Quantum Boomerang Attacks and Some Applications

Paul Frixons, María Naya-Plasencia, André Schrottenloher

Quantum Boomerang Attacks and Some Applications

Paul Frixons[1,2], María Naya-Plasencia[2], and André Schrottenloher[3]

[1] Orange Labs, Caen, France
[2] Inria, Paris, France (firstname.lastname@inria.fr)
[3] Cryptology Group, CWI, Amsterdam, The Netherlands
(firstname.lastname@m4x.org)

Abstract. In this paper, we study quantum key-recovery attacks on block ciphers. While it is well known that a quantum adversary can generically speed up an exhaustive search of the key, much less is known on how to use specific vulnerabilities of the cipher to accelerate this procedure. In this context, we show how to convert classical boomerang and mixing boomerang attacks into efficient quantum key-recovery attacks. In some cases, we can even obtain a quadratic speedup, the same as simple differential attacks. We apply this technique to a 5-round attack on SAFER++.

Keywords: boomerang attack, post-quantum security, mixing boomerang attack, SAFER++, AES

1 Introduction

In symmetric cryptography, cryptanalysis is the base of the confidence we have in the primitives we use. In order to consider a primitive secure, we need to constantly evaluate it with respect to all known attack families. One of the most well known families of attacks is boomerang attacks, introduced by Wagner in [29]. They are a particular type of differential attacks that, instead of considering a long differential trail in the primitive (a propagation of differences from the plaintext through the ciphertext), combine several short ones that have high individual probabilities. While differential attacks usually consider pairs of plaintexts having a certain difference, boomerang attacks use quartets instead. They have shown to be effective (or the most effective known attacks) against several primitives like [16,3], and have seen many improvements, like recently [11].

The quantum security of symmetric primitives has attracted an increasing interest in the last few years. Some works have targeted the security of generic constructions, for example [21,22,20], while others have studied the security of actual designs [7,19,5]. Many have proposed quantum versions of popular classical attacks, like for instance [20,6].

In this paper, as in most of these previous works, we consider key-recovery attacks in the single secret-key setting. Here it is well-known that Grover's quantum search algorithm [18] can be used to quadratically speed up the exhaustive

search of the key. However, when the design under study admits some vulnerability (e.g., a classical attack exists), a faster quantum key-recovery procedure might exist. Designing such procedures from classical design patterns is often a non-trivial task. It is known that if a classical attack can be represented as a sequence of nested exhaustive searches, then it can be converted into a sequence of quantum searches: this framework was applied in [7] to Square [12,17] and DS Meet-in-the-Middle attacks [13,14]. But not all attacks can be covered this way, and many of them still seem difficult to translate.

Boomerang Attacks. In this paper, we consider Boomerang attacks, which form a particular type of differential attacks introduced by Wagner in [29]. We study how to build an efficient quantum version of boomerang attacks and of mixing boomerang attacks [15] recently introduced in the context of AES. We propose, for the first time, efficient *quantum boomerang attacks*, and we apply them to several reduced-round versions of well known ciphers.

The quantum attacks studied in this paper are also based on quantum search. One should note that, since quantum search always provides a quadratic speedup, our procedures admit a quadratic speedup at best. By comparing with the quadratic speedup of exhaustive key search, it follows that we will not be able to attack more rounds than in the classical setting: any quantum attack in our framework can be converted back into a valid classical attack. Though this result can seem rather negative at first sight, our new attacks, like previous works with similar conclusions [7,20], tend to show that block ciphers should be assumed to retain half of their bits of classical security against quantum adversaries, even when this security has been reduced with respect to the generic key search.

Outline. In Section 2, we give a broad introduction to the boomerang attacks we will quantize later and detail the quantum tools that we use within this paper. In Section 3, we present quantum algorithms for boomerang distinguishers, last-round attacks and mixing boomerang attacks. In Section 4, we show an attack on 5 rounds of the block cipher SAFER++ (where the best classical attack reaches 5.5 rounds). In Section 5 we study a related-key attack against AES-256, with more details given in Appendix B. We conclude the paper in Section 6.

2 Preliminaries

In this section, we introduce the classical Boomerang attack from [29] and the Mixing Boomerang attack from [15]. We give generic formulas for their time complexity depending on the parameters of the cipher attacked. Next, we introduce our quantum tools. We stress that knowledge of quantum computing is not required to understand our work, as we will use standard quantum algorithms (such as Grover search) as black-box components, and design our attacks from a rather abstract perspective.

Throughout this paper, we consider an n-bit block cipher E, with unknown key k. Standard block ciphers (SPNs or Feistel schemes) are built by iterating

a round function, and we will sometimes decompose E into subciphers, e.g., $E = E_2 \circ E_1$ where E_1 forms the r_1 first rounds and E_2 the r_2 last rounds.

Boomerang cryptanalysis is a subset of differential cryptanalysis, which studies the propagation of differences in a cipher. Given a cipher (or subcipher) E, and a pair of differences $(\Delta_{\text{in}}, \Delta_{\text{out}})$, we say that $\Delta_{\text{in}} \to \Delta_{\text{out}}$ is a *differential* for E of probability:

$$\Pr\big(\Delta_{\text{in}} \xrightarrow{E} \Delta_{\text{out}}\big) = \Pr_X(E(X \oplus \Delta_{\text{in}}) \oplus E(X) = \Delta_{\text{out}}) \quad .$$

Since E is a keyed function, the probability of a differential depends on the choice of key. In the analysis, it is usually computed on average over the key. When running an attack, we consider a black box with a fixed given key. We then assume that the differential probability is equal to the analyzed average (even if there is a small variation in practice, we may run the attack again with another estimate – this is valid for classical as well as quantum attacks).

2.1 The Classical Boomerang Attack

We briefly describe the *boomerang distinguisher* introduced in [29] and the last-round key-recovery attack that can be based on it. The notation that we introduce here $(E, p, q, \alpha, \ldots)$ will be kept throughout the paper.

