the nonces the challenger uses to answer encryption queries are picked uniformly at random. In Appendix D, we even go one step further and break IND-qCPA security of AES-OTR considered as a pure AE scheme, i.e., with empty AD. To do so, we consider a stronger adversarial setting in which the adversary is allowed to pick the classical nonces adaptively. We will be extending upon techniques as utilized in [22].

3.1 Specifications of AES-OTR

We begin by describing the AES-OTR mode by following the specifications as proposed in [24] for the third round of the CAESAR competition. Let n, k, τ, κ as labeled in Section 2, where $k \in \{128, 192, 256\}$, $\tau \in \{32, 40, ..., 128\}$ and $\kappa \in \{8, 16, ..., 120\}$ are of a fixed length. Since AES-OTR uses AES as its underlying block cipher, $n = 128$ is fixed as well and we assume E_K to denote the AES

encryption function with key K. Also, the lengths of both a plaintext M and associated data A are required to fulfill $|M|, |A| \in \{0,1\}^{8*}$ such that $|M|_8, |A|_8 \leq 2^{64}$. We note that [24] provides sets of recommended parameters which imply that for both instantiations of AES-OTR either with AES-128 or AES-256 a 16-byte tag should be used. This recommendation becomes relevant for our attack in Section 3.2. For further details on the parameters we refer to [24].

Below, we provide a simplified description of AES-OTR for both variants of processing AD, namely in parallel (on the left) and in serial (on the right). To indicate how the AD is processed, we use p for parallel and s for serial processing and write OTR-$\mathcal{E}_{K,p}(N, A, M)$ or OTR-$\mathcal{E}_{K,s}(N, A, M)$ respectively. We omit the description of the decryption algorithm, as decryption is not relevant for our attacks. We again refer to [24] for the details. To be more precise, Algorithm 4 corresponds to the encryption core and Algorithms 5 and 6 describe the authentication core of the AEAD scheme described in Algorithm 1 and 2 for parallel and serial AD processing respectively. Note that the encryption core of AES-OTR with parallel (normal box) and serial (dashed box) AD processing only differ in the way U is defined. Algorithm 3 outlines how the nonce N is formatted before being incorporated into the mask U, used to encrypt the plaintext. (This formatting will play an important role for our attack in Section D with adaptive nonces.) Notice that for a single block message AES-OTR only encrypts it by xor-ing it with some value depending on U. We will exploit this property for our IND-qCPA attacks.

Algorithm 1 OTR-$\mathcal{E}_{K,p}(N, A, M)$

$(C, TE) \leftarrow$ EF-P$_{K,\tau}(N, M)$
if $A \neq \varepsilon$ **then**
  $TA \leftarrow$ AF-P$_K(A)$
else $TA \leftarrow 0^n$
$T \leftarrow \text{msb}_\tau(TE \oplus TA)$
return (C, T)

Algorithm 2 OTR-$\mathcal{E}_{K,s}(N, A, M)$

if $A \neq \varepsilon$ **then**
  $TA \leftarrow$ AF-S$_K(A)$
else $TA \leftarrow 0^n$
$(C, TE) \leftarrow$ EF-S$_{K,\tau}(N, M, TA)$
$T \leftarrow \text{msb}_\tau(TE)$
return (C, T)

Algorithm 3 Format(τ, N)

return $\text{bin}(\tau \bmod n, 7)||0^{n-8-\kappa}||1||N$

Algorithm 4 $\boxed{\text{EF-P}_{K,\tau}(N,M)}$, $\boxed{\text{EF-S}_{K,\tau}(N,M,TA)}$

$\Sigma \leftarrow 0^n$
$\boxed{U \leftarrow E_K(\text{Format}(\tau,N))}$
$\boxed{U \leftarrow 2\big(E_K(\text{Format}(\tau,N)) \oplus TA\big)}$
$L \leftarrow U$, $L^{\#} \leftarrow 3U$
$M_1||...||M_m \leftarrow M$ s.t. $|M_i| = n$
for $i \in \{1,...,\lceil m/2 \rceil - 1\}$ **do**
    $C_{2i-1} \leftarrow E_K(L \oplus M_{2i-1}) \oplus M_{2i}$
    $C_{2i} \leftarrow E_K(L^{\#} \oplus C_{2i-1}) \oplus M_{2i-1}$
    $\Sigma \leftarrow \Sigma \oplus M_{2i}$
    $L \leftarrow L \oplus L^{\#}$, $L^{\#} \leftarrow 2L^{\#}$
if m is even **then**
    $Z \leftarrow E_K(L \oplus M_{m-1})$
    $C_m \leftarrow \text{msb}_{|M_m|}(Z) \oplus M_m$
    $C_{m-1} \leftarrow E_K(L^{\#} \oplus \underline{C_m}) \oplus M_{m-1}$
    $\Sigma \leftarrow \Sigma \oplus Z \oplus \underline{C_m}$
    $L^* \leftarrow L^{\#}$
else if m is odd **then**
    $C_m \leftarrow \text{msb}_{|M_m|}(E_K(L)) \oplus M_m$
    $\Sigma \leftarrow \Sigma \oplus \underline{M_m}$
    $L^* \leftarrow L$
if $|M_m| \neq n$ **then**
    $TE \leftarrow E_K(3^2 L^* \oplus \Sigma)$
else $TE \leftarrow E_K(7L^* \oplus \Sigma)$
$C \leftarrow C_1||...||C_m$
return (C, TE)

Algorithm 5 $\text{AF-P}_K(A)$

$\Xi \leftarrow 0^n$
$Q \leftarrow E_K(0^n)$
$A_1||...||A_a \leftarrow A$, s.t. $|A_i| = n$
for $i \in \{1,...,a-1\}$ **do**
    $\Xi \leftarrow \Xi \oplus E_K(Q \oplus A_i)$
    $Q \leftarrow 2Q$
$\Xi \leftarrow \Xi \oplus \underline{A_a}$
if $|A_a| \neq n$ **then**
    $TA \leftarrow E_K(3Q \oplus \Xi)$
else $TA \leftarrow E_K(3^2 Q \oplus \Xi)$
return TA

Algorithm 6 $\text{AF-S}_K(A)$

$\Xi \leftarrow 0^n$
$Q \leftarrow E_K(0^n)$
$A_1||...||A_a \leftarrow A$, s.t. $|A_i| = n$
for $i \in \{1,...,a-1\}$ **do**
    $\Xi \leftarrow E_K(A_i \oplus \Xi)$
$\Xi \leftarrow \Xi \oplus \underline{A_a}$
if $|A_a| \neq n$ **then**
    $TA \leftarrow E_K(2Q \oplus \Xi)$
else
    $TA \leftarrow E_K(4Q \oplus \Xi)$
return TA

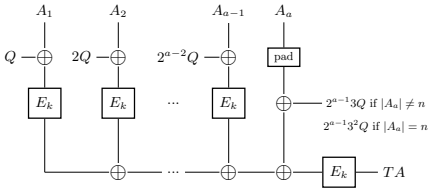

Fig. 1: Computation of the value TA of the authentication core of AES-OTR with parallel AD processing with $Q = E_K(0^n)$.

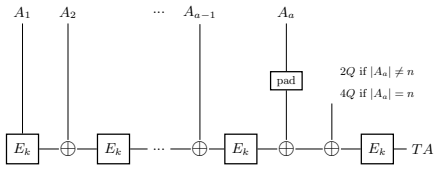

Fig. 2: Computation of the value TA of the authentication core of AES-OTR with serial AD processing with $Q = E_K(0^n)$.

In the pictorial description of how the value TA in Algorithms 5 and 6 for parallel and serial AD processing is computed, the value Q is used as part of the

masking and is defined as $Q = E_K(0^n)$. Note that Q is a constant value and is independent of the nonce N. This is the key observation we use in our attacks in Sections 3.2 and 3.3 that exploit the way AD is being processed.

It is worth mentioning that there are some prior works (specifically [17,20]) which proposed approaches to attack *unforgeability* of AES-OTR using Simon's algorithm when given quantum access to the corresponding encryption oracle. On a high level, the above works exploit the way AD is processed to compute collisions in the intermediate variable TA (see Figures 1 and 2); we present a detailed description of their attacks in Appendix B. We will use similar ideas for our following quantum attacks on the *confidentiality* of AES-OTR.

3.2 IND-qCPA Attack on AES-OTR with Parallel AD Processing

In this section, we show that AES-OTR is insecure in the IND-qCPA setting with random nonces when it is used as an AEAD scheme with associated data processed in parallel. In our IND-qCPA attack, we exploit the way AD is processed to find collisions for the output value TA of Algorithm 5, as well as exploit the fact that the encryption algorithm essentially performs a one-time pad encryption when given a single-block message.

As a general attack strategy, we want to create a periodic function f_1, whose period can be computed using Simon's algorithm. To construct our Simon's function f_1, we use a similar approach as described in [22, Section 4.3] for breaking IND-qCPA security of OCB2. Note that the way OCB2 authenticates AD in [22, Figure 3] is (up to multiplication with constants) essentially the same as for AES-OTR in Algorithm 5. Thus, we can follow a very similar argument.

