Single-query Quantum Hidden Shift Attacks

Xavier Bonnetain[1] & André Schrottenloher[2]

[1] Université de Lorraine, CNRS, Inria, LORIA, Nancy, France
[2] Univ Rennes, Inria, CNRS, IRISA
firstname.lastname@inria.fr

Abstract. Quantum attacks using superposition queries are known to break many classically secure modes of operation. While these attacks do not necessarily threaten the security of the modes themselves, since they rely on a strong adversary model, they help us to draw limits on the provable security of these modes.
Typically these attacks use the structure of the mode (stream cipher, MAC or authenticated encryption scheme) to embed a period-finding problem, which can be solved with a dedicated quantum algorithm. The hidden period can be recovered with a few superposition queries (e.g., $O(n)$ for Simon's algorithm), leading to state or key-recovery attacks. However, this strategy breaks down if the period changes *at each query*, e.g., if it depends on a nonce.
In this paper, we focus on this case and give dedicated state-recovery attacks on the authenticated encryption schemes Rocca, Rocca-S, Tiaoxin-346 and AEGIS-128L. These attacks rely on a procedure to find a Boolean hidden shift with a *single* superposition query, which overcomes the change of nonce at each query. As they crucially depend on such queries, we stress that they do not break any security claim of the authors, and do not threaten the schemes if the adversary only makes classical queries.

Keywords: Quantum cryptanalysis, Quantum Fourier Transform, Authenticated encryption, Boolean hidden shift, Rocca, Tiaoxin, AEGIS

1 Introduction

Since Shor [38], the enhanced computational power of quantum devices has been known to impact the security of public-key cryptosystems. Nowadays, *post-quantum* (public-key) cryptography is structured around several computational problems (e.g., lattice sieving, decoding random codes...) which are believed to remain intractable.

The situation is more favorable in symmetric (secret-key) cryptography, since most of it is expected to remain secure. Generic attacks on primitives are now well understood, for example Grover's quantum search [15] that accelerates the recovery of a secret key from a time $\mathcal{O}(2^\kappa)$ to $\mathcal{O}\left(2^{\kappa/2}\right)$, or the BHT algorithm [11] which accelerates n-bit collision search from $\mathcal{O}\left(2^{n/2}\right)$ to $\mathcal{O}\left(2^{n/3}\right)$. Afterwards, many dedicated quantum attacks have been introduced, whether on block ciphers [8,23] or hash functions [19]. Most of the time, these attacks reach at most

a quadratic speedup (like Grover's search). In this paper, we focus on *superposition attacks* on modes of operation, which are known to allow super-quadratic speedups or sometimes total breaks of classically-secure schemes.

Superposition Queries. The literature separates quantum attacks on symmetric schemes in two categories. In the *Q1 setting*, the adversary has only classical access to the attacked function, typically an encryption scheme or MAC which contains secret information (the key or internal states). Such attacks follow the main threat model of post-quantum cryptography, which is that of an adversary recording computations to decrypt them later in time. In the *Q2 setting*, also named *superposition query model*, the adversary can query the function as a *quantum oracle*, i.e., from within a quantum computation. Obviously, this cannot model a "store now, decrypt later" scenario anymore. Despite this lack of practical applications, Q2 attacks are still a relevant source of information on the quantum security of these schemes, as they are known to break many classically secure modes of operation [24,25,22,26]. On the one hand, they can be used as a starting point or motivation for improved Q1 attacks [6,9]. On the other hand, they can be seen as impossibility results, showing that any security proof must consider an adversary making classical queries to the scheme [1].

Principle of Q2 Breaks. Consider a symmetric scheme $E_K : \{0,1\}^n \to \{0,1\}^m$ with a secret key K, to which we have quantum access.

Typically, Q2 attacks will combine some *pre-processing* function f and *post-processing* function g so that the function $g \circ E_K \circ f$ has some property that can be exploited. For example, the Even-Mansour cipher: $E_{k_1,k_2} : x \mapsto k_1 \oplus \Pi(k_2 \oplus x)$, where Π is a public permutation, can be attacked by noticing that $E_{k_1,k_2} \oplus \Pi$ is a periodic function on $\mathbb{F}_2^n$, of period k_2 [25]. Simon's algorithm [39] can recover this period in $\mathcal{O}(n)$ quantum queries. Other attacks (for example in [7], using a non-trivial f) may target an internal state instead. In MACs and authenticated encryption (AE) schemes, this can lead to forgeries.

A typical limitation of Q2 attacks is when the construction E_K admits a *nonce* N, like many MACs and AE schemes. It is indeed common [2] to assume that nonces remain classical values, and that they are not repeated from one Q2 query to another. While many attacks can also bypass the use of nonces [5,7], they cannot apply in a situation where we would query: $E_{K,N}(x) = f(x \oplus s(K,N))$ where the secret internal state s depends on K *and* N.

New Strategy. In this paper, we use a hidden shift algorithm with a *single query* from [32]. It follows a well-known strategy in quantum computing which was previously applied in [13,33] and requires, in our case, a combination with a state preparation technique [36].

We consider several AE schemes, where the recovery of the internal state leads to forgery or key-recovery attacks. Our strategy is to perform a superposition query with several message blocks which, with proper post-processing, can be turned into an oracle:

$$|x\rangle\,|0\rangle \mapsto |x\rangle\,g(x \oplus s')\,|s\rangle \quad ,$$

Table 1. New quantum attacks and comparison with generic attacks ("Grover"). "Toffoli" is an approximate count of the Toffoli gates applied during the attack. Approximately 10^3 to 10^4 qubits are required for all attacks, since the internal state of the schemes is of order 10^3 bits.

Target	Type	Setting	Queries	Toffoli	Classical time	Method
Rocca	key	Q1	1 (encr.)	2^{145}	negl.	Grover
	forgery	Q2	2^{64} (decr.)	2^{80}	negl.	Grover
			2^{57} (encr.)	2^{80}	negl.	Section 4.1
			$2^{46.4}$ (encr.)	$2^{68.4}$	$2^{125.4}$	Section 4.1
Rocca-S	key	Q1	1 (encr.)	2^{145}	negl.	Grover
	forgery	Q2	2^{30} (encr.)	2^{98}	negl.	Section 4.2
			2^{63} (encr.)	2^{85}	negl.	Section 4.2
Tiaoxin	key	Q1	1 (encr.)	2^{81}	negl.	Grover
		Q2	2^{34} (encr.)	2^{56}	negl.	Section 4.3
AEGIS-128L	key	Q1	1 (encr.)	2^{81}	negl.	Grover
	forgery	Q2	2^{62} (decr.) + 2^{56} (encr.)	2^{78}	negl.	Section 4.4

where g is a function to $\{-1, 1\}$, and s and s' are values which, together, allow to determine a whole internal state. We measure s immediately, but we cannot use Simon's algorithm to obtain s' since it depends on the nonce, and will change at the next query.

Instead, we use the hidden shift algorithm from [32]. This algorithm performs a Hadamard transform:

$$\frac{1}{2^{n/2}} \sum_x g(x \oplus s') \left|x\right\rangle \overset{H}{\longmapsto} \frac{1}{2^n} \sum_y (-1)^{s' \cdot y} \widehat{g}(y) \left|y\right\rangle \quad ,$$

with $\widehat{g}$ the Walsh-Hadamard transform of g. It then computes a multiplication by $1/\widehat{g}(y)$ in the amplitudes of this state. Such a multiplication cannot succeed with probability 1, but when it does, we obtain $\sum (-1)^{s' \cdot y} \left|y\right\rangle$ which, after another Hadamard transform, gives us s'. With s and s', we solve a system of equations which gives us the full internal state of the scheme.

The resulting attacks are summarized in Table 1.

Outline. We detail the targeted authenticated encryption schemes in Section 2. In Section 3, we give the quantum building blocks of our attack and analyze its success probability. In Section 4 we present our attacks.

Table 2. Summary of parameters for studied AE schemes and their bits of security (nonce-respecting) in the classical setting.

Cipher	Key size	Nonce size	Tag size	Forgeries	Key-recovery
AEGIS-128L	128	128	128	128	128
Tiaoxin-346	128	128	128	128	128
Rocca	256	128	128	128	256
Rocca-S	256	128	256	256	256

2 Description of the Schemes

In this section, we recall the Authenticated Encryption with Associated Data (AEAD) schemes AEGIS-128L [42], Tiaoxin-346 [31], Rocca [34] and Rocca-S [28]. Some details which are not relevant to our analysis will be omitted. In particular, we omit the processing of Associated Data and the padding of input messages.

The levels of security against key-recovery and forgery are set according to the generic attacks:

- **Key-recovery:** using a single classical known-plaintext query, an adversary can always find the κ-bit key in $\mathcal{O}(2^{\kappa})$ computations ($\mathcal{O}\left(2^{\kappa/2}\right)$ in the quantum setting using Grover's algorithm [15]);
- **Forgery:** with a t-bit tag, an adversary that can make decryption queries can create a forgery in $\mathcal{O}(2^{t})$ queries classically. This attack can be accelerated quantumly if one has access to a quantum decryption oracle. This would cost $\mathcal{O}\left(2^{t/2}\right)$ quantum queries using Grover's algorithm. Of course, classically or quantumly, this attack is relevant only if the key is larger than the tag.

All these designs are known to be insecure in the nonce-misuse scenario, i.e., if the adversary is allowed to perform multiple chosen-plaintext queries with the same nonce.

2.1 AEGIS-128L

AEGIS was originally published at SAC [41], and later submitted to the CAESAR competition [43]. We will focus here on the variant AEGIS-128L, which can be found in [42]. In the CAESAR competition, AEGIS-128 appeared in the final portfolio for use case 2 (high-performance applications), and AEGIS-128L was a finalist for this use case.

All variants of AEGIS use a large internal state, made of several 128-bit *registers*, and a simple round function which updates this state and mixes it with additional registers of input (e.g., the message blocks). This round function is based on the block cipher standard AES [29].

The AES Round. We denote the AES round function as: $A = \mathsf{MC} \circ \mathsf{SR} \circ \mathsf{SB}$. It applies on a state of 128 bits, represented as a 4×4 matrix of bytes. SB (SubBytes) applies the AES S-Box in parallel to all bytes. SR (ShiftRows) shifts

row number i in the matrix by i positions left. MC (MixColumns) multiplies each column by the AES MDS matrix.

AEGIS-128L Algorithm. AEGIS-128L accepts a key and a nonce of 128 bits each. The internal state is made of eight 128-bit registers denoted $S[i], 0 \leq i \leq 7$. The round function R takes two additional 128-bit inputs X_0, X_1 and outputs $S' = R(S, X_0, X_1)$ as:

$$\begin{aligned} S'[0] &= X_0 \oplus S[0] \oplus A(S[7]) & S'[4] &= X_1 \oplus S(4] \oplus A(S[3]) \\ S'[1] &= S[1] \oplus A(S[0]) & S'[5] &= S[5] \oplus A(S[4]) \\ S'[2] &= S[2] \oplus A(S[1]) & S'[6] &= S[6] \oplus A(S[5]) \\ S'[3] &= S[3] \oplus A(S[2]) & S'[7] &= S[7] \oplus A(S[6]) \end{aligned}$$

Without AD, the algorithm has the following phases:

- **Initialization:** after loading the key K and nonce N into the state, we run 10 round updates $R(S, N, K)$
- **Encryption:** each round of encryption takes two plaintext blocks M_i, M'_i and returns two ciphertext blocks C_i, C'_i. For all $i = 0$ to $m - 1$:

$$\begin{cases} C_i = M_i \oplus S[1] \oplus S[6] \oplus \text{AND}(S[2], S[3]) \\ C'_i = M'_i \oplus S[2] \oplus S[5] \oplus \text{AND}(S[6], S[7]) \\ S \leftarrow R(S, M_i, M'_i) \end{cases} \quad (1)$$

  where AND denotes the bit-wise Boolean AND.
- **Finalization:** the state update function is called 6 times with X_0, X_1 depending on the AD length and message length. The authentication tag is obtained by XORing the 7 first registers.