Boomerang Distinguisher. As above, let E be a block cipher of block size n and key size k, that can be decomposed into: $E = E_2 \circ E_1$. We assume that each part has a high-probability differential: $\alpha \to \beta$ for E_1 and $\delta \to \gamma$ for E_2^{-1}.

$$\Pr\big(\alpha \xrightarrow{E_1} \beta\big) = p_\downarrow, \quad \Pr\big(\beta \xrightarrow{E_1^{-1}} \alpha\big) = p_\uparrow, \quad \Pr\big(\delta \xrightarrow{E_2^{-1}} \gamma\big) = q \quad .$$

The distinguisher is given in Algorithm 1. It uses $\frac{4}{p_\downarrow p_\uparrow q^2}$ encryptions, decryptions, and negligible memory. We can check its correctness as follows (see also Figure 1 for the notations):

Step 3: using the differential on E_1; with probability $p_\downarrow$, $E_1(P_1) \oplus E_1(P_2) = \beta = E_2^{-1}(C_1) \oplus E_2^{-1}(C_2)$

Step 4: using the differential on E_2^{-1}; with probability q^2, $E_2^{-1}(C_3) \oplus E_2^{-1}(C_1) = \gamma$ and $E_2^{-1}(C_4) \oplus E_2^{-1}(C_2) = \gamma$. Thus, by summing these equations: $E_2^{-1}(C_3) \oplus E_2^{-1}(C_4) = E_2^{-1}(C_1) \oplus E_2^{-1}(C_2) = \beta$.

Step 5: since we have established $E_2^{-1}(C_1) \oplus E_2^{-1}(C_2) = \beta$ with probability $p_\downarrow q^2$, it remains to satisfy the differential on E_1^{-1}, and we obtain $P_3 \oplus P_4 = \alpha$ with probability $p_\uparrow \times p_\downarrow q^2$.

Then, the full path is satisfied with probability $(p_\downarrow p_\uparrow q)^2$ instead of 2^{-n} for a random permutation, and making $1/(p_\downarrow p_\uparrow q)^2$ trials is enough to determine the case. Usually, we also have $p_\downarrow = p_\uparrow = p$ and this formula simplifies into $(pq)^2$.

Algorithm 1 Boomerang Distinguisher

Input: oracle access to $E = E_2 \circ E_1$, and its inverse (or a random permutation)
Output: "E" or "Random"

Repeat $(p_\uparrow p_\downarrow q)^{-2}$ **times** ▷ (probability of success of $\frac{1}{2}$)
  Select P_1 at random and set $P_2 = P_1 \oplus \alpha$
  Encrypt: $C_1 = E(P_1)$ and $C_2 = E(P_2)$
  Compute: $C_3 = C_1 \oplus \delta$ and $C_4 = C_2 \oplus \delta$
  Decrypt: $P_3 = E^{-1}(C_3)$ and $P_4 = E^{-1}(C_4)$
  if $P_4 = P_3 \oplus \alpha$ **then return** "E"
EndRepeat
return "Random"

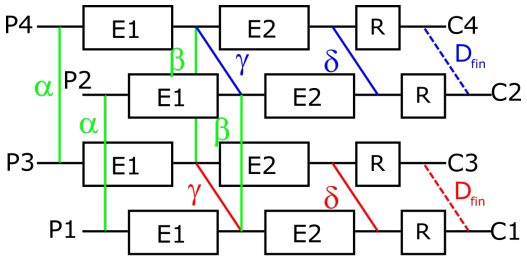

Fig. 1. Last-round key recovery

Key Recovery on the Last Round. We now explain how to build a key-recovery attack using this boomerang distinguisher.

We append one (or more) additional rounds R to the cipher, which is now $E = R \circ E_2 \circ E_1$ (if the additional round is at the beginning, consider E^{-1}). Let D_{fin} denote the set of differences that can be obtained from δ after the additional rounds, p_{out} the probability that we get δ back from an element in D_{fin} by computing the last rounds backwards, and k_{out} the number of key bits involved in these additional rounds. We need D_{fin} to be a vector space. This is usually the case as most of the time, non-zero components of δ become unknown after a passage through an S-Box. This property is used to make data *structures* of the form $\{X \oplus \Delta / \Delta \in D_{\text{fin}}\}$, from which we can extract quadratically many pairs with differences in D_{fin}: $\forall \Delta_1, \Delta_2, (X \oplus \Delta_1) \oplus (X \oplus \Delta_2) = \Delta_1 \oplus \Delta_2 \in D_{\text{fin}}$.

To simplify for now, we assume that $\frac{1}{\sqrt{p_{\text{out}}}pq} < |D_{\text{fin}}|$, in which case a single structure is needed. Otherwise when $\frac{1}{\sqrt{p_{\text{out}}}pq} \geq |D_{\text{fin}}|$, we will actually generate multiple structures with $|D_{\text{fin}}|$ ciphertexts each, so that they form a total of $\frac{1}{p_{\text{out}}(pq)^2}$ pairs.

Algorithm 2 detects a good guess of the k_{out} key bits using the boomerang distinguisher. We will now explain its correctness and compute its average time complexity (up to a constant factor).

We start by generating a structure S of $\frac{1}{\sqrt{p_{out}}pq}$ ciphertexts with differences in D_{fin}. This first step costs a time and memory: $\frac{1}{\sqrt{p_{out}}pq}$.

Algorithm 2 Boomerang last-round attack

Input: oracle access to $E = R \circ E_2 \circ E_1$
Output: guess of partial key K_{out} involved in the round(s) R

1: Generate a *structure* S of $\frac{1}{\sqrt{p_{\text{out}}}pq}$ ciphertexts of the form: $S \subseteq \{X \oplus \Delta, \Delta \in D_{\text{fin}}\}$
2: $L_{\text{pairs}} \leftarrow \emptyset$
3: Compute all the $C' = E(E^{-1}(C) \oplus \alpha)$ for all $C \in S$, sort S by values of C'
4: **ForEach** pair C'_i, C'_j such that $C'_i \oplus C'_j \in D_{\text{fin}}$
5: $\quad L_{\text{pairs}} \leftarrow L_{\text{pairs}} \cup \{(C_i, C_j, C'_i, C'_j)\}$
6: **EndFor** ▷ Here $|L_{\text{pairs}}| = \frac{|D_{\text{fin}}|}{2^n}\frac{1}{p_{\text{out}}(pq)^2}$
▷ Note that S being sorted, these pairs are computed efficiently
7: $L_{\text{triplets}} \leftarrow \emptyset$
8: **ForEach** pair $C_i, C_j, C'_i, C'_j \in L_{\text{pairs}}$
9: $\quad$ **ForEach** possible value K_{out} of the k_{out} bits of key
▷ $2^{k_{\text{out}}}$ trials, $2^{k_{\text{out}}}p_{\text{out}}^2$ solutions
10: $\quad\quad$ If both (C_i, C_j) and (C'_i, C'_j) lead to a difference δ through R^{-1} under K_{out}
11: $\quad\quad$ $L_{\text{triplets}} \leftarrow L_{\text{triplets}} \cup \{(C_i, C_j, K_{\text{out}})\}$
12: $\quad$ **EndFor**
13: **EndFor** ▷ Here $|L_{\text{triplets}}| = \frac{|D_{\text{fin}}|}{2^n}\frac{1}{(pq)^2}2^{k_{\text{out}}}p_{\text{out}}$
14: **return** all guesses of K_{out} in the triplet list