We define $f_1 : \{0,1\}^n \to \{0,1\}^\tau$

$$f_1(A) = \text{OTR-}\mathcal{E}_{K,p}(N, A||A||0^n, \varepsilon).$$

As the plaintext is chosen to be empty, the ciphertext is empty as well which is the reason the quantum encryption oracle OTR-$\mathcal{E}_{K,p}(N,\cdot)$ returns a tag of length τ only.[1] Now notice that f_1 has period $s = 3Q = Q \oplus 2Q$ since

$$\begin{aligned}
f_1(A) &= \text{msb}_\tau(TA \oplus TE) \\
&= \text{msb}_\tau\Big(E_K\big(2^2 3^2 Q \oplus E_K(A \oplus Q) \oplus E_K(A \oplus 2Q)\big) \oplus TE\Big) \\
&= \text{msb}_\tau\Big(E_K\big(2^2 3^2 Q \oplus E_K(A \oplus Q \oplus 2Q \oplus 2Q) \oplus E_K(A \oplus 2Q \oplus Q \oplus Q)\big) \oplus TE\Big) \\
&= \text{msb}_\tau\Big(E_K\big(2^2 3^2 Q \oplus E_K(A \oplus 3Q \oplus 2Q) \oplus E_K(A \oplus 3Q \oplus Q)\big) \oplus TE\Big) \\
&= f_1(A \oplus s).
\end{aligned}$$

[1] We notice that in the context of Simon's algorithm the domain and the co-domain of Simon's function are required to be of the same dimension. Since the size of the tag τ is possibly less than n, we technically need to append an additional $n - \tau$ bits of zeros (or any other fixed bit string of size $n - \tau$). Importantly, this does not change the periodicity of f_1, as these bits are fixed. For the sake of convenience however, we refrain from appending them in each step of the analysis of f_1.

Following the arguments made in [22, Section 3.1], using a single quantum query to the encryption oracle OTR-$\mathcal{E}_{K,p}(N,\cdot)$, the function f_1 can be computed in superposition. We need this to be done in a single quantum query, as the nonce N changes with each quantum query made to the oracle. We thus can apply Simon's algorithm to f_1, which computes a vector $y \in \{0,1\}^n$ that is orthogonal to the period $s = 3Q = 3E_K(0^n)$. Notice that there do not exist any "unwanted periods" as defined in Section 2 with overwhelming probability, since we can apply the same reasoning as in [17, Section 5.3] using that AES is a PRP.

It is important that the period is independent of the nonce N, such that despite the nonce changing with each quantum query, Simon's algorithm still returns a random vector y orthogonal to the fixed period. This means, after recovering $O(n)$ such independent orthogonal vectors y, we can recover the value $3Q = 3E_K(0^n)$ and thus the value $E_K(0^n)$ as well with $O(n)$ quantum queries to the AES-OTR encryption oracle.

For the rest of this section, we assume that AES-OTR is instantiated using the recommended parameter sets from [24]. In particular, we assume that the size of the tag is 16-bytes, i.e., $\tau = n$. It turns out that not only is this assumption necessary for our attack to succeed efficiently but is also necessary for the IND-qCPA attack on OCB2 in [22, Section 4.3]. We discuss this assumption in detail in Appendix C.

As a next step towards breaking IND-qCPA security we want to gain raw block cipher access i.e. the ability to compute $E_K(inp)$ for any given input $inp \in \{0,1\}^n$. To realize this, we use Deutsch's algorithm like done in [22] as follows. Having recovered the value $Q = E_K(0^n)$, define two fixed single-block associated data inputs $\alpha_0 = 3^2Q$ and $\alpha_1 = 3^2Q \oplus inp$ for any given input $inp \in \{0,1\}^n$. We continue by considering the n functions $f^{(i)} : \{0,1\} \to \{0,1\}$,

$$f^{(i)}(b) = i\text{th bit of } \{\text{OTR-}\mathcal{E}_{K,p}(N, \alpha_b, \varepsilon)\} \tag{3.1}$$

for a random nonce N and empty message. Here again, the output of OTR-$\mathcal{E}_{K,p}(N,\cdot)$ is only the tag of length $\tau = n$, as the message is kept empty. Following the argument in [22], we can compute $f^{(i)}(b)$ in superposition with a single quantum query to the AES-OTR encryption oracle by also truncating out the unneeded $n-1$ bits of the output of OTR-$\mathcal{E}_{K,p}(N,\cdot)$. This gives us the ability to apply Deutsch's algorithm on $f^{(i)}$ and recover the value

$$\begin{aligned} f^{(i)}(0) \oplus f^{(i)}(1) &= i\text{th bit of}\{TE \oplus E_K(\alpha_0 \oplus 3^2Q)\} \\ &\quad \oplus i\text{th bit of}\{TE \oplus E_K(\alpha_1 \oplus 3^2Q)\} \\ &= i\text{th bit of}\{E_K(0^n) \oplus E_K(inp)\} \end{aligned} \tag{3.2}$$

with a single quantum query. Thus, by applying Deutsch's algorithm to each of the n functions $f^{(i)}$, we are able to recover all n bits of $\big(E_K(0^n) \oplus E_K(inp)\big)$ and from this, since we already know $E_K(0^n)$, we can recover $E_K(inp)$. It is worth pointing out, that despite the nonce N changes with each application of Deutsch's algorithm, we are still able to recover $\big(E_K(0^n) \oplus E_K(inp)\big)$ since it is independent of N; note that the nonce-dependent value TE gets "xored-out" in Equation (3.2).

As a result of the observations above, we can now sketch our IND-qCPA attack against AES-OTR with parallel AD processing:

1. Recover the value $3Q$ and thus the value $E_K(0^n)$ using $O(n)$ quantum queries with Simon's algorithm as discussed above.
2. Pick arbitrary but different single-block messages M_0 and M_1 and define the associated data to be empty, i.e. $A = \varepsilon$. We now give these values as an input to the challenger and record the nonce N which is used to encrypt M_b as well as the output (C^*, T) of the challenger.
3. Using Deutsch's algorithm $2n$ times (i.e., using raw block cipher access twice) as described above, we compute the value $V = E_K\Big(E_K\big(\text{Format}(\tau, N)\big)\Big)$ using a total of $2n$ quantum queries. Here, we use the nonce N the challenger used to encrypt the challenge query.
4. Output the bit $b'' = b'$ if $C^* = V \oplus M_{b'}$.

It remains to show that our attack outputs the correct bit b'':

To see this, we notice that for a single-block message M and empty AD the output of the encryption oracle is the ciphertext-tag pair (C^*, T) where

$$C^* = \text{msb}_{|M|}\big(E_K(L)\big) \oplus M$$

with $L = E_K(\text{Format}(\tau, N))$ following the description of Algorithm 4. Since the attack relies on recomputing exactly the value used to encrypt M_b in a one-time pad-like manner, the attack succeeds with high probability.

3.3 IND-qCPA Attack on AES-OTR with Serial AD Processing

The aim of this section is to break IND-qCPA security of AES-OTR used as an AEAD scheme with random nonces but now with associated data processed in serial by Algorithm 6. We again exploit the way AD is processed to compute collisions for the variable TA.

Similar to the attack above, we define a periodic function f_2 whose period can be computed using Simon's algorithm. In this case however, as a consequence of our choice of f_2, computing the period of the function already gives us raw block cipher access. This is in contrast to the IND-qCPA attack in [22, Section 4.3] and our previous attack in Section 3.2, where we first recover $E_K(0^n)$ and then gain raw block cipher access via Deutsch's algorithm. In this section, Simon's function is defined in a very different fashion than before, as the function can distinguish two different cases depending on an input bit b and treat them accordingly – this affects either having one or two blocks of associated data as an input to the quantum encryption oracle respectively. We therefore also need to argue why we actually have quantum access to Simon's function we define below.

This is precisely the scenario we described in Section 1.1 w.r.t. an adversary being able to query a superposition of (associated) data with *unequal* length to the quantum encryption oracle. We achieve this by a novel modelling of the encryption oracle in our quantum circuit for f_2, such that the circuit also queries

the encryption oracle only *once* each time f_2 is queried; we emphasize the latter aspect is due to the changing nonces as described in Section 3.2. At this point however, we make the assumption that we have quantum access to the Simon's function f_2 and are able to compute it with a single query to the quantum encryption oracle. We later validate this assumption and further discuss the quantum accessibility as well as the issue of having to achieve this with a single quantum query near the end of this section.