Security. Third-party cryptanalysis has shown that AEGIS is nonce-misuse insecure [21] and that it exhibits linear keystream biases [14]. However, these attacks did not contradict its security claims. To the best of our knowledge, there has been no quantum security analysis of AEGIS.

2.2 Tiaoxin-346

Tiaoxin-346 was submitted to the CAESAR competition [31] where it reached the third round. It accepts 128-bit keys and 128-bit tags. The internal state T is made of thirteen 128-bit registers separated into substates T_3, T_4, T_6 with 3, 4 and 6 registers respectively denoted as $T_j[i]$. The round function $R(T, X_0, X_1, X_2)$ takes a 3-register input X_0, X_1, X_2 and updates the state as shown on Figure 1. In particular, it can be noted that the round function processes independently the substates T_j.

In the initialization phase, the key and nonce are loaded in T, then, 15 rounds of the round function $R(T, Z_0, Z_1, Z_0)$ are applied where Z_0 and Z_1 are constants.

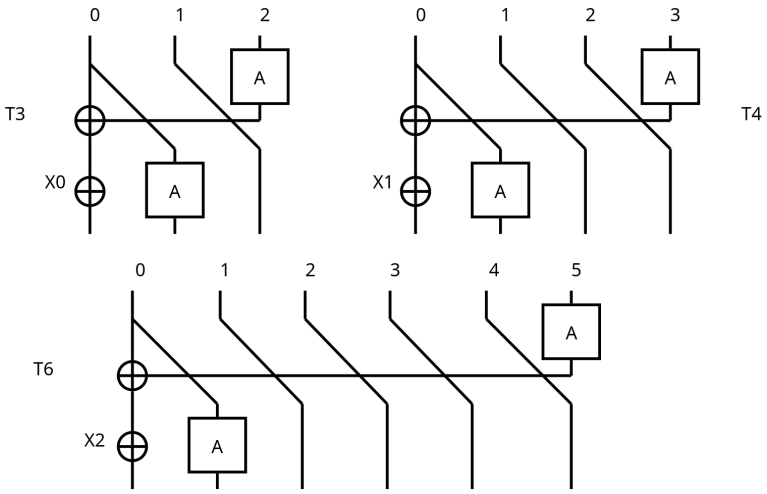

Fig. 1. Round function of Tiaoxin-346.

In the encryption phase, message blocks are also encrypted by pairs M_i, M'_i. For all $i = 0$ to $m - 1$:

$$\begin{cases} T \leftarrow R(T, M_i, M'_i, M_i \oplus M'_i) \\ C_i = T_3[0] \oplus T_3[2] \oplus T_4[1] \oplus \text{AND}(T_6[3], T_4[3]) \\ C'_i = T_6[0] \oplus T_4[2] \oplus T_3[1] \oplus \text{AND}(T_6[5], T_3[2]) \end{cases} \tag{2}$$

It can be noted that the state update is performed *before* outputting the ciphertexts, and not after like the other designs in this section. Finally, the finalization performs 20 unkeyed rounds $R(T, Z_1, Z_0, Z_1)$ and outputs the tag as the XOR of all registers $T_j[i]$.

Security. An important difference between Tiaoxin and AEGIS is that the round function of Tiaoxin is invertible, as well as the initialization phase. Thus, recovering the internal state at any point of the ciphering process leads to a key-recovery. Furthermore it is enough to recover a single substate T_j.

A few third-party works have studied the security: a key-recovery attack in a nonce-misuse scenario has been proposed [21], and Tiaoxin reduced to 8 rounds of initialization has been shown to have weak keys [27]. To the best of our knowledge, there has been no quantum security analysis of Tiaoxin.

2.3 Rocca

Rocca is an AEAD for beyond-5G applications. As such, it also aims at quantum security and uses keys of 256 bits. The internal state S is made of eight 128-bit registers denoted $S[i], 0 \leq i \leq 7$. The round function R (Figure 2) takes two

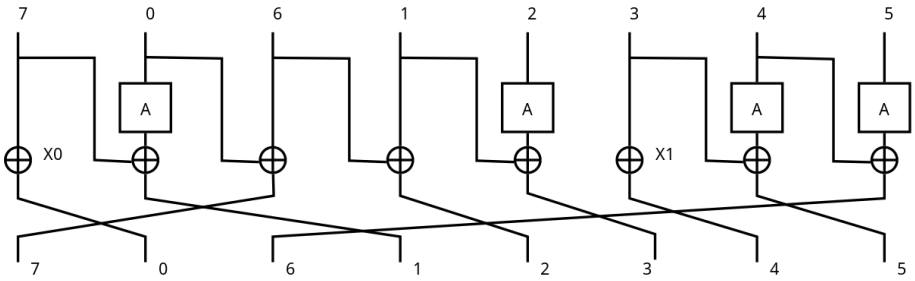

Fig. 2. Round function of Rocca.

additional 128-bit inputs X_0, X_1 and outputs $S' = R(S, X_0, X_1)$ defined as:

$$\begin{aligned}
S'[0] &= S[7] \oplus X_0 & S'[4] &= S[3] \oplus X_1 \\
S'[1] &= A(S[0]) \oplus S[7] & S'[5] &= A(S[4]) \oplus S[3] \\
S'[2] &= S[1] \oplus S[6] & S'[6] &= A(S[5]) \oplus S[4] \\
S'[3] &= A(S[2]) \oplus S[1] & S[7] &= S[0] \oplus S[6]
\end{aligned}$$

Algorithm. The specification that we give here is from the latest version (2023-03-16) of the ePrint report [35]. After the publication of the conference version [34] and subsequent third-party cryptanalysis [17], the authors added a key feedforward in the initialization phase to make it non-invertible, which was not present in the conference version.

The key is divided into two 128-bit key blocks K_0, K_1. The scheme also uses a pair of constants Z_0, Z_1. Rocca (without AD) runs as follows:

- **Initialization phase:** the state S is initialized using the nonce and key. Then, 20 rounds $R(S, Z_0, Z_1)$ are applied. Then, K_0, K_1 are XORed to $S[0], S[4]$ respectively.
- **Encryption:** message blocks are encrypted by pairs (M_i, M_i') into pairs of ciphertexts (C_i, C_i'). For all i from 0 to $m-1$:

$$\begin{cases}
C_i = A(S[1]) \oplus S[5] \oplus M_i \\
C_i' = A(S[0] \oplus S[4]) \oplus S[2] \oplus M_i' \\
S \leftarrow R(S, M_i, M_i')
\end{cases}$$

- **Finalization:** the state is updated 20 times using $R(S, |AD|, |M|)$, where $|AD|$ and $|M|$ are the respective lengths of the AD and message, and the 128-bit tag is computed as the XOR or all message blocks.

Classical Security. The authors of Rocca claimed 128-bit security against forgery attacks and 256-bit security against key-recovery attacks. Importantly, they did not make any claims in the nonce-misuse setting.

In [17], Hosoyamada et al. introduced a nonce-misuse attack that could recover the internal state using only *one* nonce-repeated pair. It follows a strategy of introducing a difference in certain message blocks, in order to observe some output differences, and solving the obtained equations to recover state values. Since the finalization function is key-less, recovering the state allows to create forgeries.

They then observed that one could turn this attack into a nonce-respecting one, by making decryption queries (which are authorized to repeat the nonces). After making a first nonce-respecting query to the encryption oracle, the adversary introduces a difference in the obtained ciphertext and tries to decrypt by trying all possible tags. If the number of decryption queries is not limited, this will eventually succeed after 2^{128} such queries, leading to a recovery of the state. In the first version of Rocca, where the initialization phase was invertible, the state recovery led to a key-recovery attack, breaking the claims. However, with the modified initialization, a recovery of the state does not lead to a recovery of the key.

Quantum Security. The authors of Rocca made no claim against Q2 attacks. Anand and Isobe studied specifically the quantum security of Rocca [3] and found a forgery attack that requires 2^{75} superposition queries. This attack is nonce-respecting and makes Q2 decryption queries.

2.4 Rocca-S

Rocca-S is a new version of Rocca which was proposed for standardization by the IETF [28]. Among the versions of the draft standard, we refer to the latest one (published march 2^{nd}, 2023).

Round Function. The internal state of Rocca-S is made of 7 registers of 128 bits. The round function $S' = R(S, X_0, X_1)$ (Figure 3) updates this state as follows:

$$\begin{aligned}
S'[0] &= S[6] \oplus S[1] & S'[4] &= A(S[3]) \oplus X_1 \\
S'[1] &= A(S[0]) \oplus X_0 & S'[5] &= A(S[4]) \oplus S[3] \\
S'[2] &= A(S[1]) \oplus S[0] & S'[6] &= A(S[5]) \oplus S[4] \\
S'[3] &= A(S[2]) \oplus S[6]
\end{aligned}$$

Algorithm. The algorithm (without associated data) runs as follows:

- **Initialization:** after loading the key K_0, K_1 and nonce, 16 rounds of $R(S, Z_0, Z_1)$ are applied, followed by a key addition in all state registers.
- **Encryption:** for all $i = 0$ to $m - 1$:

$$\begin{cases} C_i = A(S[3] \oplus S[5]) \oplus S[0] \oplus M_i \\ C'_i = A(S[4] \oplus S[6]) \oplus S[2] \oplus M'_i \\ S \leftarrow R(S, M_i, M'_i) \end{cases} \tag{3}$$

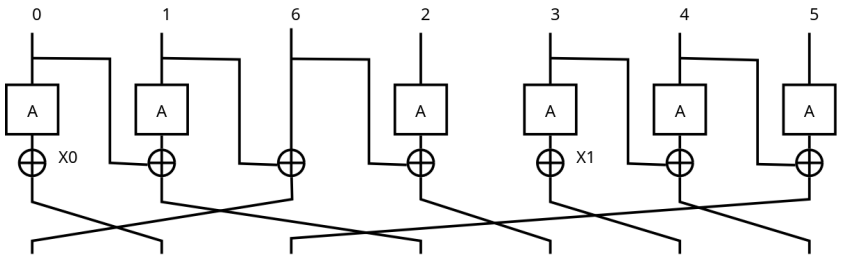

Fig. 3. Round function of Rocca-S.

– **Finalization:** the round function $R(S, |AD|, |M|)$ is iterated 16 times. Then, the 256-bit tag is computed as:

$$T = (S[0] \oplus S[1] \oplus S[2] \oplus S[3]) \| (S[4] \oplus S[5] \oplus S[6]) \ . \tag{4}$$

Security. The increased tag size allows the authors of Rocca-S to claim 256 bits of security against forgery, state and key-recovery attacks (nonce-respecting). In the quantum setting, they claim 128 bits of security against nonce-respecting forgery and key-recovery attacks. However, like Rocca, they did not consider attacks in the Q2 setting and did not make security claims in this model. To the best of our knowledge, Rocca-S remains secure in the Q1 setting.

3 Tools

In this section we give the main algorithmic tools of our attacks. We assume basic knowledge of the quantum circuit model [30] (Toffoli / CNOT / Hadamard gates, ket $|\cdot\rangle$ notations). As is commonly done in previous works [5,12,22], we query AE schemes using a *standard oracle*.