Now we will partially decrypt these ciphertexts and obtain pairs with difference δ. Among the $\frac{1}{p_{\text{out}}(pq)^2}$ pairs (C_i, C_j) in S, we expect $\frac{1}{(pq)^2}$ of them to be such that $R^{-1}(C_i) \oplus R^{-1}(C_j) = \delta$, where R involves the actual partial key K_{out} used in the last rounds. Then we define $C'_i = E(E^{-1}(C_i) \oplus \alpha)$ and $C'_j = E(E^{-1}(C_j) \oplus \alpha)$. By the boomerang property, with probability $(pq)^2$, we will have $R^{-1}(C'_i) \oplus R^{-1}(C'_j) = \delta$. This difference gets then mapped to D_{fin} with probability 1. Obtaining the list L_{pairs} costs $\frac{2}{\sqrt{p_{\text{out}}}pq}$ data and memory and $\frac{1}{\sqrt{p_{\text{out}}}pq}\log\left(\frac{1}{\sqrt{p_{\text{out}}}pq}\right)$ computations for sorting the C'_i (since D_{fin} is a vector space, we can sort according to its cosets). This sorting allows to extract efficiently the pairs C'_i, C'_j such that $C'_i \oplus C'_j \in D_{\text{fin}}$ at Step 4 of Algorithm 2.

There are enough quartets (C_i, C_j, C'_i, C'_j) in L_{pairs} to ensure that, for the good key guess of K_{out}, one of them is an actual boomerang quartet. Let C_{out} be the time complexity to obtain all valid keys K_{out} for a given pair (C_i, C_j) (Step 9 in Algorithm 2 is the naive way to do so, but there is usually a better algorithm). Since we expect on average $2^{k_{\text{out}}}p_{\text{out}}^2$ such keys, we have $C_{\text{out}} \geq 2^{k_{\text{out}}}p_{\text{out}}^2$.

After having obtained the list L_{pairs} of size $\frac{|D_{\text{fin}}|}{2^n}\frac{1}{p_{\text{out}}(pq)^2}$, we find all possible key guesses for each of these pairs, that is, all possible K_{out} such that (C_i, C_j) and (C'_i, C'_j) lead to a difference δ through R^{-1} (in which case they form a boomerang quartet). This costs a time $\frac{|D_{\text{fin}}|}{2^n}\frac{1}{p_{\text{out}}(pq)^2}C_{\text{out}}$. The list of triplets obtained is of size: $\frac{|D_{\text{fin}}|}{2^n}\frac{2^{k_{\text{out}}}p_{\text{out}}}{(pq)^2}$, and we know that the actual key K_{out} occurs in one of these triplets, because one of the pairs went through the boomerang. Thus, if we have: $\frac{|D_{\text{fin}}|}{2^n}\frac{2^{k_{\text{out}}}p_{\text{out}}}{(pq)^2} \leq 2^{k_{\text{out}}}$, we have reduced the number of possibilities for K_{out}. We perform an exhaustive search over the remaining $k - k_{\text{out}}$ bits of key. We obtain

a key-recovery attack of total time complexity:

$$\frac{1}{\sqrt{p_{\text{out}}}pq}\log\left(\frac{1}{\sqrt{p_{\text{out}}}pq}\right)+\frac{|D_{\text{fin}}|}{2^n}\frac{1}{p_{\text{out}}(pq)^2}\left(C_{\text{out}}+p_{\text{out}}^2 2^k\right) \quad , \tag{1}$$

in number of queries to E and E^{-1} or evaluations of the cipher. Note that C_{out} is counted relatively to the cost of an evaluation of E. The memory used is: $\max\left(\frac{2}{\sqrt{p_{\text{out}}}pq}, \frac{|D_{\text{fin}}|}{2^n}\frac{2^{k_{\text{out}}}p_{\text{out}}}{(pq)^2}\right)$. The data used is : $\frac{2}{\sqrt{p_{\text{out}}}pq}$.

Remark 1. Another situation commonly encountered in Boomerang attacks is when two (or more) boomerang pairs occur within the trials. In that case, it is possible to recognize immediately the good guess of K_{out}, as the only one that appears in two (or more) triples in the list L_{triplets}.

2.2 Mixing Boomerang Attacks

This variant of boomerang attacks [15] is related to mixture distinguishers.

Distinguisher. As in Section 2.1, the cipher is decomposed as $E = E_2 \circ E_1$ and we assume that a good (truncated) differential $\alpha \to \beta$ exists on E_1 with probability $p_\downarrow$ forwards and with probability $p_\uparrow$ backwards. We further assume that E_2 can be divided into a "left" and a "right" part $E_2 = (E_{2,L}, E_{2,R})$ (see the notations on Figure 2).

The distinguisher relies on the independence of $E_{2,L}$ and $E_{2,R}$. It requires $\frac{4}{p_\downarrow p_\uparrow}$ encryptions-decryptions, operations and negligible memory. We repeat:

1. Select P_1 at random, set $P_2 = P_1 \oplus \alpha$ and encrypt $C_1 = E(P_1)$ and $C_2 = E(P_2)$
2. Compute $C_3 = (C_{1,L}, C_{2,R})$ and $C_4 = (C_{2,L}, C_{1,R})$
3. Decrypt: $P_3 = E^{-1}(C_3)$ and $P_4 = E^{-1}(C_4)$

Indeed, in Step 1 we have $E_1(P_1) \oplus E_1(P_2) = \beta$ with probability $p_\downarrow$. By definition of E, $E_1(P_i) = E_2^{-1}(C_i)$, so in Step 2, we know that $E_2^{-1}(C_1) \oplus E_2^{-1}(C_2) = \beta$. If we separate $\beta = \beta_L || \beta_R$, this rewrites:

$$\begin{cases} E_{2,L}^{-1}(C_{1,L}) \oplus E_{2,L}^{-1}(C_{2,L}) = \beta_L \\ E_{2,R}^{-1}(C_{1,R}) \oplus E_{2,R}^{-1}(C_{2,R}) = \beta_R \end{cases} . \tag{2}$$

Thus, by definition of C_3, C_4, we also have $E_2^{-1}(C_3) \oplus E_2^{-1}(C_4) = \beta$. Finally in Step 3, with probability $p_\uparrow$, β gets mapped to α. After $(p_\downarrow p_\uparrow)^{-1}$ iterations, we obtain a valid quartet with probability 1/2.

Attack on First Round. We append one or more additional rounds R before those covered by the distinguisher. Similarly as before, we write $E = E_2 \circ E_1 \circ R$. We let D_{start} denote the set of differentials that can lead to α for the additional rounds R, p_{in} the probability to get α from an element of D_{start} and k_{in} the number of bits of the key involved in the additional rounds.