We realize gaining raw block cipher access by choosing an arbitrary $B \in \{0,1\}^n$ (for which we want to know its encryption $E_K(B)$) and set Simon's function to be $f_2 : \{0,1\}^{n+1} \to \{0,1\}^\tau$,

$$f_2(b||A) = \begin{cases} \text{OTR-}\mathcal{E}_{K,s}(N, B||A, \varepsilon) & \text{if } b = 0 \\ \text{OTR-}\mathcal{E}_{K,s}(N, A, \varepsilon) & \text{if } b = 1 \end{cases}$$

where $b \in \{0,1\}$ is a single bit and $A \in \{0,1\}^n$ represents one block of associated data of size n. Note that here f_2 gives us a value in $\{0,1\}^\tau$, since the ciphertext is empty as a result of the plaintext being chosen as empty. More precisely, for a general set of associated data D and empty plaintext we get by Algorithm 4 and Algorithm 2 that

$$\text{OTR-}\mathcal{E}_{K,s}(N, D, \varepsilon) = \text{msb}_\tau\Big(E_K\Big(3^3 2\Big(TA_D \oplus E_K\big(\text{Format}(\tau, N)\big)\Big)\Big)\Big)$$

where $TA_D = \text{AF-S}_K(D)$. This implies that the period s of f_2 only depends on the function $\text{AF-S}_K(D)$. Therefore, we define a new function $g : \{0,1\}^{n+1} \to \{0,1\}^n$,

$$g(b||A) = \begin{cases} \text{AF-S}_K(B||A) & \text{if } b = 0 \\ \text{AF-S}_K(A) & \text{if } b = 1 \end{cases}$$

We claim that g, and therefore f_2 as well, has period $s = 1||E_K(B)$. Indeed:

$$\begin{aligned} g\big(0||A \oplus 1||E_K(B)\big) &= g\big(1||A \oplus E_K(B)\big) = \text{AF-S}_K\big(A \oplus E_K(B)\big) \\ &= E_K(4Q \oplus A \oplus E_K(B)) = \text{AF-S}_K(B||A) = g(0||A) \quad (3.3) \\ g(1||A \oplus 1||E_K(B)) &= g(0||A \oplus E_K(B)) = \text{AF-S}_K(B||A \oplus E_K(B)) \\ &= E_K\Big(4Q \oplus A \oplus E_K(B) \oplus E_K(B)\Big) = E_K(4Q \oplus A) \\ &= \text{AF-S}_K(A) = g(1||A) \quad (3.4) \end{aligned}$$

where $Q = E_K(0^n)$, and for Equations 3.3 and 3.4, we used the definition of Algorithm 6.

Under the assumption that we can in fact compute f_2 in superposition using a *single* quantum query to the encryption oracle $\text{OTR-}\mathcal{E}_{K,s}(N, \cdot)$, we can apply Simon's algorithm to f_2. Hence, with a similar argument as in Section 3.2 we can recover the value $s = 1||E_K(B)$ and thus the value $E_K(B)$ for any $B \in \{0,1\}^n$ with $O(n)$ quantum queries. It is important to mention that the period s is again

independent of the nonce. So even though the nonce, and hence f_2 changes with each quantum query, Simon's algorithm still returns a random vector orthogonal to the *fixed* period $s = 1||E_K(B)$ in each of the n iterations. We conclude that this grants us raw block cipher access without having to use Deutsch's algorithm like in the attack described in Section 3.2.

Unlike our IND-qCPA attack in Section 3.2, this attack succeeds with high probability even if the tags were truncated. This is justified because in the previous section truncation only became an issue when we applied Deutsch's algorithm. Here, we do not use Deutsch's algorithm but Simon's algorithm only. Since the function f_2 is still periodic as its periodicity is unaffected by the truncation of the tag, running Simon's algorithm does not run into any issues.

Now we can again sketch an IND-qCPA attack against AES-OTR but this time with serial AD processing:

1. Pick arbitrary but different single-block messages M_0 and M_1 and define the AD to be empty. We give these values as an input to the challenger and record the nonce N which was used to encrypt either M_0 or M_1 as well as the output (C^*, T) of the challenger.
2. Compute the value $V = E_K\Big(2 \cdot E_K\big(\text{Format}(\tau, N)\big)\Big)$ using $2\mathcal{O}(n)$ quantum encryption queries via two applications of Simon's algorithm $\big($using the raw block cipher access twice as discussed above$\big)$.
3. Output the bit $b'' = b'$ if $M_{b'} = C^* \oplus V$.

To see that the above attack succeeds we note that for a single-block message M of size n and empty AD we have

$$\text{OTR-}\mathcal{E}_{K,s}(N, \varepsilon, M)\Big|_C = E_K\Big(2 \cdot E_K\big(\text{Format}(\tau, N)\big)\Big) \oplus M.$$

where by $|_C$ we indicate truncating out the tag T. This is essentially a one-time pad encryption with mask V. Since we are able to compute V, we also recover the correct bit b''.

On the Quantum Accessibility of Function f_2. It remains to argue that we actually have quantum access to the function f_2 with a *single* query to the encryption oracle each time we query f_2. This is done by coming up with a suitable quantum circuit that describes f_2 such that it uses only a single encryption unitary gate; this is because the nonce N changes with each quantum query made to the encryption oracle.

To achieve this, we have to modify the quantum encryption oracle queries in a slight way: we add an additional n-qubit input register which encodes the length of our message. By doing so, the encryption oracle knows how many bits of the message it should actually encrypt. Thus, we also need to define f_2 in a different manner:

$$f_2(b||A) = \begin{cases} \text{OTR-}\tilde{\mathcal{E}}_{K,s}(N, \text{bin}(2n, n)||B||A, \varepsilon) & \text{if } b = 0. \\ \text{OTR-}\tilde{\mathcal{E}}_{K,s}(N, \text{bin}(n, n)||A||0^n, \varepsilon) & \text{if } b = 1. \end{cases}$$

The circuit also uses so-called Fredkin gate (as described e.g. in [30, Section 2.2]), which is a controlled swap gate. The Fredkin gate, upon input basis state (b, I_1, I_2) with b a single bit, produces output (b, O_1, O_2) where $O_1 = \bar{b}I_1 + bI_2$, $O_2 = bI_1 + \bar{b}I_2$. So the gate essentially swaps the inputs when $b = 1$ and does nothing if $b = 0$. Note that the way f_2 is defined, both $|\text{bin}(2n, n)||B||A\rangle$ and $|\text{bin}(n, n)||A||0^n\rangle$ are $3n$ qubit states and hence can be swapped (the swap gate can only operate on inputs that are of the same length, as the swapping is done qubit wise). In this case, when the encryption oracle gets such an input, it first parses the first n qubits of the query to figure out how many blocks of the input have to be encrypted. For $b = 0$ it just takes $B||A$ as an input but if $b = 1$ then it ignores the remaining 0^n block of the query and only encrypts A. The corresponding quantum circuit therefore looks as follows:

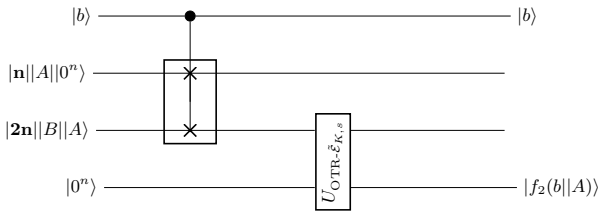

Fig. 3: The quantum circuit implementing f_2 using a single quantum query via the gate $U_{\text{OTR-}\tilde{\mathcal{E}}_{K,s}}$. We indicate the n-bit encoding of n and $2n$ with bold letters.

Note that the output of the second register is set to $|\mathbf{n}||A||0^n\rangle$ if $b = 0$ and $|\mathbf{2n}||B||A\rangle$ if $b = 1$. For the third register it is the other way round.

Comparison with the Quantum Attack on CMAC in [17]. Note that the serial AD processing component of AES-OTR (Algorithm 6) is essentially equivalent to the message authentication code CMAC [9]. And Kaplan *et al.* [17] show how to break the quantum unforgeability of CMAC using Simon's algorithm (technically they present a quantum attack against the closely related CBC-MAC, but it is straightforward to extend the attack to CMAC).

On a high-level, the extension of their CMAC attack to AES-OTR with serial AD processing would proceed as follows. Choose two arbitrary blocks $B_0, B_1 \in \{0,1\}^n$ and define the function $\tilde{g}(b||A) = \text{AF-S}_K(B_b||A)$ (corresponding to Simon's function $\tilde{f}_2(b||A) = \text{OTR-}\mathcal{E}_{K,s}(N, B_b||A, \varepsilon)$). It is not hard to see that $\tilde{g}$ (and $\tilde{f}_2$) has period $\tilde{s} = 1||(E_K(B_0) \oplus E_K(B_1))$. Hence we can use Simon's algorithm as in [17, Section 5.1] to recover the value $E_K(B_0) \oplus E_K(B_1)$. Also note that we no longer need to query the quantum encryption oracle on a superposition of unequal length AD, unlike our IND-qCPA attack above. However, just knowing $E_K(B_0) \oplus E_K(B_1)$ is not sufficient to obtain the raw block cipher access $E_K(\cdot)$ we would need to break IND-qCPA security, even if B_0, B_1 are under our control.

4 Quantum Key-Recovery Attack on OPP

The *Offset Public Permutation Mode* (OPP) was proposed in [10] and [11] and it essentially tries to generalize OCB3 by replacing the underlying block cipher by a public permutation and a different form of masking. In this section we will show that using a public permutation the way OPP does, actually leads to a devastating key recovery attack in the quantum setting.

Our attack uses a similar strategy as the IND-qCPA attack against OCB2 in [22, Section 4.4] with adaptively chosen nonces; but instead of choosing a new nonce adaptively to break IND-qCPA security, we are able to recover the value $\Omega := P(X||K)$ where P is an efficiently invertible public permutation and K is the key (X can be seen as a formatting of the nonce). In contrast to OTR being based on a block cipher that may only be inverted knowing the key, we are here dealing with a public permutation that can be inverted efficiently. This is the key issue of OPP and the reason we are able to recover the key knowing Ω.