Definition 1 (Standard oracle). *For $f : \{0,1\}^n \to \{0,1\}^m$, the* standard oracle *to f is the quantum circuit O_f that maps $|x\rangle |y\rangle$ to $|x\rangle |y \oplus f(x)\rangle$.*

It is well-known that this oracle is equivalent to a *phase oracle* that maps $|x\rangle |y\rangle$ to $(-1)^{y \cdot f(x)} |x\rangle |y\rangle$, by composing it with Hadamard transform. Also, if one knows a classical circuit that implements f, both oracles are easy to construct.

These AE schemes are nonce-based. While the nonce can be chosen by the adversary, it cannot be repeated between two queries. Since Q2 queries are merely an extension of classical queries, the same can be said in the quantum setting. Therefore, we impose that *each of the Q2 queries is answered using a different, classical nonce.* Using a classical nonce or randomness is common in proofs of quantum security for encryption and AE modes [2,4].

That is, the adversary has access to a family of oracles: $O_{N,m}$ for different nonces N and message lengths m (we assume that the AD is empty), and they cannot make two queries with the same nonce.

Each oracle encrypts several (pairs of) message blocks (M_i, M'_i), depending on the selected length, and returns the corresponding (pairs of) ciphertexts (C_i, C'_i), and the tag:

$$\begin{aligned} O_{N,m} \ : |M_0, M'_0, \ldots, M_{m-1}, M'_{m-1}\rangle \, |y_0, y'_0, \ldots, y_{m-1}, y'_{m-1}, y\rangle \\ \mapsto |M_0, M'_0, \ldots, M_{m-1}, M'_{m-1}\rangle \\ |y_0 \oplus C_0, y'_0 \oplus C'_0, \ldots, y_{m-1} \oplus C_{m-1}, y'_{m-1} \oplus C'_{m-1}, y \oplus T\rangle \quad . \end{aligned}$$

Quantum Search. Grover's exhaustive search algorithm [15] is a procedure to find a "good" element in a search space of size 2^n in $\left\lfloor \frac{\pi}{4} 2^{n/2} \right\rfloor$ iterates; each iterate queries a phase oracle that flips only the phase of this good element. *Amplitude amplification* [10] generalizes this to any algorithm $\mathcal{A}$ (even a quantum algorithm) that outputs a good element with probability p. It then makes about $\frac{\pi}{4} \frac{1}{\sqrt{p}}$ iterates, with two calls to $\mathcal{A}$ and one query to the oracle per iterate, to succeed with overwhelming probability.

Grover Search Cost Estimates. All AE schemes studied in this paper are based on the AES round function. Quantum attacks on them require to implement AES components. Since the scope of this paper is only to demonstrate the existence of attacks, we will use approximate quantum gate and query counts (by a factor 2 at best). For example, Table 4 in [20] gives a count $12240 = 2^{13.58}$ Toffoli gates for a full (10-round) AES-128. We use this to assume that a single round of AES can be implemented with 2^{10} Toffoli gates (we focus only on Toffoli counts for simplicity).

For all four schemes, implementing Grover's exhaustive key search requires to recompute the initialization of the scheme. The number of iterates depends on the key size (128 or 256 bits) and the cost of the Grover iterate is dominated by this initialization function which, in turn, can be estimated using the number of AES rounds it contains. These estimates are summarized in Table 3.

Table 3. Toffoli count of Grover's key search for studied schemes. The exponent is rounded to the nearest integer.

Cipher	Key length	AES rounds	Toffoli count
Rocca	256	$4 \times 20 = 80$	2^{145}
Rocca-S	256	$6 \times 16 = 96$	2^{145}
Tiaoxin	128	$6 \times 15 = 90$	2^{81}
AEGIS-128L	128	$8 \times 10 = 80$	2^{81}

3.1 Linear Post-processing

The generic approach to post-process the output of an oracle requires two identical calls, due to the reversibility of quantum computations. This is not doable

in our case since we only query the oracle once. Fortunately, truncations [18] and more generally linear functions [4] can be computed from a single call.

For our purposes, we need to separate the linear function in two parts, one of which goes directly into the phase. This can be obtained using [4, Lemma 2] as a black-box, but we give the whole proof (adapted directly from [4]) to be self-contained.

Lemma 1 (Extended linear post-processing, adapted from [4]). *Let $f : \{0,1\}^n \to \{0,1\}^m$ be a function, let O_f be a standard oracle for f: $|x\rangle|0\rangle \mapsto |x\rangle|f(x)\rangle$. Let $g : \{0,1\}^m \to \{0,1\}^\ell$ and $h : \{0,1\}^m \to \{0,1\}$ be two linear functions, i.e., $\forall x, y, g(x \oplus y) = g(x) \oplus g(y), h(x \oplus y) = h(x) \oplus h(y)$, with standard oracles O_g and O_h Then there exists a circuit implementing the operator:*

$$|x\rangle|y\rangle \mapsto (-1)^{h(f(x))}|x\rangle|y \oplus g(f(x))\rangle \quad ,$$

which makes a single *query to O_f, two queries to O_g and two queries to O_h.*

Proof. On input $|x\rangle|y\rangle$, create the uniform superposition over outputs z and append a qubit in the state $H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$:

$$|x\rangle|y\rangle \frac{1}{2^{m/2}} \sum_{z \in \mathbb{F}_2^m} |z\rangle (H|1\rangle)$$

Compute O_h with register z as input and the last qubit as output; compute O_g with register z as input and y as output:

$$|x\rangle \frac{1}{2^{m/2}} \sum_{z \in \mathbb{F}_2^m} |y \oplus g(z)\rangle |z\rangle (-1)^{h(z)} (H|1\rangle)$$

Apply O_f with register x as input and z as output:

$$|x\rangle \frac{1}{2^{m/2}} \sum_{z \in \mathbb{F}_2^m} (-1)^{h(z)} |y \oplus g(z)\rangle |z \oplus f(x)\rangle (H|1\rangle)$$

Redo the computations of O_h and O_g:

$$|x\rangle \frac{1}{2^{m/2}} \sum_{z \in \mathbb{F}_2^m} (-1)^{h(z)+h(z \oplus f(x))} |y \oplus g(z) \oplus g(z \oplus f(x))\rangle |z \oplus f(x)\rangle (H|1\rangle)$$

Erase the qubit $(H|1\rangle)$ and use the linearity of g, h to rewrite:

$$|x\rangle \frac{1}{2^{m/2}} \sum_{z \in \mathbb{F}_2^m} (-1)^{h(f(x))} |y \oplus g(f(x))\rangle |z \oplus f(x)\rangle$$
$$= (-1)^{h(f(x))} |x\rangle |y \oplus g(f(x))\rangle \frac{1}{2^{m/2}} \sum_{z \in \mathbb{F}_2^m} |z \oplus f(x)\rangle \quad .$$

The last register becomes disentangled and always contains a uniform superposition over $\{0,1\}^m$, which we can erase, leading to the result. □

In particular, we can truncate the output of a stream cipher and separate it in two parts, one that remains in the computational basis state, and one that goes into the phase.

3.2 Properties of the Walsh Transform

Let $f : \{0,1\}^n \to \{-1,1\}$ be a function. The Walsh-Hadamard transform of f is defined as: $\widehat{f}(y) = \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} f(x)$. It corresponds to the Fourier transform in the group $\mathbb{Z}_2^n$. The quantum Hadamard transform $H^{\otimes n}$ computes a Walsh-Hadamard transform on the amplitudes of its n-qubit input state. That is:

$$\frac{1}{2^{n/2}} \sum_x f(x) |x\rangle \stackrel{H^{\otimes n}}{\longmapsto} \frac{1}{2^n} \sum_y \widehat{f}(y) |y\rangle \quad .$$

In the following, we need the following important properties of the Walsh transform.

Proposition 1. *1. (Shift) Let $s \in \{0,1\}^n$ be a constant and $g : x \mapsto f(x \oplus s)$. Then for all x, $\widehat{g}(x) = (-1)^{x \cdot s} \widehat{f}(x)$.*

2. (Convolution theorem) Let $f, g : \{0,1\}^n \to \{-1,1\}$. Then for all x, $\widehat{\widehat{f}\widehat{g}}(x) = 2^n \sum_y f(y) g(x \oplus y)$.

3. (Product) Let $g_1, \ldots, g_n$ be functions of codomain $\{-1,1\}$. Then the Walsh transform of $(x_1, \ldots, x_n) \mapsto \prod_i g_i(x_i)$ is $(x_1, \ldots, x_n) \mapsto \prod_i \widehat{g_i}(x_i)$.

3.3 Quantum Hidden Shift Algorithm with a Single Query

We want to solve the following problem.

Problem 1 (Hidden shift). Let $g : \{0,1\}^n \to \{-1,1\}$ be a known function, and s a secret value. Given access to $f : x \mapsto g(x \oplus s)$, find s.

The algorithm that we present here (Algorithm 1) is from [32], and uses quantum rejection sampling. Several special cases have appeared before in cryptanalysis: for example, shifted multiplicative characters [13] and bent functions [33]. In both cases, the algorithm avoids the rejection sampling by considering a situation in which the Fourier transform of the shifted function is easy to compute: in the former case, it's a multiplicative character, and in the latter, a constant.

Analysis of Algorithm 1. The first step is to query f and to perform a Hadamard transform. This places the Walsh coefficients of f into the amplitudes of the state. Next, we remark that by Proposition 1, these coefficients are actually those of g, multiplied by $(-1)^{x \cdot s}$.

The next step is to *correct* the amplitudes by multiplying them by $1/\widehat{g}(x)$. There are several ways to perform this operation with high precision; we use the method of [36]. Typically one appends a qubit register which is transformed into a superposition of the form: $\frac{M}{|\widehat{g}(x)|} |0\rangle + |\psi\rangle$ where $|\psi\rangle$ is a superposition of

non-zero basis states. Then one tries to measure $|0\rangle$ in the last register, which collapses the state on the wanted superposition.

Ideally, we obtain the state $\sum(-1)^{x\cdot s}|x\rangle$, which is the Hadamard transform of $|s\rangle$. However, the product operation is not possible if $\widehat{g}(x)=0$. Furthermore, if the smallest values of $\widehat{g}(x)$ are very small compared to the average, the probability to measure 0 (and succeed) gets smaller. Thus, the best strategy, as suggested in [32], is to dismiss the small Walsh coefficients of g.

We introduce some bound M and do the following: we multiply by $\frac{M}{\widehat{g}(x)}$ if $|\widehat{g}(x)|\geq M$, and otherwise, by 0, meaning that we eliminate the coordinate. Let $G=\#\{x,|\widehat{g}(x)|\geq M\}$. Then, the rejection sampling succeeds with probability:

$$p=\frac{1}{2^{2n}}\sum_{x,|\widehat{g}(x)|\geq M}\frac{M^2}{|\widehat{g}(x)|^2}|\widehat{g}(x)|^2=\frac{M^2}{2^{2n}}G \ . \tag{5}$$

When it succeeds (i.e., we measure 0 in the ancilla register), we obtain:

$$\frac{1}{\sqrt{G}}\sum_{x,|\widehat{g}(x)|\geq M}(-1)^{x\cdot s}|x\rangle \ .$$

We then apply H:

$$\frac{1}{\sqrt{2^nG}}\sum_y\sum_{x,|\widehat{g}(x)|\geq M}(-1)^{x\cdot s+x\cdot y}|y\rangle \ .$$

Afterwards, the probability to measure $y=s$ is:

$$p'=\frac{1}{2^nG}\times G^2=\frac{G}{2^n}=\Pr_x(|\widehat{g}(x)|\geq M) \ . \tag{6}$$

All in all, the total probability to succeed is: $pp'=M^2G^2{2^{-3n}}$. We have to choose M to maximize this probability, which requires to know $\widehat{g}$. In the cases we are interested in, g will be the product of many small-range independent functions. Then $\widehat{g}$ is easy to compute by taking the product of Walsh coefficients (see Proposition 1).