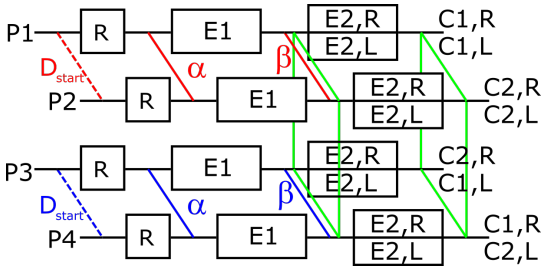

Fig. 2. First-round mixing boomerang attack.

Step 1: Generating Pairs. We generate a list of $\frac{1}{p_{\text{in}}p_{\downarrow}p_{\uparrow}}$ pairs of plaintexts (P_1, P_2) such that their difference is in D_{start}. Again, there can be one or multiple structures, but as the term $\frac{1}{p_{\text{in}}p_{\downarrow}p_{\uparrow}}$ (the number of pairs) will appear in the complexity, there is no difference between these two cases.

For each pair (P_1, P_2) we compute:

$$(C_1 = E(P_1), C_2 = E(P_2)), \quad (C_3 = (C_{1,L}, C_{2,R}), C_4 = (C_{2,L}, C_{1,R}), \\ (P_3 = E^{-1}(C_3), P_4 = E^{-1}(C_4)) \ . \quad (3)$$

For each pair, we have $R(P_1) \oplus R(P_2) = \alpha$ with probability p_{in}. Then, $(E_2 \circ E_1)^{-1}(C_3) \oplus (E_2 \circ E_1)^{-1}(C_4) = \alpha$ with probability $p_{\downarrow}p_{\uparrow}$. Thus, we expect that one pair satisfies the boomerang distinguisher; by decrypting through R we fall back on D_{start}.

Thus, we keep only the pairs (P_1, P_2) such that $P_3 \oplus P_4 \in D_{\text{start}}$, among which the pair that went through the boomerang must remain. We get a list of $\frac{|D_{\text{start}}|}{2^n}\frac{1}{p_{\text{in}}p_{\downarrow}p_{\uparrow}}$ pairs at a cost of $\frac{4}{p_{\text{in}}p_{\downarrow}p_{\uparrow}}$ computations and data.

Step 2: Sieving Key Bits. For each one of the kept pairs (P_1, P_2), we determine the values of K_{in} (the k_{in} bits of key that intervene in R) such that P_1, P_2 and P_3, P_4 both lead to a difference α through R. The average expected number of possible values for K_{in} for each pair is thus $2^{k_{\text{in}}}(p_{\text{in}})^2$. We let C_{in} denote the cost of finding all these possible values.

At a cost of $\frac{|D_{\text{start}}|}{2^n}\frac{1}{p_{\text{in}}p_{\downarrow}p_{\uparrow}}C_{\text{in}}$ computations, we get a list of $\frac{|D_{\text{start}}|}{2^n}\frac{1}{p_{\downarrow}p_{\uparrow}}2^{k_{\text{in}}}p_{\text{in}}$ elements $(P_1, P_2, K_{\text{in}})$. If we have:

$$\frac{|D_{\text{start}}|}{2^n}\frac{1}{p_{\downarrow}p_{\uparrow}}2^{k_{\text{in}}}p_{\text{in}} < 2^{k_{\text{in}}} \ , \quad (4)$$

then we have reduced the number of possible values for K_{in} and we do an exhaustive search of the remaining $k - k_{\text{in}}$ bits of key. The attack has a total time complexity:

$$\frac{1}{p_{\text{in}}p_{\downarrow}p_{\uparrow}} + \frac{|D_{\text{start}}|}{2^n}\frac{1}{p_{\text{in}}p_{\downarrow}p_{\uparrow}}\left(C_{\text{in}} + p_{\text{in}}^2 2^k\right) \ , \quad (5)$$

in number of queries to E and E^{-1} or evaluations of the cipher. Note that C_{in} is counted relatively to the cost of an evaluation of E.

2.3 Quantum tools

In this paper, we consider quantum algorithms written in the *quantum circuit model* (see [27] for an introduction). A quantum circuit starts with a pool of qubits initialized in a basis state. The state is modified through the application of *quantum gates*. The *time complexity* is the number of gates of the circuit. The memory complexity is the width of the circuit, or the number of qubits.

Complexity Estimations. A quantum procedure qualifies as an attack if it runs with a lower time complexity than the corresponding quantum generic procedure (in this paper, key-recovery via Grover's algorithm). Since we only need to perform comparisons, we can use as cost metric the evaluation of the primitive that is attacked. In our case, the time is counted in evaluations of the cipher E *as a quantum circuit*, and the memory in multiples of its block size n.

Quantum RAM. We shall specify whether or not our algorithms require the *quantum RAM* model, which can be seen as the quantum counterpart of classical random-access. In this model, a "qRAM" gate of cost 1 allows to perform memory lookups in superposition (see [2]).

Amplitude Amplification. We will use as a black-box the Amplitude Amplification framework of [8], which generalizes Grover's quantum search [18].

Theorem 1 ([8], Theorem 2 and 4). *Let $\mathcal{A}$ be a quantum algorithm that uses no intermediate measurements, let $f : X \to \{0,1\}$ be a boolean function that tests if an output of $\mathcal{A}$ is "good". Let a be the success probability of $\mathcal{A}$. Let $\theta_a = \arcsin\sqrt{a}$ and $0 < \theta_a \leq \frac{\pi}{2}$. There exists a quantum algorithm $\mathsf{Amplify}(\mathcal{A}, f)$ that runs in time: $\left\lfloor \frac{\pi}{4\theta_a} \right\rfloor (2T_{\mathcal{A}} + T_f + \mathcal{O}(\log_2 |X|))$, where $T_{\mathcal{A}}$ is the time complexity of $\mathcal{A}$ and T_f is the time complexity of a quantum algorithm for f. Measuring the output of $\mathsf{Amplify}(\mathcal{A}, f)$ yields a "good" output of $\mathcal{A}$ with probability $\max(1-a, a)$. If a is known exactly, then the probability of success can be brought to 1.*

We will neglect the term $\mathcal{O}(\log_2 |X|)$ since our complexity is expressed in quantum circuits for the attacked cipher, usually much more expensive. The memory consumed by the procedure itself is also usually negligible with respect to the memory of $\mathcal{A}$ and f.