4.1 Specification of OPP

In this section, it is assumed that the plaintexts we are dealing with always have a size that is a multiple of the block length. As a consequence, it is not necessary to treat the last block of the plaintext any differently in our analysis, and furthermore we are also excluding the specifications for how OPP encrypts plaintexts that do not meet this assumption. In the same manner this also applies to the way we describe the processing of associated data.

Let n, k, τ, κ as labeled in Section 2 such that $\kappa \leq n - k - 1$. We begin by describing the OPP mode as proposed in [10] in a simplified manner that also maintains consistent labeling of variables in previous descriptions of modes such as OTR.

A set of functions $\Phi = \{\alpha, \beta, \gamma\}$ is given by $\alpha, \beta, \gamma : \{0,1\}^n \to \{0,1\}^n$, $\alpha(x) = \varphi(x)$, $\beta(x) = \varphi(x) \oplus x$ and $\gamma(x) = \varphi^2(x) \oplus \varphi(x) \oplus x$ where for $x = x_0||...||x_{15}$ and $x_i \in \{0,1\}^{64}$ the function $\varphi : \{0,1\}^{1024} \to \{0,1\}^{1024}$ is defined as

$$\varphi(x_0, ..., x_{15}) = \big(x_1, ..., x_{15}, (x_0 \lll 53) \oplus (x_5 \ll 13)\big).$$

OPP uses the so called tweakable Even-Mansour construction MEM, where a tweak space $\mathcal{T}$ of the form $\mathcal{T} \subseteq \{0,1\}^{n-k} \times \mathbb{N}^3$, as outlined in [10, Lemma 4], is considered. For further details about tweaks and tweakable block ciphers we refer to [10] as this specific notion is not relevant for the subsequent attack.

The encryption function $\widetilde{E} : \{0,1\}^k \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ is then defined as

$$\widetilde{E}(K, X, \bar{i}, M) = P\big(\delta(K, X, \bar{i}) \oplus M\big) \oplus \delta(K, X, \bar{i})$$

where $\delta : \{0,1\}^k \times \mathcal{T} \to \{0,1\}^n$ is called the masking function and for $\bar{i} = (i_0, i_1, i_2) \in \mathbb{N}^3$ it is set to be $\delta(K, X, \bar{i}) = \gamma^{i_2} \circ \beta^{i_1} \circ \alpha^{i_0}\big(P(X||K)\big)$. For convenience the shorthand notation $\widetilde{E}^{\bar{i}}_{K,X}(M) = \widetilde{E}(K, X, \bar{i}, M)$ is being used in the algorithmic description of OPP.

The decryption function $\widetilde{D}$ corresponding to $\widetilde{E}$ is defined in a straightforward manner. However it should be noted that P is a public permutation such that P^{-1} can be computed efficiently, since the inverse permutation is necessary to perform decryption.

Below, we present a simplified description of the OPP mode based on the specification in [10]. We only provide a description of the encryption and authentication part of the algorithm, as the details regarding decryption are not relevant to our attack and are therefore omitted. To be precise, the authentication core of OPP is described in Algorithm 9 and the encryption core corresponds to Algorithm 8.

Algorithm 7 OPP-$\mathcal{E}(K, N, AD, M)$

1: $X \leftarrow \text{pad}^0_{n-\kappa-k}(N)$
2: $C, S \leftarrow \text{OPPEnc}(K, X, M)$
3: $T \leftarrow \text{OPPAbs}(K, X, AD, S)$
4: **return** C, T

Algorithm 8 OPPEnc(K, X, M)

1: $M_0||...||M_{m-1} \leftarrow M$, s.t. $|M_i| = n$
2: $C \leftarrow \varepsilon$
3: $S \leftarrow 0^n$
4: **for** $i \in \{0, ..., m-1\}$ **do**
5: $\quad C_i \leftarrow \widetilde{E}^{i,0,1}_{K,X}(M_i)$
6: $\quad C \leftarrow C||C_i$
7: $\quad S \leftarrow S \oplus M_i$
8: **return** $C, \widetilde{E}^{m-1,2,1}_{K,X}(S)$

Algorithm 9 OPPAbs(K, X, A, S)

1: $A_0||...||A_{a-1} \leftarrow A$, s.t. $|A_i| = n$
2: $S' \leftarrow 0^n$
3: **for** $i \in \{0, ..., a-1\}$ **do**
4: $\quad S' \leftarrow S' \oplus \widetilde{E}^{i,0,0}_{K,X}(A_i)$
5: **return** $\text{msb}_\tau(S' \oplus S)$

4.2 Our Quantum Key-Recovery Attack

In this attack, we adapt the techniques used in [22, Section 4.4] for breaking IND-qCPA security of OCB2 with adaptively chosen nonces. However, in this case we are able to recover the key instead. Our attack is focused solely on the encryption part, so OPP is used as a pure AE scheme, and does not make use of the way associated data is processed. This is in contrast to our previous attacks in Sections 3.2 and 3.3 as there we could exploit the fact that AD processing was not dependent on a nonce but rather on the constant value $Q = E_k(0^n)$. In the case of OPP this is different because the nonce is used in the associated data processing, as the value $P(X||K)$ where $X = \text{pad}^0_{n-\kappa-k}(N)$ is dependent on the nonce N (see Algorithm 7). Since the nonce changes with each call to the encryption oracle, OPP never processes a fixed set of AD the same way. This is the reason we can't apply Simon's algorithm (which calls the oracle multiple times) in the same manner.

Before we formulate the attack itself, we observe a crucial property of the xor of two consecutive ciphertext blocks considered as a function of its corresponding plaintext blocks. Define $\Omega = P(X||K)$ and recall that OPP encrypts the i-th

plaintext block M_i as $C_i = P(\delta(K, X, (i, 0, 1)) \oplus M_i) \oplus \delta(K, X, (i, 0, 1))$ where $\delta(K, X, (i, 0, 1)) = \varphi^{i+2}(\Omega) \oplus \varphi^{i+1}(\Omega) \oplus \varphi^i(\Omega)$. We now define functions $f_i : \{0,1\}^n \to \{0,1\}^n$ such that

$$f_i(M) = P\big(\delta(K, X, (i, 0, 1)) \oplus M\big) \oplus \delta(K, X, (i, 0, 1)),$$

which correspond to the i-th ciphertext block C_i considered as a function of its underlying plaintext block M. Further, by defining $s := \varphi^{i+3}(\Omega) \oplus \varphi^i(\Omega)$ we see that

$$\begin{aligned}
&f_i(M \oplus s) \oplus f_{i+1}(M \oplus s) \\
&\quad = P\Big(\varphi^{i+2}(\Omega) \oplus \varphi^{i+1}(\Omega) \oplus \varphi^i(\Omega) \oplus M \oplus s\Big) \\
&\qquad \oplus P\Big(\varphi^{i+3}(\Omega) \oplus \varphi^{i+2}(\Omega) \oplus \varphi^{i+1}(\Omega) \oplus M \oplus s\Big) \oplus \varphi^{i+3}(\Omega) \oplus \varphi^i(\Omega) \\
&\quad = P\Big(\varphi^{i+3}(\Omega) \oplus \varphi^{i+2}(\Omega) \oplus \varphi^{i+1}(\Omega) \oplus M\Big) \\
&\qquad \oplus P\Big(\varphi^{i+2}(\Omega) \oplus \varphi^{i+1}(\Omega) \oplus \varphi^i(\Omega) \oplus M\Big) \oplus \varphi^{i+3}(\Omega) \oplus \varphi^i(\Omega) \\
&\quad = f_{i+1}(M) \oplus f_i(M)
\end{aligned}$$

So if we define $F_{i,i+1} : \{0,1\}^n \to \{0,1\}^n$ as $F_{i,i+1}(M) = f_i(M) \oplus f_{i+1}(M)$ we see from the above calculations that $F_{i,i+1}(M \oplus s) = F_{i,i+1}(M)$, i.e., $F_{i,i+1}$ is a periodic function with period $s = \varphi^{i+3}(\Omega) \oplus \varphi^i(\Omega)$.

The idea is now to apply a linear function to $2n + 1$ ciphertext blocks to capture this observation and create a periodic function that itself contains n copies of the periodic function $F_{i,i+1}$ from above. To do so consider the function $g : \{0,1\}^{(2n+1)n+\tau} \to \{0,1\}^{(n+1)n}$

$$g(C_0, C_1, ..., C_{2n}, t) = (C_0, C_1 \oplus C_2, ..., C_{2n-1} \oplus C_{2n}).$$

Here, the C_i's are n-bit blocks and t is a τ-bit block. It is not hard to see that g is in fact a linear function - i.e., it satisfies $g(C \oplus C') = g(C) \oplus g(C')$ for any valid inputs C and C'. Furthermore let $\tilde{f}_N : \{0,1\}^{n^2} \to \{0,1\}^{(n+1)n}$ such that

$$\tilde{f}_N(M_1, ..., M_n) = g \circ \text{OPP-}\mathcal{E}(K, N, \varepsilon, 0^n||M_1||M_1||M_2||...||M_n||M_n) \tag{4.1}$$

for some randomly chosen nonce N and empty associated data. We can also reformulate $\tilde{f}_N$ in terms of the functions f_i and $F_{i,i+1}$ from above. To be precise, it holds

$$\tilde{f}_N(M_1, ..., M_n) = \big(f_0(0^n), F_{1,2}(M_1), ..., F_{2n-1,2n}(M_n)\big). \tag{4.2}$$

Moreover, we have included an all-zero plaintext block at the beginning, which will be useful later on for verifying correctness of the recovered key.