Remark 1 (Global phase). If we have access to $\pm g(x\oplus s)$, where the leading sign is not known, it turns into a global phase that is irrelevant for the algorithm. At the final step, we will still measure s.

Remark 2 (Self-correlation). A technique similar to this algorithm appeared also in [37], where instead of dividing by the Walsh coefficient, one multiplies by it. This would compute the discrete convolution: $(f*g)(y)=\sum_x f(x\oplus y)g(x)$ in the amplitudes of the state, and lead to a similar result since $(f*g)(y)$ is greater for $y=s$. However the analysis when cutting off the small Walsh coefficients is more difficult, so we settled for the easier method.

Algorithm 1 Quantum hidden shift with rejection sampling and the technique of [36].

Input: Quantum access to $f(x) = g(x \oplus s)$ for a known g, a bound M
Output: s, with probability pp'

1: Start from n qubits initialized to 0 ▷ $|0^n\rangle$
2: Apply $H^{\otimes n}$ ▷ $\frac{1}{2^{n/2}} \sum_x |x\rangle$
3: Query f in the phase ▷ $\frac{1}{2^{n/2}} \sum_x f(x) |x\rangle$
4: Apply $H^{\otimes n}$ ▷ $\frac{1}{2^n} \sum_x \widehat{f}(x) |x\rangle = \frac{1}{2^n} \sum_x (-1)^{x \cdot s} \widehat{g}(x) |x\rangle$
5: Compute the amplitude multiplier:

$$\alpha_x := \begin{cases} M/|\widehat{g}(x)| \text{ if } |\widehat{g}(x)| \geq M \\ 0 \text{ otherwise} \end{cases} \tag{7}$$

where $0 \leq \alpha_x \leq 1$ in an additional register ▷ $\frac{1}{2^n} \sum_x (-1)^{x \cdot s} \widehat{g}(x) |x\rangle |\alpha_x\rangle$
▷ Notice that $|\widehat{g}(x)|$ is between 0 and 2^n, so we compute to an n-bit precision here
6: Compute the sign $\beta_x = sgn(\widehat{g}(x))$ in the phase ▷ $\frac{1}{2^n} \sum_x (-1)^{x \cdot s} |\widehat{g}(x)| |x\rangle |\alpha_x\rangle$
7: Append an ancilla register $|0^n\rangle$ and apply $H^{\otimes n}$ on it

$$\frac{1}{2^n} \sum_x (-1)^{x \cdot s} |\widehat{g}(x)| |x\rangle |\alpha_x\rangle \frac{1}{2^{n/2}} \sum_y |y\rangle \tag{8}$$

8: Perform a comparison between y and $2^n \alpha_x$ and store the result in a new ancilla qubit

$$\frac{1}{2^n} \sum_x (-1)^{x \cdot s} |\widehat{g}(x)| |x\rangle |\alpha_x\rangle \frac{1}{2^{n/2}} \left(\sum_{0 \leq y < 2^n \alpha_x} |y\rangle |0\rangle + \sum_{2^n \alpha_x \leq y < 2^n} |y\rangle |1\rangle \right) \tag{9}$$

9: Apply $H^{\otimes n}$ on the register holding y: the amplitude on the $|0^{n+1}\rangle$ component is α_x
10: Measure the last register. If the obtained value is different from 0^{n+1}, abort
11: Otherwise, the state has collapsed to:

$$\frac{1}{\sqrt{G}} \sum_{x, |\widehat{g}(x)| \geq M} (-1)^{x \cdot s} |x\rangle \tag{10}$$

12: Apply $H^{\otimes n}$, measure and return the result.

3.4 Hidden Shift with Smaller Correlation

For the attack on AEGIS-128L (Section 4.4), we will not be able to query a known function with a hidden shift. Instead, we can query a function of the form: $f(x) = g(x \oplus s)h(x)$ where f, g, h all take values in $\{-1, 1\}$, and h is unknown, *but biased*. We assume here that: $\Pr_x(h(x) = 1) = \frac{1}{2}(1 + c)$ for some $0 < c \leq 1$ (the *correlation*). The previous section considered the case $c = 1$. Intuitively, the closer c is to 0, the harder it will be to recover some information about s.

We follow Algorithm 1. We start by computing the state:

$$|\psi\rangle\,|1\rangle + \frac{1}{2^n}\sum_{x,|\widehat{g}(x)|\geq M}\frac{M}{\widehat{g}(x)}\widehat{f}(x)\,|x\rangle\,|0\rangle \quad .$$

Using the convolution theorem, we have that:

$$\widehat{f}(x) = 2^{-n}\sum_z (-1)^{z\cdot s}\widehat{g}(z)\widehat{h}(x \oplus z) \quad .$$

Notice that if h is constant and equal to 1, it has a single nonzero Walsh coefficient in 0 ($z = x$), equal to 2^n, and we recover the equality $\widehat{f}(x) = (-1)^{x\cdot s}\widehat{g}(x)$.

Next, we apply H, obtaining:

$$H\,|\psi\rangle\,|1\rangle + \frac{M}{2^{3n/2}}\sum_y\sum_{x,|\widehat{g}(x)|\geq M}(-1)^{x\cdot y}\frac{\widehat{f}(x)}{\widehat{g}(x)}\,|y\rangle\,|0\rangle \quad . \tag{11}$$

We focus on the amplitude of $|s\rangle\,|0\rangle$ in this state:

$$\frac{M}{2^{5n/2}}\sum_{x,|\widehat{g}(x)|\geq M}(-1)^{x\cdot s}\frac{1}{\widehat{g}(x)}\left(\sum_z (-1)^{z\cdot s}\widehat{g}(z)\widehat{h}(x \oplus z)\right) \quad . \tag{12}$$

We separate the term $z = x$ from the rest, noticing that $\widehat{h}(0) = \sum_x h(x) = 2^n c$ by our definition of c:

$$\frac{M}{2^{5n/2}}\sum_{x,|\widehat{g}(x)|\geq M}(-1)^{x\cdot s}\frac{1}{\widehat{g}(x)}\left(\sum_{z\neq x}(-1)^{z\cdot s}\widehat{g}(z)\widehat{h}(x+z)\right) + \frac{M}{2^{5n/2}}\sum_{x,|\widehat{g}(x)|\geq M}(-1)^{x\cdot s}\frac{1}{\widehat{g}(x)}(-1)^{x\cdot s}\widehat{g}(x)(2^n c) \quad , \tag{13}$$

where, introducing $G := \#\{x, |\widehat{g}(x)| \geq M\}$ like before, the second term is equal to $MGc/2^{3n/2}$.

In order to finish our analysis, we need to make a heuristic assumption. Indeed, it seems that if g and h were not sufficiently independent, then the first term could influence heavily the total amplitude. However, if they are independent and random, the first term should be positive with probability $1/2$. In that case, we can lower bound the amplitude on $|s\rangle\,|0\rangle$ by the second term.

It follows that, after performing this operation, we will measure $|s\rangle|0\rangle$ with probability at least: $2^{-1} \times (MGc)^2/2^{3n}$.

As with the $c = 1$ case, if the second register is not 0 (and only in this case), we know the process has failed. Following a similar computation, the probability to measure 0 is equal to:

$$\frac{1}{2^{4n}} \sum_{x,|\widehat{g}(x)|\geq M} \left| \frac{M}{\widehat{g}(x)} \sum_z (-1)^{z\cdot s} \widehat{g}(z)\widehat{h}(x\oplus z) \right|^2$$
$$= \frac{1}{2^{4n}} \sum_{x,|\widehat{g}(x)|\geq M} \left| M(2^n c) + \frac{M}{\widehat{g}(x)} \sum_{z\neq x} (-1)^{z\cdot s} \widehat{g}(z)\widehat{h}(x\oplus z) \right|^2 \quad (14)$$

Under the same assumption of randomness for $\widehat{h}$ (except the coefficient in 0) and $\widehat{g}$, we can assume this probability to be close to the original one, multiplied by c^2. In other words, the additional error due to the imperfect correlation intervenes in the first stage of the algorithm rather than the second one. We summarize this by the following heuristic:

Heuristic 1. *When $f(x) = g(x\oplus s)h(x)$, $\Pr_x(h(x) = 1) = \frac{1}{2}(1+c)$ and h and g are independent, the sequence of operations of Algorithm 1 succeeds with probability approximately $2^{-1}(pc^2)p'$ where $pc^2 = \frac{M^2Gc^2}{2^{2n}}$ approximates the probability to measure 0 (succeed in the first step) and $p' = \frac{G}{2^n}$ approximates the probability to succeed in the second step.*

4 Applications

4.1 State-recovery on Rocca

Our attack combines Algorithm 1 with linear post-processing to recover the internal state. Recall that the nonce and key are fixed classical values, which means that after initialization, the internal state is a fixed value. Assume that we encrypt a couple of fixed message blocks (e.g., 0), then the internal state S remains a fixed value. Our goal is to recover this S. We encrypt 5 pairs of message blocks in superposition. We unroll several of the corresponding ciphertexts:

$$C_0 = M_0 \oplus S[5] \oplus A(S[1])$$
$$C_0' = M_0 \oplus S[2] \oplus A(S[0] \oplus S[4])$$
$$C_1 = M_1 \oplus S[3] \oplus A(S[4]) \oplus A(S[7] \oplus A(S[0]))$$
$$C_1' = M_1 \oplus S[1] \oplus S[6] \oplus A(M_0 \oplus M_0' \oplus S[3] \oplus S[7])$$
$$C_2' = M_2 \oplus S[4] \oplus S[7] \oplus A(S[0]) \oplus A(S[5])$$
$$\oplus A(M_1 \oplus M_1' \oplus S[0] \oplus S[1] \oplus S[6] \oplus A(S[2]))$$
$$C_4 = M_4 \oplus S[0] \oplus S[6] \oplus A(M_0 \oplus S[7]) \oplus A\Big[M_2' \oplus S[7] \oplus A(S[0]) \oplus A(S[1] \oplus S[6])\Big]$$

$$\oplus A\Big[S[4] \oplus S[7] \oplus A(S[0]) \oplus A(S[5])\Big] \oplus A\Big[M_1 \oplus M'_0 \oplus S[0] \oplus S[3] \oplus S[6] \oplus A(M_0 \oplus M_2 \oplus S[4] \oplus S[7] \oplus A(S[5])) \oplus A(S[3] \oplus A(S[4]))\Big] \ .$$

We now introduce 5 independent 128-bit variables $X_0, \ldots, X_4$ and make the message blocks depend on them as follows (the others are simply put to 0):

$$M_0 = X_2, \qquad M'_0 = X_0 \oplus X_2, \qquad M_1 = X_0 \oplus X_2 \oplus X_4$$
$$M'_1 = X_0 \oplus X_2 \oplus X_4 \oplus X_1, \qquad M_2 = X_2, \qquad M'_2 = X_3$$

So that the following equations are satisfied:

$$M_0 \oplus M'_0 = X_0, \qquad M_1 \oplus M'_1 = X_1, \qquad M_0 = X_2$$
$$M'_2 = X_3, \qquad M_1 \oplus M'_0 = X_4, \qquad M_0 \oplus M_2 = 0$$

Next, we define:

$$\begin{aligned}
T_0 &:= S[3] \oplus S[7] \\
T_1 &:= S[0] \oplus S[1] \oplus S[6] \oplus A(S[2]) \\
T_2 &:= S[7] \\
T_3 &:= S[7] \oplus A(S[0]) \oplus A(S[1] \oplus S[6]) \\
T_4 &:= S[0] \oplus S[3] \oplus S[6] \oplus A(S[4] \oplus S[7] \oplus A(S[5])) \oplus A(S[3] \oplus A(S[4]))
\end{aligned}$$

Here, T_i is precisely the hidden shift that the variable X_i will allow us to obtain.