To the quantum procedure $\mathsf{Amplify}(\mathcal{A}, f)$, there corresponds a *classical* exhaustive search procedure with about $\left(\lfloor \pi/(4\theta_a) \rfloor\right)^2$ iterations, which consists in a **Repeat** loop that calls $\mathcal{A}$ and tests its output until it satisfies the constraint f. Since $\mathcal{A}$ in Theorem 1 can be *any* quantum algorithm, especially another quantum search, there is a recursive correspondence between classical procedures made of such **Repeat** loops and quantum procedures made of nested quantum searches [7]. It follows that many quantum attacks, including the ones of this paper, can be first described classically.

More precisely, we will write down our algorithms using a **Repeat** block which specifies a number of iterations. Inside, we run a randomized algorithm $\mathcal{A}$

followed by a test f (**If** block) that checks if the current state is "good" or "bad". It is then expected that, if a good state exists, it is found after the specified number of iterations. Using Theorem 1, we make this algorithm correspond to $\mathsf{Amplify}(\mathcal{A}, f)$. If a solution exists, then $\mathsf{Amplify}(\mathcal{A}, f)$ produces it (or a superposition of possible solutions).

Searching for Collisions. We will also need to solve the following problem: given access to a random function $f : \{0,1\}^n \to \{0,1\}^m$, where $n \geq \frac{m}{2}$, find a pair (x, y) such that $f(x) = f(y)$. This can be done classically by sorting $\mathcal{O}\left(2^{m/2}\right)$ queries to f. In the quantum setting, we can either use quantum search among the pairs (x, y), for a time $\mathcal{O}\left(2^{m/2}\right)$ and a negligible memory, or *quantum collision search*. In that case we use Ambainis' algorithm [2], which works even in the worst case $n = \frac{m}{2}$ (where a single collision exists on expectation). It requires the qRAM model. The precise analysis made in [10] gives a number of $\left(\frac{\pi}{2}\right)^2 2^{m/3}$ queries in our case for an overwhelming success. These queries are the dominant term. Note that the lower bound on quantum collision search for random functions [1,30] prevents the existence of a more efficient algorithm

2.4 On Quantum Scenarios

In quantum symmetric cryptanalysis, there exist two scenarios that depend on the capabilities of the attacker. • the Q1 setting allows an attacker to perform quantum computations, but he can only use classical queries to black-box primitives (in this paper, an encryption oracle in the secret-key model). • the Q2 setting allows the attacker to query the black-box as a *quantum* oracle instead of a classical oracle. Although the latter is much more powerful, it remains non-trivial. Besides, as shown in [7], when attacks requires low data, it is sometimes possible to replace Q2 queries by Q1. Thus, when following classical design patterns to perform quantum key-recoveries, like in [7], it seems more suitable to consider the Q2 model by default and then see if they can be avoided afterwards. Note that it is not unusual for quantum attacks to require superposition queries, as it is the case of many attacks in [20].

3 Quantum Boomerang Attacks

In this section, we adapt the attacks presented in Section 2 to the quantum setting.

3.1 Quantum Boomerang Distinguisher

Similarly to the differential attacks from [20], the boomerang distinguisher of Algorithm 1 can be accelerated *in the Q2 setting* (Algorithm 3). The **Repeat** loop becomes a quantum search with $\frac{\pi}{4}(pq)^{-1}$ iterations looking for an element P_1 such that $P_1, P_1 \oplus \alpha$ forms a boomerang pair. If E satisfies the expected property, such a P_1 will be found, otherwise measuring the result will give us an

Algorithm 3 Quantum Boomerang Distinguisher

Input: *superposition* oracle access to $E = E_2 \circ E_1$, or a random permutation
Output: "E" or "Random"
Repeat $\frac{\pi}{4}(pq)^{-1}$ **times**
  Select P_1 at random
  If $E^{-1}(E(P_1) \oplus \delta) = E^{-1}(E(P_1 \oplus \alpha) \oplus \delta) \oplus \alpha$ **then**
    the state (P_1) is "good" **Else** the state (P_1) is "bad"
EndRepeat
Measure and check if a solution was found
If it was found **return** "E" **Else Return** "Random"

invalid pair. We make in total $\frac{\pi}{4}(pq)^{-1} \times 8$ queries to E and E^{-1} (each query is made twice for reversibility), which are in superposition since the quantum search space is on the value P_1.

3.2 Quantum Boomerang Last-rounds Attack

Recall that the last-rounds attack of Section 2.1 is in two phases: first, we use the boomerang property to sieve the k_{out} bits that intervene in the last rounds R. Second, we perform an exhaustive search on the $k - k_{\text{out}}$ remaining key bits.

For now, let us put aside the small constant factors. We count all quantum times in evaluations of E or E^{-1} and consider a Q2 query to E or E^{-1} to cost the same. The generic key-recovery is Grover's exhaustive search, with a time $2^{k/2}$ and a negligible number of Q1 queries. Our first idea is to go below that using the sieve on the k_{out} bits of key that the Boomerang property gives. If we can produce the superposition of valid K_{out} with a time complexity below $2^{(k-k_{\text{out}})/2}$, and if there are strictly less than $2^{k_{\text{out}}}$ elements in this superposition, then we can do better than Grover search: we Amplify an algorithm that first samples a K_{out}, then tries to complete the key in time $2^{(k-k_{\text{out}})/2}$.

In order to produce a possible value of K_{out}, we look for a pair C_1, C_2 such that: • $C_1 \oplus C_2 \in D_{\text{fin}}$ and $C'_1 \oplus C'_2 \in D_{\text{fin}}$, where $C'_i = E(E^{-1}(C_i) \oplus \alpha)$; • there exists a subkey K_{out} such that (C_1, C_2) and (C'_1, C'_2) decrypt to the difference δ through R^{-1} under K_{out}.

Finding such a valid pair C_1, C_2 is a collision search problem, solved with Ambainis' algorithm[4]. Let us assume that $D_{\text{fin}} \geq 2^{n/3}$, which corresponds to the case where a single structure is enough. A structure contains $|D_{\text{fin}}|^2$ pairs, and the constraint $C'_1 \oplus C'_2 \in D_{\text{fin}}$ selects a proportion $\frac{|D_{\text{fin}}|}{2^n}$ of them. Thus there is at least one solution.

We now check the number of iterations in the loops of Algorithm 4.

Step 1: the number of iterations to perform depends on the probability that a subkey guess K_{out}, obtained with the procedure in the loop, is the good one. In the classical analysis of the procedure, we obtained a list of $\frac{|D_{\text{fin}}|}{2^n} \frac{2^{k_{\text{out}}} p_{\text{out}}}{(pq)^2}$ tuples $(C_1, C_2, C'_1, C'_2, K_{\text{out}})$, and we took the key K_{out} from one of these

[4] If there are many solutions, an alternative is the BHT algorithm [9].