The crucial property required for the success of the attack is that $\tilde{f}_N$ has n linearly independent periods $\langle s_i \rangle_{i \in [n]}$ where

$$s_i = \Big((0^n)^{i-1}||\varphi^{2i+2}(\Omega) \oplus \varphi^{2i-1}(\Omega)||(0^n)^{n-i}\Big).$$

This directly follows from the observation on the periodicity of $F_{i,i+1}$ and the fact that each pair of the two consecutive ciphertext blocks C_{2i-1} and C_{2i} for $i \in [n]$ encrypt the same plaintext block M_i but with different mask δ.

Following the same argument as in [22, Section 4.4], we can apply [4, Lemma 2] which assures the ability to compute a linear function of a quantum oracle's output. Therefore, we can compute $\tilde{f}_N$ with a single quantum query to the OPP-$\mathcal{E}(K, N, \cdot)$ oracle. Once more, we can apply Simon's algorithm to $\tilde{f}_N$ which, with a single quantum query, recovers a vector $y = (y_1, ..., y_n) \in \{0,1\}^{n^2}$ with $y_i \in \{0,1\}^n \ \forall i \in [n]$ that is orthogonal to each of the periods s_i. The algorithm successfully computes such a vector with overwhelming probability as there do not exist any "unwanted periods" to which y could be orthogonal to.

We justify this claim by building upon the argument presented in [4, Section 3.2], which treats the absence of "unwanted periods" in a very similar attack on a variant of OCB. We recall that OCB uses a block cipher instead of a public permutation like it is the case for OPP, so we need to adjust the reasoning to the setting of a public permutation. If we assume the existence of an unwanted period $\tilde{s}$ of $\tilde{f}_N$ with a probability greater than $\frac{1}{2}$, then at least one of the $F_{i,i+1}$ in Equation 4.2 would also have to admit an unwanted period $\tilde{s}_{i,i+1}$ with probability greater than $\frac{1}{2n}$. We now draw upon the reasoning presented in [17, Section 3.2], which shows the non-occurrence of higher order differentials in the Even-Mansour construction. More accurately, $F_{i,i+1}$ admitting such an unwanted period is equivalent to saying that P admits a high-probability higher-order differential. But these only happen with negligible probability for a random choice of P according to [17]. This argument makes sure that the probability for unwanted periods to appear is bounded and thus Simon's algorithm computes a vector y as described above with overwhelming probability.

By orthogonality of y we get n equations of the form

$$y_i \cdot \left(\varphi^{2i+2}(\Omega) \oplus \varphi^{2i-1}(\Omega)\right) = 0. \tag{4.3}$$

Before we can proceed, we recall that for $x = x_0||...||x_{15}$ and $x_i \in \{0,1\}^{64}$ the function φ is defined as

$$\varphi(x_0, ..., x_{15}) = \left(x_1, ..., x_{15}, (x_0 \lll 53) \oplus (x_5 \ll 13)\right)$$

As described in [10] we see that the function φ is in fact a linear map and it therefore can be represented by a matrix M. Following this, Equation 4.3 is equivalent to

$$y_i \cdot \left(M^{2i+2} \cdot \Omega \oplus M^{2i-1} \cdot \Omega\right) = 0 \tag{4.4}$$

Knowing M and using associativity of matrix multiplication, we are able to solve the n Equations in 4.4 and thus recover the value $\Omega = P(X||K)$. But since P is a public permutation and its inverse is assumed to be computable efficiently due P^{-1} being needed for decryption, we can just apply P^{-1} to Ω and we thus are able to acquire $X||K$. In particular, we gain possession of the key K.

Observe, that in addition when running Simon's algorithm, we recover the fixed classical value of the first ciphertext block $C_0 = \widetilde{E}^{0,0,1}_{K,X}(0^n)$ when we measure the quantum register corresponding to the output of $\tilde{f}_N$ as part of our

application of Simon's algorithm. It is important that we fix the value to 0^n (or any other arbitrary fixed classical value of length n also works) in order for C_0 to be a classical value.

We sketch our key-recovery attack below:

1. Given access to a quantum encryption oracle of OPP for a random nonce N, i.e., OPP-$\mathcal{E}(K, N, \varepsilon, \cdot)$, we recover with a single quantum encryption query the classical values $\Omega = P(X||K)$ and $C_0 = \widetilde{E}^{0,0,1}_{K,X}(0^n)$ as discussed above. In particular, we recover the classical value of the key K.
2. We perform a sanity check on the key: using Ω we recompute the encryption $\widetilde{C}$ of the one-block plaintext 0^n with respect to the same key K and nonce N as used in the above encryption oracle query as

$$\widetilde{C} = P\big(\varphi^2(\Omega) \oplus \varphi(\Omega) \oplus \Omega\big) \oplus \varphi^2(\Omega) \oplus \varphi(\Omega) \oplus \Omega.$$

   Check that $C_0 = \widetilde{C}$. If this turns out to be false we repeat step 1., else we are certain to have recovered the right key K.

It remains to argue why this attack is successful. We perform a sanity check in order to assure correctness of the key K. This is where the first ciphertext block is useful. Indeed, having recovered the key K as described above, we can now just recompute the encryption of 0^n, again with respect to the same key-nonce pair (K, N) as in the provided encryption oracle, as

$$\begin{aligned}\widetilde{C} &= P\big(\delta(K, X, (0,0,1)) \oplus 0^n\big) \oplus \delta(K, X, (0,0,1)) \\ &= P\big(\varphi^2(\Omega) \oplus \varphi(\Omega) \oplus \Omega\big) \oplus \varphi^2(\Omega) \oplus \varphi(\Omega) \oplus \Omega\end{aligned}$$

where $\Omega = P(X||K)$ and $X = \mathrm{pad}^0_{n-\kappa-k}(N)$. If now $\widetilde{C} = C_0$, i.e. the encryption of the oracle and our manual computation coincide, we can be sure that we recovered the right key K. Else, we can just repeat the attack until the assertion returns to be true.

With the included sanity check, we are certain to recover the key K at some point, as step one of our attack involves an application of Simon's algorithm that already succeeds with high probability thanks to the non-existence of "unwanted periods" as argued before. Finally, we compare our quantum key-recovery attack on OPP with the corresponding attack on the generic Even-Mansour construction of [17] in Appendix E.

Acknowledgements. It is our pleasure to thank Xavier Bonnetain for helpful discussions, and the anonymous reviewers of SAC 2023 for their constructive comments and suggestions.

References

1. Caesar: Competition for authenticated encryption: Security, applicability, and robustness, 2012-2019. Last accessed 23 March 2023, `https://competitions.cr.yp.to/caesar.html`.

2. G. Alagic, C. Bai, J. Katz, and C. Majenz. Post-quantum security of the even-mansour cipher. In *Advances in Cryptology – EUROCRYPT 2022, Part III*, pages 458–487, 2022.
3. G. Alagic, D. Cooper, Q. Dang, T. Dang, J. M. Kelsey, J. Lichtinger, Y.-K. Liu, C. A. Miller, D. Moody, R. Peralta, R. Perlner, A. Robinson, D. Smith-Tone, and D. Apon. Status report on the third round of the nist post-quantum cryptography standardization process, 2022.
4. R. Bhaumik, X. Bonnetain, A. Chailloux, G. Leurent, M. Naya-Plasencia, A. Schrottenloher, and Y. Seurin. QCB: Efficient quantum-secure authenticated encryption. In *Advances in Cryptology – ASIACRYPT 2021, Part I*, pages 668–698, 2021.
5. D. Boneh and M. Zhandry. Secure signatures and chosen ciphertext security in a quantum computing world. In *Advances in Cryptology – CRYPTO 2013, Part II*, pages 361–379, 2013.
6. X. Bonnetain, A. Hosoyamada, M. Naya-Plasencia, Y. Sasaki, and A. Schrottenloher. Quantum attacks without superposition queries: The offline Simon's algorithm. In *Advances in Cryptology – ASIACRYPT 2019, Part I*, pages 552–583, 2019.
7. X. Bonnetain, G. Leurent, M. Naya-Plasencia, and A. Schrottenloher. Quantum linearization attacks. In *Advances in Cryptology – ASIACRYPT 2021, Part I*, pages 422–452, 2021.
8. D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London Series A*, 400(1818):97–117, 1985.
9. M. Dworkin. Recommendation for block cipher modes of operation: The cmac mode for authentication. Technical Report NIST Special Publication (SP) 800-38B, National Institute of Standards and Technology, Gaithersburg, MD, 2005.
10. R. Granger, P. Jovanovic, B. Mennink, and S. Neves. Improved masking for tweakable blockciphers with applications to authenticated encryption. Cryptology ePrint Archive, Paper 2015/999, 2015. `https://eprint.iacr.org/2015/999`.
11. R. Granger, P. Jovanovic, B. Mennink, and S. Neves. Improved masking for tweakable blockciphers with applications to authenticated encryption. In *Advances in Cryptology – EUROCRYPT 2016, Part I*, pages 263–293, 2016.
12. L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, page 212–219, 1996.
13. A. Inoue, T. Iwata, K. Minematsu, and B. Poettering. Cryptanalysis of OCB2: Attacks on authenticity and confidentiality. In *Advances in Cryptology – CRYPTO 2019, Part I*, pages 3–31, 2019.
14. A. Inoue and K. Minematsu. Cryptanalysis of OCB2. Cryptology ePrint Archive, Report 2018/1040, 2018. `https://eprint.iacr.org/2018/1040`.
15. T. Iwata. Plaintext recovery attack of OCB2. Cryptology ePrint Archive, Report 2018/1090, 2018. `https://eprint.iacr.org/2018/1090`.
16. J. Jaeger, F. Song, and S. Tessaro. Quantum key-length extension. In *TCC 2021: 19th Theory of Cryptography Conference, Part I*, pages 209–239, 2021.
17. M. Kaplan, G. Leurent, A. Leverrier, and M. Naya-Plasencia. Breaking symmetric cryptosystems using quantum period finding. In *Advances in Cryptology – CRYPTO 2016, Part II*, pages 207–237, 2016.
18. T. Krovetz and P. Rogaway. The software performance of authenticated-encryption modes. In *Fast Software Encryption – FSE 2011*, pages 306–327, 2011.