Because the message blocks are either constant, or linear functions of the X_i variables, we can add them into the ciphertext C_i, C'_i. Next, we use a linear post-processing (Lemma 1). This gives us access to the oracle:

$$\begin{aligned}
&|X_0, \ldots, X_4\rangle\,|0\rangle \\
&\mapsto |X_0, \ldots, X_4\rangle\,|C_0 \oplus M_0, C'_0 \oplus M'_0, C_1 \oplus M_1\rangle\,(-1)^{F(C'_1 \oplus M'_1, C'_2 \oplus M'_2, C_4 \oplus M_4)} \ ,
\end{aligned}$$

where F is a linear function that we will define. Notice first that $C_0 \oplus M_0, C'_0 \oplus M'_0, C_1 \oplus M_1$ are constants, so after making the oracle query we can already measure them without disrupting the rest of the algorithm. We obtain the following values:

$$\begin{cases}
S[5] \oplus A(S[1]) \\
S[2] \oplus A(S[0] \oplus S[4]) \\
S[3] \oplus A(S[4]) \oplus A(S[7] \oplus A(S[0]))
\end{cases} \tag{15}$$

Hidden Shift Problem. Now we define the function F. Recall that A is a single AES round, of the form: $A = \mathsf{MC} \circ \mathsf{SR} \circ \mathsf{SB}$. Let us select an 8-bit mask β of hamming weight 1 (i.e., a single bit). On input a 128-bit AES state $Z = (z_0, \ldots, z_{15})$, we define the function: $L(Z) = \sum_i \beta \cdot z_i$. Then, the function F is simply: $F(Z_1, Z_2, Z_3) = \bigoplus_i L \circ \mathsf{MC}^{-1}(Z_i)$. In other words, it removes the last

MC layer, uses a single-bit linear mask on each byte and XORs them all. By definition:

$$\begin{aligned} F(C'_1 \oplus M'_1, C'_2 \oplus M'_2, C_4 \oplus M_4) &= L \circ \mathsf{MC}^{-1}(S[1] \oplus S[6]) \oplus L \circ \mathsf{MC}^{-1} \circ A(X_0 \oplus T_0) \\ &\oplus L \circ \mathsf{MC}^{-1}(A(S[0]) \oplus A(S[5]) \oplus S[4] \oplus S[7]) \oplus L \circ \mathsf{MC}^{-1} \circ A(X_1 \oplus T_1) \\ &\oplus L \circ \mathsf{MC}^{-1}(S[0] \oplus S[6] \oplus A(S[4] \oplus S[7] \oplus A(S[0]) \oplus A(S[5])) \\ \oplus L \circ \mathsf{MC}^{-1} \circ A(X_2 \oplus T_2) &\oplus L \circ \mathsf{MC}^{-1} \circ A(X_3 \oplus T_3) \oplus L \circ \mathsf{MC}^{-1} \circ A(X_4 \oplus T_4) \ . \end{aligned}$$

Notice that $L \circ \mathsf{MC}^{-1} \circ A(X) = L \circ \mathsf{SB}(X)$ since L is invariant by permutation of the bytes. Next, we define the functions g and f:

$$\begin{cases} g(X_0, \ldots, X_4) := (-1)^{\sum_{i<5} L(\mathsf{SB}(X_i))} \\ f(X_0, \ldots, X_4) := (-1)^{F(C'_1 \oplus M'_1, C'_2 \oplus M'_2, C_4 \oplus M_4)} = \pm g(X_0 \oplus T_0, \ldots, X_4 \oplus T_4) \ , \end{cases}$$

where f has a leading unknown bit depending on the constant terms.

We will now retrieve the hidden shift $T_0, \ldots, T_4$ using Algorithm 1. We rename the individual bytes of $X_0, \ldots, X_4$ as $x_0, \ldots, x_{79}$ and rewrite g as:

$$g(x_0, \ldots, x_{79}) = \prod_{i=0}^{79} (-1)^{\beta \cdot S(x_i)} \ . \tag{16}$$

In particular, f is still a shifted version of this function. Now, to bound the runtime and success probability of Algorithm 1, we need to analyze the Walsh coefficients of g.

Analysis of $\widehat{g}$. Using Proposition 1, we notice that $\widehat{g}$, like g, is the product of 80 individual functions of one byte: $g_s : x \mapsto (-1)^{\beta \cdot S(x)}$. The Walsh transform of such a function corresponds to a line in the Linear Approximation Table of S, the AES S-Box. In fact, since we are interested only in the distribution of Walsh coefficients, all lines are equivalent, i.e., we can take any non-zero mask β. For example, we truncate the S-Box output to a single bit. The obtained distribution is given in Table 4.

Table 4. Number of occurrences of Walsh coefficients with given absolute value for a row of the AES S-box's LAT.

|LAT coefficient|	0	2^2	2×2^2	3×2^2	4×2^2	5×2^2	6×2^2	7×2^2	8×2^2
Occurences	17	48	36	40	34	24	36	16	5

It could be a priori difficult to compute the Walsh spectrum of g, since it has a 640-bit input. However, by representing the distribution of Walsh coefficients as a table like in Table 4, we can compute the exact distribution, which is actually quite sparse. For 80 S-Boxes, the table contains approximately 7.5 million non-zero coefficients.

To run Algorithm 1, we need to select the threshold M maximizing the success probability. Recall that it is the product pp', where $G := \#\{x, |\widehat{g}(x)| \geq M\}$, $p = \frac{M^2}{2^{2n}}G$ is the success in the first step (which we can detect) and $p' = \frac{G}{2^n}$ is the success in the second step. Since we know the entire Walsh spectrum of g, we select M to minimize $pp' = M^2G^2 2^{-3n}$:

$$M := 2^{326.23}, \quad G = 2^{610.60}, \quad p = 2^{-16.94}, \quad p' = 2^{-29.40}, \quad pp' = 2^{-46.34} \ .$$

These parameters will minimize the query complexity of the attack, however they might not be the best if we want to minimize the time complexity, as we will see below.

Quantum Arithmetic. Finally, we must design a quantum circuit that computes $\widehat{g}(x)$, compares $|\widehat{g}(x)|$ with M and computes $M/|\widehat{g}(x)|$. First, we notice that since all Walsh coefficients are divisible by 2^2, we can actually rescale everything by $(2^2)^{80} = 2^{160}$ and we save 160 bits of precision. Next, we compute the coefficients for the independent functions g_s in parallel, and we take their product. We perform the comparison with M, obtain the sign, and finish by a division with 640-bit precision.

We give only imprecise upper bounds for the cost of this circuit, using the results of [40] and [16] for quantum arithmetic operations: we estimate that an n-bit addition or comparison takes $2n - 1$ Toffoli gates and an n-bit product takes $\frac{3}{2}(n^2 + n)$ Toffoli gates.

The computation of each g_s requires a circuit with approximately $2^8 \times 8 \times 4$ Toffoli and CNOT gates that, for each i, compares its input with i, and writes the corresponding output value. We do this 80 times in parallel. Overall, this costs $\leq 2^{20}$ Toffoli gates. Afterwards, we take the product of all coefficients, two by two: the bit-length of the numbers that we multiply increases at each product. This step costs $\leq 2^{16}$ Toffoli gates.

The last step is to compute $M/|\widehat{g}(x)|$ to 640-bit precision. By taking the same cost as a product, we obtain $\leq 2^{20}$ Toffoli gates. In total, the overhead in Algorithm 1 with respect to the query of f can be upper bounded by 2^{22} Toffoli gates (with additional CNOT and NOT gates, that we did not count).

Recovering the Internal State: Step 1. When Algorithm 1 succeeds in both steps (rejection sampling and final measurement), we obtain the values for all hidden shifts $T_0, \ldots, T_4$, byte by byte, which we combine with the values of Equation 15

to create a system of equations:

$$\begin{cases} S[5] \oplus A(S[1]) \\ S[2] \oplus A(S[0] \oplus S[4]) \\ S[3] \oplus A(S[4]) \oplus A(S[7] \oplus A(S[0])) \\ S[3] \oplus S[7] \\ S[0] \oplus S[1] \oplus S[6] \oplus A(S[2]) \\ S[7] \\ S[7] \oplus A(S[0]) \oplus A(S[1] \oplus S[6]) \\ S[0] \oplus S[3] \oplus S[6] \oplus A(S[4] \oplus S[7] \oplus A(S[5])) \oplus A(S[3] \oplus A(S[4])) \end{cases}$$

The next step is to solve this system. First, we obtain directly $S[3]$ and $S[7]$. We then consider the smaller system:

$$\begin{cases} (E1) & S[5] \oplus A(S[1]) \\ (E2) & S[2] \oplus A(S[0] \oplus S[4]) \\ (E3) & A(S[4]) \oplus A(S[7] \oplus A(S[0])) \\ (E4) & S[0] \oplus S[1] \oplus S[6] \oplus A(S[2]) \\ (E5) & A(S[0]) \oplus A(S[1] \oplus S[6]) \\ (E6) & S[0] \oplus S[6] \oplus A(S[4] \oplus S[7] \oplus A(S[5])) \oplus A(S[3] \oplus A(S[4])) \end{cases}$$

Focusing on $(E2)$ to $(E5)$, we deduce the following, where $*$ are known values:

$$\begin{cases} S[2] \oplus A(S[0] \oplus S[4]) = * \\ A(S[4]) \oplus A(* \oplus A(S[0])) = * \\ A(S[0]) \oplus A(S[0] \oplus A(S[2]) \oplus *) = * \end{cases} \text{ i.e. } \begin{cases} S[2] \oplus A(S[0] \oplus S[4]) = * \\ \mathsf{SB}(S[4]) \oplus \mathsf{SB}(* \oplus A(S[0])) = * \\ \mathsf{SB}(S[0]) \oplus \mathsf{SB}(S[0] \oplus A(S[2]) \oplus *) = * \end{cases}$$

$$\implies \begin{cases} S[4] = \mathsf{SB}^{-1}\left(\mathsf{SB}(* \oplus A(S[0])) \oplus *\right) \\ S[2] = * \oplus A\left[S[0] \oplus S[4]\right] \\ \mathsf{SB}(S[0]) \oplus \mathsf{SB}\left[S[0] \oplus A(S[2]) \oplus *\right] = * \end{cases} \tag{17}$$

This sub-system in $S[0], S[2], S[4]$ admits on average one solution, and we can solve it in time 2^{96} by the following strategy:

- Guess two columns and two diagonals of $S[0]$. Obtain two columns of $S[4]$ by the first equation
- Deduce two columns of $S[0] \oplus S[4]$
- Obtain two columns and two diagonals of $A(S[2])$ by the third equation
- Deduce two diagonals of $S[2]$
- Using the second equation, solve the obtained linear system in the 2 remaining diagonals of $S[2]$; obtain the whole $S[2]$ (on average one solution)
- Using the third equation, obtain $S[0]$. Each S-Box equation of the form $S(* \oplus x) \oplus S(* \oplus x) = *$ has on average one solution; half of the time they have zero solutions and half of the time, they have two solutions. So, if one of these equations has no solution, we backtrack.