Algorithm 4 Quantum Last-rounds Key-recovery

Input: *superposition* oracle access to E and E^{-1}
Output: the full key

Repeat $(\frac{|D_{\text{fin}}|}{2^n}\frac{2^{k_{\text{out}}}p_{\text{out}}}{(pq)^2})$ **times**
  Sample a subkey guess K_{out} using:
  Repeat $(\frac{1}{2^{k_{\text{out}}}p_{\text{out}}^2})$ **times**
    Use Ambainis' algorithm to find a valid pair (C_1, C_2, C_1', C_2')
    Repeat $2^{k_{\text{out}}}$ **times**
      Sample a guess of K_{out}
      If $R^{-1}(C_1) \oplus R^{-1}(C_2) = \delta = R^{-1}(C_1') \oplus R^{-1}(C_2')$ then K_{out} is "good"
    EndRepeat
    If we obtained a valid K_{out} then (C_1, C_2, C_1', C_2') is "good"
  EndRepeat
  Given this subkey guess K_{out}:
  Repeat $2^{(k-k_{\text{out}})}$ **times**
    Sample a choice of the $k - k_{\text{out}}$ other key bits, check if the full K matches
  EndRepeat
  If we obtained a valid full key K then K_{out} is "good"
EndRepeat
Measure K_{out}, recompute K and return it

tuples. We know that the good one is in one of these tuples. The computation inside the **Repeat** loop is the same, so the probability of success (the full key is found) is $\frac{2^n}{|D_{\text{fin}}|}\frac{(pq)^2}{2^{k_{\text{out}}}p_{\text{out}}}$ and the number of iterations follows.

Step 3: here, we are iterating on tuples (C_1, C_2, C_1', C_2') until we find a key candidate. The probability that a tuple yields a key candidate is $2^{k_{\text{out}}}p_{\text{out}}^2$, thus there are $\max(1, \frac{1}{\sqrt{2^{k_{\text{out}}}p_{\text{out}}^2}})$ iterations. We will assume $2^{k_{\text{out}}}p_{\text{out}}^2 \leq 1$. Note that the key candidate (actually the superposition of all possible candidates for the given pair) is obtained by exhaustive search, but a more involved procedure could likely be applied in order to reduce the time complexity.

Step 12: having obtained a candidate for K_{out}, it remains to try it: for this, we perform a simple Grover search over the remaining $k - k_{\text{out}}$ bits.

By changing the loops into nested quantum searches, the *quantum* time complexity of Algorithm 4 is:

$$\sqrt{\frac{|D_{\text{fin}}|}{2^n}\frac{2^{k_{\text{out}}}p_{\text{out}}}{(pq)^2}}\left(\frac{1}{\sqrt{2^{k_{\text{out}}}p_{\text{out}}^2}}\left(\left(\frac{2^n}{|D_{\text{fin}}|}\right)^{1/3} + 2^{k_{\text{out}}/2}\right) + 2^{(k-k_{\text{out}})/2}\right)$$

$$= \underbrace{\sqrt{\frac{|D_{\text{fin}}|}{2^n}\frac{2^{k_{\text{out}}}p_{\text{out}}}{(pq)^2}}}_{<2^{k_{\text{out}}/2} \text{ by the classical analysis}} \left(\frac{2^{n/3}}{2^{k_{\text{out}}/2}p_{\text{out}}|D_{\text{fin}}|^{1/3}} + \frac{1}{p_{\text{out}}} + \underbrace{2^{(k-k_{\text{out}})/2}}_{\text{Exhaustive search}}\right).$$

The algorithm requires Q2 queries to E and E^{-1}, and uses $\left(2^n/|D_{\text{fin}}|\right)^{1/3}$ qRAM due to the use of Ambainis' algorithm (or BHT) to find valid pairs.

However, if we can replace the factor $\left(2^n/|D_{\text{fin}}|\right)^{1/3}$ by $\left(2^n/|D_{\text{fin}}|\right)^{1/2}$, then we can use a Grover search instead of Ambainis' algorithm, and the algorithm will now require a small number of qubits only.

Making the Attack Q1. Recall that the attack requires only a structure of $\frac{1}{\sqrt{p_{\text{out}}}(pq)}$ ciphertexts. For each of these, we will have to compute $f(C) = E(E^{-1}(C) \oplus \alpha)$. Thus, all the queries made to E and E^{-1} asked during the attack fall actually in a set of size $\frac{1}{\sqrt{p_{\text{out}}}(pq)}$. If we have $\frac{1}{\sqrt{p_{\text{out}}}(pq)} < 2^{k/2}$, then we can *make all the queries classically beforehand* and store the results in a quantum RAM of size $\frac{1}{\sqrt{p_{\text{out}}}(pq)}$. Inside Algorithm 4, we replace the on-the-fly computation of f by a memory lookup.

3.3 Quantum Mixing Boomerang Distinguisher

We now study the Mixing Boomerang attacks presented in Section 2.2. Since the distinguisher is a single loop similar to Algorithm 1, it can be replaced by a quantum search of a valid pair $(P_1, P_1 \oplus \alpha)$ with about $1/\sqrt{p_\downarrow p_\uparrow}$ iterations, with Q2 queries and negligible memory.

Quantum First-round Mixing Attack. The attack will work similarly as the quantum last-round attack of Algorithm 4. The main difference is that we do not need a collision search algorithm to filter out the valid pairs (P_1, P_2) such that $P_3 \oplus P_4 \in D_{\text{start}}$.

We analyze the time complexity of Algorithm 5, when translated into nested Amplitude Amplifications.

Step 1: here, the number of iterations depends on the number of possible keys K_{in} obtained in the sieving step. From the classical analysis, we know that there will be $\frac{|D_{\text{start}}|}{2^n}\frac{1}{p_\uparrow p_\downarrow}2^{k_{\text{in}}}p_{\text{in}} < 2^{k_{\text{in}}}$ tuples $(P_1, P_2, P_3, P_4, K_{\text{in}})$ obtained, among which the good subkey K_{in} appears (at least once). Thus there are at most $\left(\frac{|D_{\text{start}}|}{2^n}\frac{1}{p_\uparrow p_\downarrow}2^{k_{\text{in}}}p_{\text{in}}\right)^{1/2}$ iterations.

Step 3: from a valid quadruple (P_1, P_2, P_3, P_4), i.e., one that satisfies $P_1 \oplus P_2 \in D_{\text{start}}$ and $P_3 \oplus P_4 \in D_{\text{start}}$, the probability that we obtain a subkey guess is $2^{k_{\text{in}}}p_{\text{in}}^2$. This gives the number of iterations.