19. H. Kuwakado and M. Morii. Security on the quantum-type even-mansour cipher. *2012 International Symposium on Information Theory and its Applications*, pages 312–316, 2012.
20. X. W. Lipeng Chang, Yuechuan Wei and X. Pan. Collision forgery attack on the aes-otr algorithm under quantum computing. *Symmetry*, 2022. `https://doi.org/10.3390/sym14071434`.
21. V. Maram. Private communication, 2023.
22. V. Maram, D. Masny, S. Patranabis, and S. Raghuraman. On the quantum security of OCB. *IACR Transactions on Symmetric Cryptology*, 2022(2):379–414, 2022.
23. K. Minematsu. Parallelizable rate-1 authenticated encryption from pseudorandom functions. In *Advances in Cryptology – EUROCRYPT 2014*, pages 275–292, 2014.
24. K. Minematsu. Aes-otr v3.1. Third-Round Candidate Submission to CAESAR Competition, 2016. `https://competitions.cr.yp.to/round3/aesotrv31.pdf`.
25. B. Poettering. Shorter double-authentication preventing signatures for small address spaces. Cryptology ePrint Archive, Report 2018/223, 2018. `https://eprint.iacr.org/2018/223`.
26. P. Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In *Advances in Cryptology – ASIACRYPT 2004*, pages 16–31, 2004.
27. P. Rogaway, M. Bellare, J. Black, and T. Krovetz. OCB: A block-cipher mode of operation for efficient authenticated encryption. In *ACM CCS 2001: 8th Conference on Computer and Communications Security*, pages 196–205, 2001.
28. P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41(2):303–332, 1999.
29. D. R. Simon. On the power of quantum computation. *SIAM Journal on Computing*, 26(5):1474–1483, 1997.
30. H. Thapliyal, N. Ranganathan, and S. Kotiyal. *Reversible Logic Based Design and Test of Field Coupled Nanocomputing Circuits*, pages 133–172. 2014.

A AEAD Definitions

We begin by defining a *nonce-based authenticated encryption with associated data (AEAD) scheme* as both AES-OTR and OPP are one.

Definition 4. *(Nonce-based AEAD scheme) A nonce-based AEAD scheme is a tuple of probabilistic polynomial-time algorithms Π = (Enc,Dec) with a key space $\mathcal{K} = \{0,1\}^k$ such that*

Enc(K, N, A, M)**:** *This algorithm takes as input a key K, a nonce N, associated data (AD) A and a message M. It produces as an output a ciphertext C and a tag T. We write $(C, T) \leftarrow Enc_K(N, A, M)$.*

Dec(K, N, A, C, T)**:** *This algorithm takes as input a key K, a nonce N, AD A, a ciphertext C and a tag T and outputs a message M or $\perp$. We write $Dec_K(N, A, C, T)$ to denote this output.*

The AEAD scheme needs to satisfy correctness, which is given, if for any N, A and M we have

$$Pr[Dec_K(N, A, Enc_K(N, A, M)) = M] \geq 1 - \epsilon,$$

$\forall K \in \mathcal{K}$ and $\epsilon > 0$ is negligible.

B Prior Quantum Attacks on Unforgeability of AES-OTR

Prior work in [20] already proposed an approach to attack the unforgeability of AES-OTR which uses quantum access to the encryption oracle and applies Simon's algorithm. The way associated data is processed and its impact on the tag T (more specifically on the intermediate variable TA) is exploited. Here we give a short description of said properties and refer to [20] for specific details.

On a high level, the processing of associated data, whether in parallel or serial, enables the derivation of collisions for the intermediate variable TA, which is utilized to compute the tag T. For convenience, we assume that the last block of AD is always of full size to avoid having to treat it differently in the analysis as padding would be applied otherwise. Additionally, for the sake of convenience we introduce a subscript to the value TA in order to keep track to which set of AD it belongs. E.g. for a set of associated data A we set $TA_A = \text{AF-P}_K(A)$.

B.1 Finding Collisions when AD Processed in Parallel.

We first consider the case when associated data is processed in parallel as described in Algorithm 5 with output TA. Given a set of associated data $A = A_1||A_2||...||A_a$, where $A_i \in \{0,1\}^n \ \forall i \in [a]$, we define a set of intermediate variables $X_A[i]$ for $i \in [a-1]$ as $X_A[i] = E_K(A_i \oplus 2^{i-1}Q)$ such that (see Figure 1)

$$TA_A = E_K(X_A[1] \oplus ... \oplus X_A[a-1] \oplus A_a \oplus 2^{a-1}3^2Q).$$

Note that even though the value Q is constantly being updated in Algorithm 5, we here consider Q to be a fixed value and set it to $Q = E_K(0^n)$.

For the given associated data A we can construct a second set of associated data $B = B_1||...||B_a$, $A \neq B$, where $B_1 = A_2 \oplus 3Q, B_2 = A_1 \oplus 3Q$ and $B_i = A_i$ for $i \in \{3, ..., a\}$ which yields the same value TA after being processed in parallel. Note, that $3Q = Q \oplus 2Q$ by definition (see Section 2). This implies

$$\begin{aligned} X_B[1] &= E_K(B_1 \oplus Q) = E_K(A_2 \oplus 3Q \oplus Q) = E_K(A_2 \oplus 2Q) = X_A[2] \\ X_B[2] &= E_K(B_2 \oplus 2Q) = E_K(A_1 \oplus 3Q \oplus 2Q) = E_K(A_1 \oplus Q) = X_A[1] \\ X_B[i] &= X_A[i] \end{aligned}$$

for $i \in \{3, ..., a\}$ and therefore

$$\begin{aligned} TA_B &= E_K(X_B[1] \oplus X_B[2] \oplus X_B[3] \oplus ... \oplus X_B[a-1] \oplus B_a \oplus 2^{a-1}3^2Q) \\ &= E_K(X_A[2] \oplus X_A[1] \oplus X_A[3] \oplus ... \oplus X_A[a-1] \oplus A_a \oplus 2^{a-1}3^2Q) = TA_A \end{aligned}$$

B.2 Finding Collisions when AD Processed in Serial.

We now consider the case when associated data is processed in serial as described in Algorithm 6 with output TA. For a given set of AD $A = A_1||...||A_a$ with $A_i \in \{0,1\}^n$, we want to produce a second set of AD $B = B_1||...||B_{a-1}$, where $A \neq B$, such that $TA_A = TA_B$.

Again, we define some intermediate variables $X_A[i]$ for $i \in [a-1]$ as (see Figure 2):

$$X_A[1] = A_1,\ X_A[j] = A_j \oplus E_K\big(X_A[j-1]\big),\ X_A[a] = 4Q \oplus A_a \oplus E_K\big(X_A[a-1]\big)$$

for $j \in \{2, ..., a-1\}$ and $Q = E_K(0^n)$ such that $TA_A = E_K\big(X_A[a]\big)$.

We construct a second set of associated data $B = B_1||...||B_{a-1}$, where $B_1 = A_2 \oplus E_K(A_1)$ and $B_i = A_{i+1}$ for $i \in \{2, ..., a-1\}$. We claim that this causes the TA values to coincide. We want to show that $X_A[j] = X_B[j-1] \forall j \in [a]$ in a proof by induction on $2 \leq j \leq a-1$ and first observe that

$$X_A[2] = A_2 \oplus E_K\big(X_A[1]\big) = A_2 \oplus E_K(A_1) = X_B[1].$$

We now assume $X_A[j] = X_B[j-1]$ as the induction hypothesis. It holds

$$X_A[j+1] = A_{j+1} \oplus E_K\big(X_A[j]\big) = B_j \oplus E_K\big(X_B[j-1]\big) = X_B[j].$$

Here the second equality uses the induction hypothesis and the definition of B. Thus, we may conclude by induction that

$$\begin{aligned} X_B[a-1] &= 4Q \oplus B_{a-1} \oplus E_K\big(X_B[a-2]\big) \\ &= 4Q \oplus A_a \oplus E_K\big(X_A[a-1]\big) = X_A[a]. \end{aligned}$$

And therefore we get

$$TA_B = E_K\big(X_B[a-1]\big) = E_K\big(X_A[a]\big) = TA_A. \tag{B.1}$$

This means that for any given AD $A = A_1||...||A_a$, the above constructed AD B produces the same TA value.