– Otherwise, we have found 2^{16} possibilities for $S[0]$. We use the first equation to compute $S[4]$ and we check that all equations are satisfied.

Though we need to examine 2^{16} solutions for $S[0]$, this will be done only 2^{96-16} times, so overall the time to solve the sub-system is 2^{96}. For each guess there are a few AES rounds to compute and 16 S-Box differential equations to solve.

Recovering the Internal State: Step 2. Having obtained $S[0], S[2], S[3], S[4], S[7]$, three equations remain:

$$\begin{cases} S[5] \oplus A(S[1]) = * \\ S[1] \oplus S[6] = * \\ S[6] \oplus A(* \oplus A(S[5])) = * \end{cases} \implies \begin{cases} S[5] \oplus A(S[1]) = * \\ S[1] \oplus A(* \oplus A(S[5])) = * \end{cases} \tag{18}$$

We solve this remaining sub-system as follows: we guess two diagonals of $S[5]$, which give two columns of $A(S[5])$, and two diagonals of $A(S[5])$, for a total of 12 bytes. By the first equation, we deduce two diagonals of $A(S[1])$. By the second, we deduce two columns of $S[1]$. This gives us a linear system in the 4 remaining bytes of $S[1]$, which we solve. Afterwards, we check both equations. The time complexity of this step is therefore slightly smaller than the first one, since it does not require to solve S-Box differential equations.

Summary: Hybrid Attack. So far we are using a classical algorithm for the state-recovery part. With the selection of M that minimizes the number of superposition queries, the adversary queries its oracle for Rocca a total of $2^{46.34}$ times on average. After each query, they perform the amplitude product, costing 2^{22} Toffoli gates, and succeed in the first step $2^{29.40}$ times on average. For each of these successes, they retrieve a candidate value for the hidden shift and solve the equation system. Once the system is solved, the candidate internal state can be tested by computing backwards a few rounds and checking the ciphertexts.

Overall, this first hybrid attack costs $2^{46.34}$ superposition queries, $2^{96+29.4} = 2^{125.4}$ classical time to solve the system, and $2^{68.34}$ additional Toffoli gates.

Quantum Attack. We can accelerate the attack by using quantum search to speedup the system solving. To solve the first subsystem in $S[0], S[2], S[4]$, we proceed as follows: we create a quantum algorithm that samples a valid $S[0]$, i.e., a value of $S[0]$ that passes the S-Box differential equation, then tests if one of the 2^{16} possibilities solves the entire system. We assume that solving a handful (16) S-Box differential equations costs no more than computing a handful (around 10) AES rounds. Then this algorithm is a sequence of two Grover searches with Toffoli count:

$$\left(\frac{\pi}{2}2^{16/2} + \frac{\pi}{2}2^{16/2}\right) 10 \times 2^{10} \simeq 2^{23} \ .$$

On the output of this algorithm, we use amplitude amplification [10]. By design, the probability that one of the possibilities for $S[0]$ solves the system is

2^{16-96}, so there are around $\frac{\pi}{4}2^{(96-16)/2}$ iterates to make, and the total time is:

$$\frac{\pi}{2}2^{(96-16)/2} \times 2^{23} \simeq 2^{64} \quad .$$

At this point, our quantum attack requires $2^{46.34}$ superposition queries and $2^{64+29.4} = 2^{93.4}$ Toffoli gates. We can optimize this by noticing how the (average) Toffoli count depends on the probabilities p and p' to succeed in both steps of Algorithm 1:

$$\frac{1}{p'}\left[\frac{1}{p}\left(2^{22}\right) + 2^{64}\right] \quad . \tag{19}$$

By solving exactly this minimization problem (since we know the entire distribution of the Walsh coefficients), we adopt:

$$M = 2^{306.52}, \quad G = 2^{624.97}, \quad p = \frac{M^2}{2^{2n}}G = 2^{-42.00}, \quad p' = \frac{G}{2^n} = 2^{-15.03} \tag{20}$$

which gives a complexity of $1/(pp') = 2^{57.03}$ Q2 encryption queries and

$$2^{15.03}\left(2^{22+42} + 2^{64}\right) \simeq 2^{80}$$

Toffoli gates. If we count that Q2 queries should have at least the same Toffoli cost as quantum implementations of Rocca, the gate count is comparable to the generic forgery attack in 2^{64} Q2 queries, though we do not require decryption queries anymore.

4.2 State-recovery on Rocca-S

The attack on Rocca-S is very similar to the one on Rocca. We have the same strategy: combine Algorithm 1 with linear post-processing to recover enough information on the internal state with a single query, and obtain this internal state by solving a simple system of equations.

Starting from an internal state S, we encrypt several message blocks and focus on the following outputs:

$$\begin{aligned}
C_0 &= M_0 \oplus S[0] \oplus A\left(S[3] \oplus S[5]\right) \\
C'_0 &= M'_0 \oplus S[2] \oplus A\left(S[4] \oplus S[6]\right) \\
C_1 &= M_1 \oplus S[1] \oplus S[6] \oplus A\left(S[3] \oplus S[6] \oplus A\left(S[2]\right) \oplus A\left(S[4]\right)\right) \\
C'_1 &= M'_1 \oplus S[0] \oplus A\left(M'_0 \oplus S[4] \oplus A\left(S[3]\right) \oplus A\left(S[5]\right)\right) \oplus A\left(S[1]\right) \\
C'_2 &= M'_2 \oplus S[1] \oplus S[6] \oplus A\left(M_0 \oplus A\left(S[0]\right)\right) \oplus \\
&\quad A\left(M'_0 \oplus M'_1 \oplus A\left(S[3] \oplus A\left(S[4]\right)\right) \oplus A\left(S[3]\right) \oplus A\left(S[6] \oplus A\left(S[2]\right)\right)\right) \\
C'_3 &= M'_3 \oplus M_0 \oplus S[4] \oplus A\left(M_1 \oplus A\left(S[1] \oplus S[6]\right)\right) \oplus A\left(S[0]\right) \oplus A\left(S[5]\right) \oplus \\
&\quad A\Big[M'_1 \oplus M'_2 \oplus A\left(S[4] \oplus A\left(S[0] \oplus A\left(S[1]\right)\right) \oplus A\left(S[5]\right)\right) \oplus \\
&\quad\quad A\left(S[6] \oplus A\left(M'_0 \oplus A\left(S[3]\right)\right) \oplus A\left(S[2]\right)\right) \oplus A\left(S[6] \oplus A\left(S[2]\right)\right)\Big]
\end{aligned}$$

Similarly as before, we set 3 input variables X_0, X_1, X_2 such that:

$$\begin{aligned} &M'_0 = 0, && M'_1 \oplus M'_2 = 0 \\ &X_0 = M_0, && X_1 = M'_0 \oplus M'_1, && X_2 = M_1 \ , \end{aligned}$$

and the other plaintext blocks are fixed to 0. With this input, the expression of $C_0 \oplus M_0, C'_0 \oplus M'_0, C_1 \oplus M_1$ and $C'_1 \oplus M'_1$ becomes constant, so we can immediately retrieve 4 expressions of the state S:

$$\begin{cases} S[0] \oplus A\left(S[3] \oplus S[5]\right) \\ S[2] \oplus A\left(S[4] \oplus S[6]\right) \\ S[1] \oplus S[6] \oplus A\left(S[3] \oplus S[6] \oplus A\left(S[2]\right) \oplus A\left(S[4]\right)\right) \\ S[0] \oplus A\left(S[4] \oplus A\left(S[3]\right) \oplus A\left(S[5]\right)\right) \oplus A\left(S[1]\right) \end{cases} \tag{21}$$

Next, we run our attack. We define a similar periodic function, except that it only combines $3 \times 16 = 48$ S-Boxes instead of 80. This reduces somewhat the gate count overhead for arithmetic operations, which we can still upper bound at 2^{22} Toffoli gates. More importantly, it modifies the values of M, G, p and p'.

When Algorithm 1 succeeds in both steps, we obtain the value of the hidden shift in addition to the 4 state words already recovered, i.e., $A(S[0])$ (for the variable X_0), $A\left(S[3] \oplus A\left(S[4]\right)\right) \oplus A\left(S[3]\right) \oplus A\left(S[6] \oplus A\left(S[2]\right)\right)$ (for the variable X_1), $A\left(S[1] \oplus S[6]\right)$ (for the variable X_2). Using the knowledge of $S[0]$, we can simplify the system we have to solve into:

$$\begin{cases} S[0] \\ S[3] \oplus S[5] \\ S[2] \oplus A\left(S[4] \oplus S[6]\right) \\ S[3] \oplus S[6] \oplus A\left(S[2]\right) \oplus A\left(S[4]\right) \\ A\left(S[4] \oplus A\left(S[3]\right) \oplus A\left(S[5]\right)\right) \oplus A\left(S[1]\right) \\ A\left(S[3] \oplus A\left(S[4]\right)\right) \oplus A\left(S[3]\right) \oplus A\left(S[6] \oplus A\left(S[2]\right)\right) \\ S[1] \oplus S[6] \end{cases}$$

We will solve this system in about 2^{64} quantum (or 2^{128} classical) computations. First, we guess $S[5]$ and deduce $S[3]$. We use the fourth and sixth equations of above, where $*$ denotes a known value:

$$\begin{cases} S[6] \oplus A(S[2]) \oplus A(S[4]) = * \\ A(* \oplus A(S[4])) \oplus A(S[6] \oplus A(S[2])) = * \end{cases} \tag{22}$$

which implies $A(* \oplus A(S[4])) \oplus A(* \oplus A(S[4])) = *$. We can compose by the inverse of the AES linear layer to retrieve 16 independent differential S-Box equations of the form: $S(* \oplus x) \oplus S(* \oplus x) = *$, where the variable to recover is $A(S[4])$ (byte by byte). Like before, these equations have two solutions half of the time, and otherwise zero. So we need to try on average 2^{16} values of $S[3]$ until all 16 equations have solutions, and in that case, we need to check the 2^{16} different

obtained values of $A(S[4])$. We will deduce the whole internal state and check the equations.

Again, we assume that checking $S[3]$ or checking a solution costs about 10×2^{10} Toffoli gates, or a full AES-128. We use amplitude amplification [10] over an algorithm that: finds a valid $S[3]$, then, searches through the corresponding solutions using a Grover search. We will find the internal state in time:

$$\frac{\pi}{2}2^{(128-16)/2}\left(\frac{\pi}{2}2^{16/2}+\frac{\pi}{2}2^{16/2}\right)10\times 2^{10}\simeq 2^{80}\ .$$

Summary of the Attack and Optimization. We propose two optimizations of this attack: one that minimizes the number of Q2 queries (i.e., maximizes pp'), and one that minimizes the total time complexity. In the first case, we set:

$$M=2^{195.40},\quad G=2^{365.62},\quad p=2^{-11.58},\quad p'=2^{-18.37},\quad pp'=2^{-29.95}\ . \tag{23}$$

The adversary queries its oracle for Rocca-S a total of $2^{29.95} \simeq 2^{30}$ times on average before encountering a success. The quantum arithmetic requires an additional time of $2^{29.95} \times 2^{22} \simeq 2^{52}$ Toffoli gates. Solving the equation system happens only if the first step (amplitude product) succeeded, so $2^{18.37}$ times. We can solve it quantumly, for a total Toffoli count $2^{18.37} \times 2^{80} \simeq 2^{98}$.