Step 5: the probability for a given pair to be valid is simply $\frac{|D_{\text{start}}|}{2^n}$.

Step 16: similarly to Algorithm 4, we complete the quantum search for the key.

The total quantum time complexity of Algorithm 5 is given by:

$$\sqrt{\frac{|D_{\text{start}}|}{2^n}\frac{2^{k_{\text{in}}}p_{\text{in}}}{p_\uparrow p_\downarrow}}\left(\frac{1}{\sqrt{2^{k_{\text{in}}}p_{\text{in}}^2}}\left(\sqrt{\frac{2^n}{|D_{\text{start}}|}}+2^{k_{\text{in}}/2}\right)+2^{(k-k_{\text{in}})/2}\right)$$
$$=\sqrt{\frac{|D_{\text{start}}|}{2^n}\frac{2^{k_{\text{in}}}p_{\text{in}}}{p_\uparrow p_\downarrow}}\left(\frac{2^{n/2}}{2^{k_{\text{in}}/2}p_{\text{in}}|D_{\text{start}}|^{1/2}}+\frac{1}{p_{\text{in}}}+2^{(k-k_{\text{in}})/2}\right) \quad . \quad (6)$$

Algorithm 5 Quantum first-rounds key-recovery

Input: *superposition* oracle access to E and E^{-1}
Output: the full key
Repeat $\frac{|D_{\text{start}}|}{2^n}\frac{1}{p_\uparrow p_\downarrow}2^{k_{\text{in}}}p_{\text{in}}$ **times**
  Sample a subkey guess K_{in} using:
  Repeat $1/(2^{k_{\text{in}}}p_{\text{in}}^2)$ **times**
    Sample a valid (P_1, P_2) using:
    Repeat $\frac{2^n}{|D_{\text{start}}|}$ **times**
      Sample a pair $P_1, P_2 = P_1 \oplus \alpha$, compute P_3, P_4 as in the distinguisher
      If $P_3 \oplus P_4 = \alpha$, then (P_1, P_2) is "good"
    EndRepeat
    Repeat $2^{k_{\text{in}}}$ **times**
      Sample a guess of K_{in}
      If $R(P_1) \oplus R(P_2) = \alpha = R(P_3) \oplus R(P_4)$, then K_{in} is "good"
    EndRepeat
    If there is a valid K_{in}, then $(P_1, P_2, K_{\text{in}})$ is "good"
  EndRepeat
  Given this subkey guess K_{in}:
  Repeat $2^{k-k_{\text{in}}}$ **times**
    Sample a choice of the $k - k_{\text{in}}$ other key bits, check if the full K matches
  EndRepeat
  If the valid full key K was obtained, K_{in} is good
EndRepeat
Measure K_{in}, recompute the full K and return it

The baseline algorithm does not use quantum RAM and requires only a small number of qubits, but it also relies on Q2 queries. Depending on the amount of data required, it may be possible to make it Q1: for this, we query all the $\frac{1}{p_{\text{in}}p_\uparrow p_\downarrow}$ pairs required by the classical attack, and store the tuples P_1, P_2, P_3, P_4 in a quantum RAM. This can only work if $\frac{1}{p_{\text{in}}p_\uparrow p_\downarrow} < 2^{k/2}$.

4 Application to SAFER

SAFER is a family of block ciphers (Substitution-Permutation Networks) that dates back to the SAFER K-64 proposal of [23]. SAFER-K-64 was a block cipher of 64 bits. A new version SAFER+, with 128-bit blocks, was a candidate of the AES competition organized by the NIST [24]. Finally, another variant, SAFER++, was submitted to the NESSIE project [25]. Here, we focus on SAFER++ and more precisely, on the boomerang attack presented in [3]. It reaches 5.5 rounds out of 7 total rounds.

4.1 Description of the Cipher

We use $\boxplus$ and $\boxminus$ to denote addition and subtraction modulo 2^8. We consider the version of SAFER++ with a 128-bit key (and 128-bit blocks), which has

7 rounds. The round function (see Figure 4 in Appendix A, or Fig. 1 in [3]) consists of key additions, a nonlinear layer and a linear layer. The state and the round keys are represented as 16 bytes numbered from 0 to 15, partitioned into $S_1 = \{0, 3, 4, 7, 8, 11, 12, 15\}$ and $S_2 = \{1, 2, 5, 6, 9, 10, 13, 14\}$. A single round performs the following steps:

Upper key addition: bytes in S_1 of the 16-byte round key are XORed to the corresponding bytes of the block and the others are added modulo 2^8.

Nonlinear layer: it uses two functions X and L:

$$X(a) = (45^a \bmod 257) \bmod 256, \quad L(a) = \log_{45}(a) \bmod 257 \text{ and } L(0) = 128, \tag{7}$$

where L is the inverse of X. Bytes in S_1 go through X and bytes in S_2 go through L.

Lower key addition: another 16-byte round key is added to the state, using modular addition in S_1 and XOR in S_2.

Linear layer: it repeats twice the following: a permutation of the bytes, followed by a Pseudo Hadamard Transform to groups of 4 bytes. This linear layer contains a total of 48 modular additions of individual bytes and can be described by the matrix given in [3].

4.2 5-Round Classical Boomerang Attack

We present the attack from [3]. It relies on a boomerang distinguisher on 4.5 rounds of SAFER++. If A denotes the linear layer and S the nonlinear layer (X and L), the distinguisher works against a sequence $A_0 S_1 A_1 - S_2 - A_2 S_3 A_3 S_4 A_4$. It would work against 4 rounds if any S-Boxes were used, but it reaches an additional 0.5 round thanks to the following property of the inverse-based S-Boxes in SAFER++:

$$\forall a, X(a) \boxplus X(a \boxplus 128) = 1 \bmod 256 \ . \tag{8}$$

The top part of the boomerang starts with pairs of plaintexts having difference $\alpha = (0, x, 0, 0, x, x, 0, 0, 0, 0, 0, -4x, 0, 0, x, -x)$. This difference is such that it maps to a single active byte after A_0, and up to 16 active bytes after $A_1 \circ S_1$.

The difference added to ciphertexts is:

$$\delta = (0, 0, 128, 0, 128, 128, 0, 0, 0, 0, 0, 0, 128, 0, 0, 0) \ .$$

After $S_4^{-1} \circ A_4^{-1}$, it maps to a difference $(0, 0, 0, 0, 0, x, -x, 0, \ldots)$ with probability 2^{-7} (not 2^{-8} because the difference 128 maps to odd differences through X). Then after A_3^{-1}, bytes 3, 9, 11 and 14 are active.