Notice that the observations on TA values made in the above parallel and serial AD cases are in the classical setting. We now come back to the discussion of the quantum aspect of the attacks presented in [20]. At a high level, they use Simon's algorithm with respect to the quantum accessibility of the encryption oracle to generate such collisions. These collisions are then used to construct forgeries in the quantum setting.

C Assuming $\tau = n$ in the IND-qCPA Attack on AES-OTR with Parallel AD Processing

Even if AES-OTR is recommended to be instantiated using untruncated tags as mentioned in Section 3.2, we want to discuss the consequences if this is not the case.

Before we gained raw block cipher access in Section 3.2, we assumed that $\tau = n$. This is important because AES-OTR is designed to output the value $T = \mathrm{msb}_\tau(TE \oplus TA)$ as the tag of the corresponding plaintext-AD pair. If the plaintext is empty as it is defined for Function 3.1, but $\tau < n$ instead, as a

consequence the tag only gives us the first τ bits of $TE \oplus TA$ by definition. This means that Equation 3.2 would now be

$$f^{(i)}(0) \oplus f^{(i)}(1) = i\text{th bit of} \Big\{ \mathrm{msb}_\tau\big(E_K(0^n) \oplus E_K(inp)\big) \Big\}$$

for $i \in [\tau]$ and thus instead of recovering $E_K(inp)$ we would only be able to recover $\mathrm{msb}_\tau(E_K(inp))$ by following the same reasoning as above.

This difference affects our attack, as the ability to compute the value V in Step 3 is based on having raw block cipher access to all of the n bits of the (inner) $E_K(\mathrm{Format}(\tau, N))$ value, because we need to apply the raw block cipher access again to this value (of now only τ bits) to compute V. Thus, we cannot compute V as straight forward as before, as we are missing information on the last $n - \tau$ bits.

Now if the tags are not truncated significantly, then we can salvage our IND-qCPA attack by just brute-forcing all possible values of the remaining $n - \tau$ bits of $E_K(\mathrm{Format}(\tau, N))$. However if the truncation is significant, then this brute-forcing strategy becomes inefficient; we leave it as an interesting open question to analyze IND-qCPA security of AES-OTR in this latter case.

We also want to point out that the IND-qCPA attack on OCB2 in [22, Section 4.3] also runs into the very same issue but it isn't addressed accordingly. This issue was confirmed by the authors in [22] and that they indeed consider only untruncated tags in their attacks [21].

D IND-qCPA Attack on AES-OTR with Empty AD

So far, our IND-qCPA analysis of AES-OTR relied on exploiting the processing of AD in either parallel or serial manner. We now want to break confidentiality by considering AES-OTR as a pure AE scheme. To do so, we consider a stronger adversary that can adaptively pick the nonces (in a non repeating manner) and hand it to the challenger. The challenger then has to respond to encryption queries in the IND-qCPA security game using these nonces. We adapt the attack model from [22, Section 4.4], as well as the general strategy.

The strategy in this case is to recover the value $U = E_K(\mathrm{Format}(\tau, N))$ for a randomly chosen nonce N using Simon's algorithm and picking a new nonce based on the recovered value. The challenger then has to use this newly picked nonce to answer the challenge query which enables us to perform an IND-qCPA attack. However, the formatting of the nonce causes our attack to be a non-trivial extension to the one in [22]: before picking the new nonce, we have to check whether U has a desired format and if this is not the case, we have to repeat the recovery step. Observe, that here compared to our attacks in Sections 3.2 and 3.3, the value we want to recover is dependent on the nonce N. So the strategy to use $\mathcal{O}(n)$ quantum oracle queries in an application of Simon's algorithm with respect to a function as used in the previous section would not work. The reason is that if Simon's function has a period depending on U, then in each step of Simon's algorithm, we would recover a vector orthogonal to a

different value of the period. This issue arises because the nonce changes with every query and hence, U and the corresponding period also change. To resolve this obstacle, we define a function with numerous independent periods and an input that consists of multiple plaintext blocks (specifically, $4n$ blocks). By using this approach, that is inspired by the attack presented in [4, Section 3.2] (and [22, Section 4.4]), we can overcome the challenge of changing nonces, as we describe in this section.

Note, that even though encryption of AES-OTR has some slight differences when AD is processed in either parallel or serial, Algorithm 4 executes the same steps where only the value U differs by a factor of 2 if the associated data is kept empty. Thus, our attack works in both settings, but we treat the parallel case here.

Let $m \neq 0$ be even. Recall that for a message $M = M_1||...||M_m||0^n$ with $M_i \in \{0,1\}^n \ \forall i \in [m]$ we get ciphertext $C = C_1||...||C_m||C_{m+1}$ where

$$C_{2k-1} = E_K\Big(2^{k-1}U \oplus M_{2k-1}\Big) \oplus M_{2k}$$
$$C_{2k} = E_K\Big(2^{k-1}3U \oplus C_{2k-1}\Big) \oplus M_{2k-1}$$

for $k \in [m/2]$ are n-bit blocks as well. We set the last plaintext block to be all zeros since this block is encrypted differently, and we only want to focus on properties of the encryption of the first m blocks.

We define functions $h_{2k-1} : \{0,1\}^n \to \{0,1\}^n$ such that

$$h_{2k-1}(M) = E_K\Big(2^{k-1}U \oplus M\Big) \oplus M$$

for $k \in [m/2]$ which correspond to the $(2k-1)$-th ciphertext block C_{2k-1} when $M_{2k-1} = M_{2k} =: M$. Further, by defining $s := 2^{k-1}U \oplus 2^kU$ we find that

$$\begin{aligned} h_{2k-1}(M \oplus s) \oplus h_{2(k+1)-1}(M \oplus s) &= E_K\big(2^{k-1}U \oplus M \oplus s\big) \oplus E_K\big(2^kU \oplus M \oplus s\big) \\ &= E_K\big(2^kU \oplus M\big) \oplus E_K\big(2^{k-1}U \oplus M\big) \\ &= h_{2(k+1)-1}(M) \oplus h_{2k-1}(M). \end{aligned} \tag{D.1}$$

So, if we define $H_{2k-1,2(k+1)-1} : \{0,1\}^n \to \{0,1\}^n$ as

$$H_{2k-1,2(k+1)-1}(M) = h_{2k-1}(M) \oplus h_{2(k+1)-1}(M)$$

this function is in fact periodic with period $s = 2^{k-1}U \oplus 2^kU$, as the above calculations indicate.

With the function $H_{2k-1,2(k+1)-1}$ we basically link four consecutive plaintext blocks (more precisely we set $M = M_{2k-1} = M_{2k} = M_{2(k+1)-1} = M_{2(k+1)}$) in order to create a periodic function. We will further develop this idea by first encrypting $4(n+1)+1$ plaintext blocks followed by an application of a linear function that captures exactly this observation.

Consider the function $g : \{0,1\}^{n(4(n+1)+1)+\tau} \to \{0,1\}^{n(n+1)}$

$$g(C_1, ..., C_{4(n+1)+1}, T) = (C_1, C_5 \oplus C_7, C_9 \oplus C_{11}, ..., C_{4n+1} \oplus C_{4n+3})$$

Here, the C_i's are n-bit blocks and T is a τ-bit block. It is not hard to see that g satisfies $g(C \oplus C') = g(C) \oplus g(C')$ for any valid inputs C and C' i.e. g is a linear function.

We further define the function $f_N : \{0,1\}^{n^2} \to \{0,1\}^{n(n+1)}$ such that

$$f_N(M_1, ..., M_n) = g \circ \text{OTR-}\mathcal{E}_{K,p}(N, \varepsilon, 0^{4n}||M_1^4||...||M_n^4||0^n)$$

for some randomly chosen nonce N and empty AD[2]. Here, we included the last plaintext block 0^n since the last block (note, that we have an odd amount of blocks) is treated differently by the encryption algorithm and we want to avoid having to analyze encryption in this case. This choice allows us to just ignore the last block of ciphertext as it is irrelevant for our attack. We can also write f_N in terms of the functions h_{2k-1} and $H_{2k-1,2(k+1)-1}$ from above. To be precise, it holds

$$f_N(M_1, ..., M_n) = \big(h_1(0^n), H_{5,7}(M_1), ..., H_{4n+1,4n+3}(M_n)\big). \tag{D.2}$$

We proceed by claiming that f_N has the n linearly independent periods $\langle s_i \rangle_{i \in [n]}$ with $s_i = (s_{i,1}, ..., s_{i,n})$ and $s_{i,j} \in \{0,1\}^n$, where

$$s_i = \Big((0^n)^{i-1}||2^{2i}U \oplus 2^{2i+1}U||(0^n)^{n-i}\Big).$$

Hence, only the entry $s_{i,i}$ is non-zero $\forall i \in [n]$. This is the crucial property required for our attack to succeed, as we then are able to let f_N be Simon's function to which we can apply Simon's algorithm. Indeed, when we take a look at $f_N(M_1, ..., M_i \oplus 2^{2i}U \oplus 2^{2i+1}U, ..., M_n)$ it is enough to consider the $(i+1)$-st entry (as it is the only entry affected by the period s_i) given from Equation D.2 for which it holds

$$H_{4i+1,4i+3}(M_i \oplus s_{i,i}) = H_{4i+1,4i+3}(M_i),$$

as we have shown in Equation D.1 (with suiting choice of indices).