However, the average Toffoli count can be expressed as:

$$\frac{1}{p'}\left[\frac{1}{p}(2^{22})+2^{80}\right] \tag{24}$$

By minimizing this expression instead, we obtain the following choice:

$$M=2^{165.53},\quad G=2^{378.95},\quad p=2^{-58.00},\quad p'=2^{-5.05},\quad pp'=2^{-63.05}\ . \tag{25}$$

which gives a complexity of $\simeq 2^{63}$ Q2 encryption queries and $\simeq 2^{85}$ Toffoli gates.

4.3 Key-recovery on Tiaoxin

Our method allows to recover the state T_3 at some point of the encryption phase. Afterwards, we can invert the round function on T_3, and the initialization phase, and recover the key which was loaded in the initial state.

Let us fix the state $T_3[0,1,2]$ at the beginning of the encryption phase and unroll a few ciphertexts:

$$\begin{aligned}
C_0 &= M_0 \oplus T_3[0] \oplus T_3[1] \oplus A(T_3[2]) \oplus A(T_4[0]) \oplus \mathrm{AND}(T_6[2], T_4[2])\\
C_0' &= M_0 \oplus M_0' \oplus T_4[1] \oplus T_6[0] \oplus A(T_3[0]) \oplus A(T_6[5]) \oplus \mathrm{AND}(T_6[4], T_3[1])\\
C_1' &= M_0 \oplus M_1 \oplus M_0' \oplus M_1' \oplus T_6[0] \oplus A(M_0 \oplus T_3[0] \oplus A(T_3[2])) \oplus\\
&\quad A(T_6[4]) \oplus A(T_6[5]) \oplus A(T_4[0]) \oplus \mathrm{AND}(T_6[3], A(T_3[0]))\\
C_3' &= M_0 \oplus M_1 \oplus M_2 \oplus M_3 \oplus M_0' \oplus M_1' \oplus M_2' \oplus M_3' \oplus\\
&\quad T_6[0] \oplus A(T_6[3]) \oplus A(T_6[4]) \oplus A(T_6[5]) \oplus A(T_6[2])
\end{aligned}$$

$$A\big[M_0 \oplus M_1 \oplus M_2 \oplus T_3[0] \oplus A(T_3[1]) \oplus A(T_3[2]) \oplus A(A(T_3[0]))\big] \oplus$$
$$A\big[M'_0 \oplus M'_1 \oplus T_4[0] \oplus A(T_4[2]) \oplus A(T_4[3])\big] \oplus$$
$$\mathrm{AND}(T_6[1], A(M_0 \oplus M_1 \oplus T_3[0] \oplus A(T_3[1]) \oplus A(T_3[2])))$$

We set the following as variables: $X_0 = M_0, X_1 = M_0 \oplus M_1, X_2 = M_0 \oplus M_1 \oplus M_2$. The rest is fixed. We focus on C'_1 and C'_3, and define the shift values:

$$\begin{cases} R_0 := T_3[0] \oplus A(T_3[2]) \\ R_1 := T_3[0] \oplus A(T_3[1]) \oplus A(T_3[2]) \\ R_2 := T_3[0] \oplus A(T_3[1]) \oplus A(T_3[2]) \oplus A(A(T_3[0])) \end{cases} \tag{26}$$

We then observe the XOR of C'_1 and C'_3. More precisely, let L be the function that selects one bit in each column of the state and XORs them. We assume that on these 4 bits, $T_6[1] = 1$. Then we have:

$$L\left[\mathrm{AND}(T_6[1], A(M_0 \oplus M_1 \oplus T_3[0] \oplus A(T_3[1]) \oplus A(T_3[2])))\right]$$
$$= L \circ A(M_0 \oplus M_1 \oplus T_3[0] \oplus A(T_3[1]) \oplus A(T_3[2])) \ ,$$

and we define the function:

$$F(C'_1 \oplus M_0 \oplus M_1 \oplus M'_0 \oplus M'_1, C'_3 \oplus M_0 \oplus M_1 \oplus M_2 \oplus M_3 \oplus M'_0 \oplus M'_1 \oplus M'_2 \oplus M'_3)$$
$$= b \oplus L \circ A(M_0 \oplus R_0) \oplus L \circ A(M_0 \oplus M_1 \oplus R_1) \oplus L \circ A(M_0 \oplus M_1 \oplus M_2 \oplus R_2) \ ,$$

where b is an unknown bit depending on T. We have $L \circ A = (L \circ \mathsf{MC} \circ \mathsf{SR}) \circ \mathsf{SB}$, so column by column, we have a one-bit function of the S-Box outputs, which can be rewritten as: $(x_0, x_1, x_2, x_3) \mapsto \bigoplus_i \alpha_i \cdot \mathsf{SB}(x_i)$ for well-chosen masks $\alpha_0, \ldots, \alpha_3$.

The situation is thus the same as before, since the distribution of Walsh coefficients is independent of the mask α_i (as long as it's nonzero). Recovering the entire state $T_3[0, 1, 2]$ from the shifts R_0, R_1, R_2 is trivial and costs only a few AES rounds. Afterwards, we compute backwards through the 15 rounds of initialization on the state T_3 (30 AES rounds), and obtain a candidate key K that we can immediately check. All of this can be done classically.

Since there are $16 \times 3 = 48$ S-Boxes in the function, we optimize the probability similarly to Rocca-S and obtain $pp' \simeq 2^{-30}$. The Toffoli cost of the entire attack is roughly $2^{30} \times 2^{22} = 2^{52}$ and it contains 2^{30} Q2 queries. Note that we need to multiply these numbers by 2^4, since we assumed to have guessed correctly 4 bits of $T_6[1]$.

4.4 State-recovery on AEGIS-128L

Contrary to the rest of this section, the attack on AEGIS-128L uses a function of smaller correlation, making our analysis dependent on Heuristic 1.

Starting from an initial state S, we encrypt pairs of message blocks (M_i, M'_i) with $M'_i = 0$. To simplify the notations, we will express the ciphertext blocks in function of T, the state after one update, that is, $T[i] = S[i] \oplus A(S[i-1])$. Our

aim is to recover T. As they cannot be expressed from T, we ignore the first pair of ciphertext blocks and focus on:

$$
\begin{aligned}
C_1 &= M_1 \oplus T[1] \oplus T[6] \oplus \mathrm{AND}\left(T[2], T[3]\right) \\
C_1' &= M_1' \oplus T[2] \oplus T[5] \oplus \mathrm{AND}\left(T[6], T[7]\right) \\
C_2' &= M_2' \oplus T[2] \oplus T[5] \oplus A\left(M_0' \oplus T[4]\right) \oplus A\left(T[1]\right) \\
&\quad \oplus \mathrm{AND}\left(T[6] \oplus A\left(T[5]\right), T[7] \oplus A\left(T[6]\right)\right) \\
C_6 &= M_6 \oplus A\Big[M_0 \oplus M_1 \oplus M_2 \oplus M_3 \oplus M_4 \oplus T[0] \oplus A\left(T[7] \oplus A\left(T[6]\right)\right) \\
&\qquad \oplus A\left(T[7]\right) \oplus A\left(T[7] \oplus A\left(T[6] \oplus A\left(T[5]\right)\right) \oplus A\left(T[6]\right)\right) \\
&\qquad \oplus A\Big(T[7] \oplus A\left(T[6] \oplus A\left(T[5] \oplus A\left(M_0' \oplus T[4]\right)\right) \oplus A\left(T[5]\right)\right) \\
&\qquad\quad \oplus A\left(T[6] \oplus A\left(T[5]\right)\right) \oplus A\left(T[6]\right)\Big)\Big] \\
&\quad \oplus A\Big[M_0 \oplus M_1 \oplus M_2 \oplus M_3 \oplus T[0] \oplus A\left(T[7] \oplus A\left(T[6]\right)\right) \oplus A\left(T[7]\right) \\
&\qquad \oplus A\left(T[7] \oplus A\left(T[6] \oplus A\left(T[5]\right)\right) \oplus A\left(T[6]\right)\right)\Big] \\
&\quad \oplus A\Big[M_0 \oplus M_1 \oplus M_2 \oplus T[0] \oplus A\left(T[7] \oplus A\left(T[6]\right)\right) \oplus A\left(T[7]\right)\Big] \\
&\quad \oplus A\Big[M_0 \oplus M_1 \oplus T[0] \oplus A\left(T[7]\right)\Big] \\
&\quad \oplus A\left(M_0 \oplus T[0]\right) \oplus Y \oplus \mathrm{AND}(h'(M_0, M_1, M_2, M_3), h''(M_0, M_1, M_2)) \ ,
\end{aligned}
$$

where h' and h'' are two functions whose exact expression is irrelevant here, and Y is a constant (an expression in which only T and M_i' intervene). Similarly to Tiaoxin, we use a linear post-processing which truncates C_6 to only 4 bits. Therefore, though it is completely unknown (and depends on the unknown state T), the AND term will become a function h with correlation 2^{-4}. By making M_0 to M_4 vary, we obtain 5 shifts which give us:

$T[0]$	from the M_0 shift
$T[7]$	from the M_1 shift, knowing $T[0]$
$T[6]$	from the M_2 shift, knowing $T[0, 7]$
$T[5]$	from the M_3 shift, knowing $T[0, 6, 7]$
$T[4]$	from the M_4 shift, knowing $T[0, 5, 6, 7]$

Next, we focus on the ciphertext blocks C_1 to C_2' which we have also obtained. Thanks to C_1' and all the state registers that we know, we obtain $T[2]$. Next, thanks to C_2', we obtain $T[1]$. The only register of T that we are missing is $T[3]$. We can find half of it using the expression of C_1: indeed, from the known $T[i]$ we can compute $\mathrm{AND}(T[2], T[3])$. We can expect half the bits of $T[2]$ to be one, which gives us the the corresponding bits of $T[3]$.

State-recovery Attack. After performing this partial state-recovery, we can do a Grover search on the remaining 64 bits of the state. However, checking if we

have obtained a valid internal state is not trivial. Indeed, the round function of AEGIS is not invertible, so we cannot compute backwards and check with previous ciphertexts. In fact, there do not seem to be other ciphertext equations that we can exploit (either we have already used them, either they depend on the varying M_i).

Consequently, we do a Grover search using *superposition decryption queries* to test our guess of the state. That is, starting from the recovered internal state, we compute the tag (approximately 6×8 AES rounds, i.e., $\leq 2^{16}$ Toffoli gates) and we try to decipher with an oracle. If the internal state is guessed correctly, the oracle will accept. This operation requires approximately: $\frac{\pi}{2} 2^{32} \times 2^{16} \leq 2^{49}$ gates and 2^{33} decryption queries.

Since the number of S-Boxes is the same as in Rocca, the hidden shift algorithm is actually the same. We keep the same p and p', but introduce the correlation $c = 2^{-4}$. The Toffoli count and the number of queries are respectively:

$$\frac{2}{p'}\left(\frac{1}{p}\frac{1}{c^2}2^{22} + 2^{49}\right) \qquad \text{and} \qquad \frac{2}{p'}\left(\frac{1}{p}\frac{1}{c^2} + 2^{33}\right) \ . \tag{27}$$

If we optimize the Toffoli count, we get the following parameters:

$$M = 2^{324.23}, G = 2^{612.54}, p = 2^{-19.00}, p' = 2^{-27.46} \ , \tag{28}$$

which give $2^{77.46}$ Toffolis, smaller than the cost of Grover search (2^{81}), and $\frac{2\times 2^{33}}{p'} \leq 2^{62}$ decryption queries, which is also smaller than a forgery attack using a Grover search. We also use $\frac{2}{pp'c^2} \leq 2^{56}$ encryption queries.

Remark 3. Our attack benefits from parallelization, since all $\frac{2}{p'}$ instances of the main loops can be run in parallel. For comparison, running a Grover search in parallel to reduce the wall-clock time by a factor S increases its total computational cost by the same factor S.

5 Discussion

In all instances of our attack, the AE scheme (Rocca, Rocca-S, Tiaoxin, AEGIS) is believed to be secure regarding guess-and-determine attacks that aim at recovering the state. Indeed, when one only observes the ciphertext blocks, the obtained system of equations is intractable.

Our quantum attack works because we can observe hidden shifts in addition to the ciphertexts. This allows us to reduce the state-recovery to a simpler system of equations (the simplest being Tiaoxin-346 which only relies on three shifts). However, there are limitations to this approach. Notably, if we have a ciphertext $C = S_0 \oplus A(S_1 \oplus M_1)$, we have only two choices: either make $M_1 = 0$ a constant, and observe $S_0 \oplus A(S_1)$, or make M_1 a variable, and observe S_1. In the latter case, S_0 is lost. Besides, we can only use one variable for one shift, i.e., if we have $C = A(S_1 \oplus M_1)$ and $C' = A(S_2 \oplus M_1)$, we must drop one of the ciphertext blocks. Another problematic case is when we observe $A(S_0 \oplus A(S_1 \oplus M_1))$. Though we

do have a shifted function, the function is now unknown (it depends on S_0) and more complex (two rounds of AES instead of one). The attack can proceed by guessing enough bits of S_0, but becomes more difficult.

In our examples, the choice of the shifts was done by hand, trying to obtain the simplest equation system. More clever choices might still exist. Conversely, making the schemes secure against this attack means ensuring that none of the equation systems resulting from a combination of ciphertexts and shifts can be tractable.

Previously, some Q2 quantum attacks have given rise to more efficient Q1 attacks [6]. However such methods do not appear to work in this scenario, as classical queries will have different nonces, and cannot be brought together to emulate a single quantum query. Therefore, all the schemes studied in this paper remain secure against Q1 attacks.

Acknowledgments. The authors would like to thank Ravi Anand, Takanori Isobe and Yuto Nakano for helpful comments on a preliminary version of this paper. This work has been partially supported by the French Agence Nationale de la Recherche through the DeCrypt project under Contract ANR-18-CE39-0007 and the OREO project under Contract ANR-22-CE39-0015, and through the France 2030 program under grant agreement No. ANR-22-PETQ-0007 EPiQ and ANR-22-PETQ-0008 PQ-TLS.

References

1. Alagic, G., Bai, C., Katz, J., Majenz, C.: Post-quantum security of the even-mansour cipher. In: EUROCRYPT (3). Lecture Notes in Computer Science, vol. 13277, pp. 458–487. Springer (2022). https://doi.org/10.1007/978-3-031-07082-2_17
2. 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). https://doi.org/10.1007/978-3-319-29360-8_4
3. Anand, R., Isobe, T.: Quantum security analysis of rocca. Quantum Information Processing **22**(4), 164 (2023). https://doi.org/10.1007/s11128-023-03908-3
4. Bhaumik, R., Bonnetain, X., Chailloux, A., Leurent, G., Naya-Plasencia, M., Schrottenloher, A., Seurin, Y.: QCB: efficient quantum-secure authenticated encryption. In: ASIACRYPT (1). Lecture Notes in Computer Science, vol. 13090, pp. 668–698. Springer (2021). https://doi.org/10.1007/978-3-030-92062-3_23
5. Bonnetain, X.: Quantum key-recovery on full AEZ. In: SAC. Lecture Notes in Computer Science, vol. 10719, pp. 394–406. Springer (2017). https://doi.org/10.1007/978-3-319-72565-9_20
6. 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). https://doi.org/10.1007/978-3-030-34578-5_20, https://doi.org/10.1007/978-3-030-34578-5_20

7. Bonnetain, X., Leurent, G., Naya-Plasencia, M., Schrottenloher, A.: Quantum linearization attacks. In: ASIACRYPT (1). Lecture Notes in Computer Science, vol. 13090, pp. 422–452. Springer (2021). https://doi.org/10.1007/978-3-030-92062-3_15
8. Bonnetain, X., Naya-Plasencia, M., Schrottenloher, A.: Quantum security analysis of AES. IACR Trans. Symmetric Cryptol. **2019**(2), 55–93 (2019)
9. Bonnetain, X., Schrottenloher, A., Sibleyras, F.: Beyond quadratic speedups in quantum attacks on symmetric schemes. In: EUROCRYPT (3). Lecture Notes in Computer Science, vol. 13277, pp. 315–344. Springer (2022). https://doi.org/10.1007/978-3-031-07082-2_12
10. Brassard, G., Hoyer, P., Mosca, M., Tapp, A.: Quantum amplitude amplification and estimation. Contemporary Mathematics **305**, 53–74 (2002)
11. Brassard, G., Høyer, P., Tapp, A.: Quantum cryptanalysis of hash and claw-free functions. In: LATIN. Lecture Notes in Computer Science, vol. 1380, pp. 163–169. Springer (1998)
12. 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). https://doi.org/10.1007/978-3-030-65277-7_17
13. van Dam, W., Hallgren, S., Ip, L.: Quantum algorithms for some hidden shift problems. SIAM J. Comput. **36**(3), 763–778 (2006). https://doi.org/10.1137/S009753970343141X
14. Eichlseder, M., Nageler, M., Primas, R.: Analyzing the linear keystream biases in AEGIS. IACR Trans. Symmetric Cryptol. **2019**(4), 348–368 (2019). https://doi.org/10.13154/tosc.v2019.i4.348-368
15. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: STOC. pp. 212–219. ACM (1996)
16. Häner, T., Roetteler, M., Svore, K.M.: Optimizing quantum circuits for arithmetic. arXiv preprint arXiv:1805.12445 (2018)
17. Hosoyamada, A., Inoue, A., Ito, R., Iwata, T., Mimematsu, K., Sibleyras, F., Todo, Y.: Cryptanalysis of rocca and feasibility of its security claim. IACR Trans. Symmetric Cryptol. **2022**(3), 123–151 (2022). https://doi.org/10.46586/tosc.v2022.i3.123-151
18. 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). https://doi.org/10.1007/978-3-319-98113-0_21
19. Hosoyamada, A., Sasaki, Y.: Finding hash collisions with quantum computers by using differential trails with smaller probability than birthday bound. In: EUROCRYPT (2). Lecture Notes in Computer Science, vol. 12106, pp. 249–279. Springer (2020). https://doi.org/10.1007/978-3-030-45724-2_9
20. Jang, K., Baksi, A., Song, G., Kim, H., Seo, H., Chattopadhyay, A.: Quantum analysis of AES. IACR Cryptol. ePrint Arch. p. 683 (2022)
21. Kales, D., Eichlseder, M., Mendel, F.: Note on the robustness of CAESAR candidates. IACR Cryptol. ePrint Arch. p. 1137 (2017)
22. 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). https://doi.org/10.1007/978-3-662-53008-5_8

23. Kaplan, M., Leurent, G., Leverrier, A., Naya-Plasencia, M.: Quantum differential and linear cryptanalysis. IACR Trans. Symmetric Cryptol. **2016**(1), 71–94 (2016). https://doi.org/10.13154/tosc.v2016.i1.71-94
24. Kuwakado, H., Morii, M.: Quantum distinguisher between the 3-round feistel cipher and the random permutation. In: ISIT. pp. 2682–2685. IEEE (2010). https://doi.org/10.1109/ISIT.2010.5513654
25. Kuwakado, H., Morii, M.: Security on the quantum-type even-mansour cipher. In: ISITA. pp. 312–316. IEEE (2012)
26. 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). https://doi.org/10.1007/978-3-319-70697-9_6
27. Liu, F., Isobe, T., Meier, W., Sakamoto, K.: Weak keys in reduced AEGIS and tiaoxin. IACR Trans. Symmetric Cryptol. **2021**(2), 104–139 (2021). https://doi.org/10.46586/tosc.v2021.i2.104-139
28. Nakano, Y., Fukushima, K., Isobe, T.: Encryption algorithm rocca-s. IETF Draft Standard, https://www.ietf.org/archive/id/draft-nakano-rocca-s-03.html
29. National Institute of Standards and Technology: FIPS 197 advanced encryption standard (AES) (2001). https://doi.org/10.6028/NIST.FIPS.197
30. Nielsen, M.A., Chuang, I.: Quantum computation and quantum information (2002)
31. Nikolic, I.: Tiaoxin-346 (v2.1). Submission to the CAESAR competition (2016), https://competitions.cr.yp.to/round3/tiaoxinv21.pdf
32. Ozols, M., Roetteler, M., Roland, J.: Quantum rejection sampling. ACM Trans. Comput. Theory **5**(3), 11:1–11:33 (2013). https://doi.org/10.1145/2493252.2493256
33. Rötteler, M.: Quantum algorithms for highly non-linear boolean functions. In: SODA. pp. 448–457. SIAM (2010). https://doi.org/10.1137/1.9781611973075.37
34. Sakamoto, K., Liu, F., Nakano, Y., Kiyomoto, S., Isobe, T.: Rocca: An efficient aes-based encryption scheme for beyond 5G. IACR Trans. Symmetric Cryptol. **2021**(2), 1–30 (2021). https://doi.org/10.46586/tosc.v2021.i2.1-30
35. Sakamoto, K., Liu, F., Nakano, Y., Kiyomoto, S., Isobe, T.: Rocca: An efficient aes-based encryption scheme for beyond 5G (full version). IACR Cryptol. ePrint Arch. p. 116 (2022), https://eprint.iacr.org/2022/116
36. Sanders, Y.R., Low, G.H., Scherer, A., Berry, D.W.: Black-box quantum state preparation without arithmetic. Physical review letters **122**(2), 020502 (2019)
37. Schrottenloher, A.: Quantum linear key-recovery attacks using the QFT. In: CRYPTO (5). Lecture Notes in Computer Science, vol. 14085, pp. 258–291. Springer (2023). https://doi.org/10.1007/978-3-031-38554-4_9
38. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: FOCS. pp. 124–134. IEEE Computer Society (1994). https://doi.org/10.1109/SFCS.1994.365700
39. Simon, D.R.: On the power of quantum computation. SIAM J. Comput. **26**(5), 1474–1483 (1997). https://doi.org/10.1109/SFCS.1994.365701
40. Takahashi, Y., Tani, S., Kunihiro, N.: Quantum addition circuits and unbounded fan-out. arXiv preprint arXiv:0910.2530 (2009)
41. Wu, H., Preneel, B.: AEGIS: A fast authenticated encryption algorithm. In: Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 8282, pp. 185–201. Springer (2013). https://doi.org/10.1007/978-3-662-43414-7_10
42. Wu, H., Preneel, B.: AEGIS: A fast authenticated encryption algorithm (full version). IACR Cryptol. ePrint Arch. p. 695 (2013), http://eprint.iacr.org/2013/695

43. Wu, H., Preneel, B.: AEGIS: a fast authenticated encryption algorithm (v1.1). Submission to the CAESAR competition (2016), https://competitions.cr.yp.to/round3/aegisv11.pdf