Hence we can follow the reasoning in [22, Section 4.4] and apply [4, Lemma 2] that assures the ability to compute a linear function of an output of a quantum oracle.[3] Using this lemma, we can compute f_N with a single quantum query to the OTR-$\mathcal{E}_{K,p}(N, \cdot)$ oracle. Again, with similar arguments as in [22] this in turn allows us to apply Simon's algorithm to f_N where with a *single* quantum query

[2] Here, by M_i^4 we mean the concatenation of four copies of M_i.

[3] Note that the linear function g above does not use certain input ciphertext blocks, such as blocks of the form C_{2i}. In our application of [4, Lemma 2], these blocks are essentially "truncated out" wherein the qubit registers corresponding to such blocks are initialized with the uniform superposition $\sum_{z=0}^{2^n-1} |z\rangle$. After our quantum query to the OTR-$\mathcal{E}_{K,p}(N, \cdot)$ oracle, when the resulting ciphertext blocks are xored to these registers, the uniform superposition remains intact. Hence we can remove such registers as they are not entangled with other registers of relevance, thereby making sure that the "truncated" registers do not affect our subsequent application of Simon's algorithm.

we are able to recover a vector $y = (y_1, ..., y_n) \in \{0,1\}^{n^2}$, $y_i \in \{0,1\}^n \forall i \in [n]$ that is orthogonal to each of the n periods $\langle s_i \rangle_{i \in [n]}$.

With overwhelming probability the algorithm successfully computes such a vector y because we can apply a similar argument as in [4] to argue that there do not exist any "unwanted periods" in f_N to which y could be orthogonal to. To be precise, if we assume the existence of an unwanted period s' of f_N with a probability greater than $\frac{1}{2}$, then at least one of the $H_{4i+1,4i+3}$ in Equation D.2 would also admit an unwanted period $s'_{4i+1,4i+3}$ with probability greater than $\frac{1}{2n}$. But this is impossible as E_K being AES does not have a high-probability higher-order differential.

Hence, we can continue by solving the n equations we get from orthogonality

$$y_i \cdot (2^{2i}U \oplus 2^{2i+1}U) = 0$$

for $i \in [n]$ and we are thus able to recover the value $U = E_K(\text{Format}(\tau, N))$. Using only a single quantum query to recover U is crucial. Indeed, if we want to compute $U = E_K(\text{Format}(\tau, N))$ and we make a second quantum encryption query, then the nonce changes to an independent N'. Hence, the corresponding output of Simon's algorithm with respect to the second quantum encryption query depends on N' but is (most likely) independent of N. Therefore it is not cleat how the system of linear equations obtained with respect to the second query (with independent nonce N') helps in recovering $U = E_K(\text{Format}(\tau, N))$.

Furthermore, as described in [22] we are also able to recover the *fixed* first ciphertext block $C_1 = E_K(U)$ by measuring the quantum register corresponding to the output of f_N as a part of Simon's algorithm. It is important that the plaintext block is fixed to 0^n (or any other fixed classical value), so that C_1 is a classical value.

Using these values we can now formulate our IND-qCPA attack in a setting with adaptively chosen nonces. Note however, that in difference to the attack in [22], we need an additional step as here the value U differs from the one used in OCB2: for AES-OTR there is an additional formatting applied to the nonce N which does not exist for OCB2. This creates the problem that we cannot just define our new nonce N^* to be the recovered value U as it was the case in the [22] attack. because it may not have the correct format to satisfy $C_1 = E_K(U) = E_K(E_K(\text{Format}(\tau, N^*))$ which is what we need in order for our attack to work. We resolve this issue by iterating the above steps for different initial nonces N, until we find one that satisfies the desired condition. We will formalize this idea now:

1. For a randomly chosen nonce N we recover in a *single* quantum encryption query the classical values $U = E_K(\text{Format}(\tau, N))$ and $C_1 = E_K(U)$ using Simon's algorithm as described above.
2. Check if U is of the form

   $$U = \text{Format}(\tau, N') = \text{bin}(\tau \bmod n, 7) || 0^{n-8-|N'|} || 1 || N'$$

   for some $N' \in \{0,1\}^{8i}$ where $i \in [15]$. If this condition is not satisfied, we repeat step 1 for a different nonce N and else we continue with step 3.

3. Choose the nonce N^* to be $N^* = N'$ such that we guarantee the corresponding initial offset U^* in the challenge query to be

$$U^* = E_K(\text{Format}(\tau, N^*)) = E_K(U) = C_1$$

where C_1 is the value we recovered in the first step.
4. Define $m_0 = U^* \oplus U$, $m_1 = 0^n$, where U is the value we recovered in the first step with the desired format of step 2, and a random $m_0' \in \{0,1\}^n$ such that $m_0 \neq m_0'$. Select the two 2-block messages as $M_0 = m_0||m_1$ and $M_1 = m_0'||m_1$ and $A = \epsilon$ for the challenge query.
5. Record the the response (C^*, T^*) from the challenger, where $C^* = C_1^*||C_2^*$ with $C_i^* \in \{0,1\}^n$ and output $b' = 0$ if $C_2^* = C_1$ and $b' = 1$ else.

The attack succeeds because as a result of our choice of messages and nonce for the challenge query the response (C^*, T^*) satisfies

$$C_2^* = \begin{cases} E_K(U^* \oplus m_0) \oplus m_1 = E_K(U) = C_1 & \text{if } M_0 \text{ was encrypted.} \\ E_K(U^* \oplus m_0') \oplus m_1 = E_K(U^* \oplus m_1') \neq C_1 & \text{if } M_1 \text{ was encrypted.} \end{cases}$$

Note, that when we choose M_0 and M_1 to have two blocks of plaintext, they are encrypted following the case in lines 11 to 14 of Algorithm 4.

Also, Step 2 succeeds with high probability. Indeed, if we assume that the size of the nonce is $\kappa = 120$ (in general $\kappa \in \{8, 16, ..., 120\}$) we essentially require the first 8 bits of U to be fixed to $\text{bin}(\tau \bmod n, 7)||1$. This happens with probability $(1/2)^8$ and is even bigger when we allow $\kappa \in \{8, 16, ..., 120\}$ (note, that then in general $n - \kappa$ bits need to be fixed). Therefore this step eventually is successful.

E Quantum Key-Recovery Attack on MEM in [17]

We want to mention that there does already exist a quantum key-recovery attack on the Even-Mansour construction first described in [19] and further discussed in [17, Section 3.2]. We want to highlight the novelty of our quantum key-recovery attack on OPP with respect to the existing attacks on Even-Mansour based schemes (as OPP itself uses the Even-Mansour construction). We first give a short recap on the attack as described in [17].

Starting from the block cipher

$$E_{k_1,k_2}(M) = P(M \oplus k_1) \oplus k_2$$

where P is some public permutation, [17, Section 3.2] shows that either there exists a classical distinguishing attack (though this case is negligible with random P), or, using Simon's algorithm, we are able to successfully recover k_1. This is done by applying Simon's algorithm to the periodic function defined by

$$\begin{aligned} f &: \{0,1\}^n \to \{0,1\}^n \\ & f(M) = E_{k_1,k_2}(M) \oplus P(M) = P(M \oplus k_1) \oplus P(M) \oplus k_2 \end{aligned}$$

where $f(M \oplus k_1) = f(M)$. As a result, k_1 is recovered in n quantum queries. It is important to highlight that, in this particular case of the *isolated* Even-Mansour construction, both k_1 and k_2 are keys of a fixed value. If we compare this to the setting of OPP, we have that $k_1 = k_2 = \delta(K, X, (i, 0, 1)) = \varphi^{i+2}(\Omega) \oplus \varphi^{i+1}(\Omega) \oplus \varphi^i(\Omega)$ with $\Omega = P(X||K)$ and $X = \text{pad}^0_{n-\kappa-k}(N)$. The OPP setting is more complex as there encryption is dependent on a nonce N, which changes with each call to the encryption oracle. As a result, the value of f, and even more importantly the period k_1 of f, would change in each of the n iterations during an application of Simon's algorithm. This is why we cannot just extend the attack on MEM in [17] to OPP, as in each application it would recover a vector y orthogonal to a different period. Recall that in our quantum key-recovery attack in Section 4.2 we overcame this issue as we only use one *single* quantum query to the OPP-$\mathcal{E}(K, N, \cdot)$ oracle to compute the periodic function $\tilde{f}_N$ in 4.1 in order to recover Ω and thus the key K using Simon's algorithm.

To emphasize, the main difference between our attack and the existing one is twofold. Firstly, we are focusing on the authenticated encryption setting, which includes changing nonces that make the straight forward application of Simon's algorithm infeasible. Secondly, our attack is more efficient in terms of numbers of queries, as it only requires one single quantum encryption query, whereas the existing attack requires n queries.