In order to traverse the middle S-Box, the authors observe that if the byte-differences coming both from the top and the bottom part of the boomerang are 128, then the boomerang traverses this S-Box layer for free. Indeed, if we encrypt $P_1, P_2 = P \boxplus 128$ through X, we obtain $X(P), 1 \boxminus X(P)$, i.e., two values that sum to 1. If the difference coming from the bottom part is 128, then the second pair of ciphertexts $C_3, C_4 = C_1 \boxplus 128, C_2 \boxplus 128$ still sums to 1. Thus, by going through X^{-1}, they get mapped to a difference 128 with probability 1.

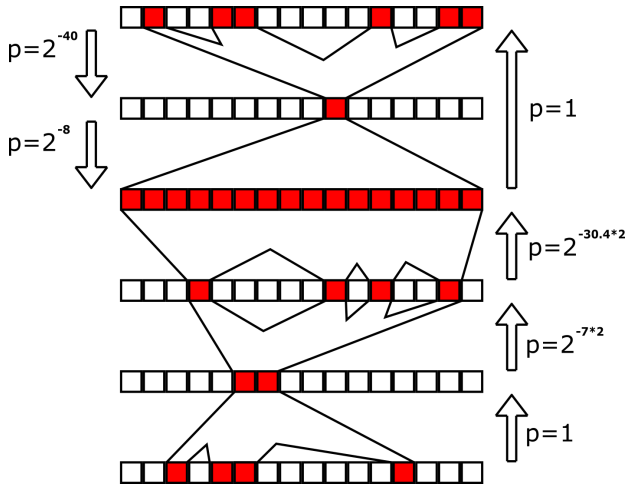

Fig. 3. Classical boomerang attack on SAFER++ (Fig. 4 in [3])

Boomerang Attack. The full differential path of the classical 5-round boomerang attack of [3] is reproduced in Figure 3. We use $S_0A_0S_1A_1 - S_2 - A_2S_3A_3S_4A_4$ to denote the whole construction. The total probability for the boomerang to occur is: $(2^{-7-30.4})^2 2^{-40-8} = 2^{-122.8}$.

Top boomerang: At the top, the difference is active in 6 bytes in positions 1, 4, 5, 11, 14, 15. This difference propagates to a single active byte after one round ($A_0 \circ S_0$) with probability 2^{-40}, as there are 5 byte-conditions to meet. This difference is then mapped to a single active byte of difference 128 after the next nonlinear layer S_1, with probability 2^{-8}, and with probability 1, we obtain a difference 128 in bytes $0, 1, 2, 3, 8, 9, 11, 13, 14, 15$ after A_1. All other bytes are inactive.

Bottom boomerang: In the lower part of the boomerang, we start with a difference 128 (changing one bit) in the 4 bytes at positions 2, 4, 5, 12. By decrypting through S_4A_4, this yields a difference $x, -x$ in bytes 5 and 6 with probability 1. By decrypting through S_3A_3, this yields a difference active in bytes $3, 9, 11, 14$ with probability 2^{-7}. Then, by decrypting through A_2, we can obtain a difference 128 in all bytes except 2, 4, 9, 12 with probability $2^{-30.4}$.

Middle: In the middle, the S-Box trick allows to traverse the layer S_2 with probability 1. When going backwards from this middle to the top, everything propagates with probability 1.

We recall the attack procedure from [3] in Algorithm 6. It requires 2^{78} encryptions.

From a pool of 2^{48} plaintexts, we get approximately 2^{95} pairs, which have to satisfy an 80-bit condition, so we get 2^{15} pairs in Step 4. The probability that the structure contains a boomerang is approximately $2^{-27.8}$, so by using

Algorithm 6 Classical attack on 5-round SAFER++ (adapted from [3], Sec. 6.3).

Input: access to E and E^{-1}

1: Prepare 2^{29} structures of 2^{48} plaintexts P_i that take all values in bytes 1, 4, 5, 11, 14 and 15 and are constant in the others

2: For each structure, obtain the ciphertexts $C_i = E(P_i)$

3: For each structure, add the difference δ (one bit in bytes 2, 4, 5, 12) to the C_i and decrypt: obtain $Q_i = E^{-1}(C_i \oplus \delta)$

4: For each structure, the Q_i on the values of the bytes that are constant in the P_i and keep the pairs Q_i, Q_j that have zero difference on these ten bytes

▷ We search for a difference after the S-Box of the form $(0, x, 0, 0, x, x, 0, 0, 0, 0, 0, -4x, 0, 0, x, -x)$

5: For each possible quartet (P_j, P_k, Q_j, Q_k), search for the upper key bytes $K_0^4, K_0^{11}, K_0^{14}, K_0^{15}$ such that after the first S-Box layer, we have a difference $(x, -4x, x, -x)$ (x odd) between P_j and P_k and between Q_j and Q_k. The quartets are kept with their partial key and the observed x.

6: For each quartet, search for the key bytes K_0^1 and $K_{0'}^2$ (upper and lower key addition in byte 0) such that the difference after the first S-Box is the right one. Repeat this for K_0^5 and $K_{0'}^6$.

7: Find a suggestion of the 8 key bytes that appears twice, and run exhaustive search on the remaining bytes.

2^{29} structures, we can expect two boomerang quartets to occur among the 2^{44} filtered quartets.

Next, we guess 4 bytes of the first upper round key $(K_0^4, K_0^{11}, K_0^{14}, K_0^{15})$ to remove wrong quartets. For each guess, we have a 25-bit condition on both pairs. Thus, 50 bits of restriction on the quartets. At this point, we obtain $2^{44} \times 2^{32} \times 2^{-50} = 2^{26}$ valid quartets and key guesses. Note that this already yields a valid key-recovery attack, as we have been able to reduce the number of possible $K_0^4, K_0^{11}, K_0^{14}, K_0^{15}$ from 2^{32} to 2^{26}.

For the next 4 key bytes guessed at Step 6, there is approximately one choice for each valid quartet currently kept. Thus, at this point, we have 2^{26} guesses of 8 bytes of the key, among which we expect the good one to occur. If we ran immediately an exhaustive search for the remaining 8 bytes, we would obtain a complexity $2^{26} \times 2^{8\times 8} = 2^{90}$.

However, there are two valid boomerang pairs: one of the key guesses occurs twice. As the key guesses are 8 bytes and there are 2^{26} of them, we can expect that random collisions are not likely to occur, and the only key guess occurring twice is the good one. Thus exhaustive search is ran only once. The bottleneck of the complexity lies in encrypting and decrypting the 2^{29} structures, for a total of 2^{78} time and queries.

Extension. Another half round can be added at the end, by making 30 bits of additional key guess and running Algorithm 6 2^{30} times.

4.3 Quantum Boomerang Attack

Our quantum attack on 5-round SAFER++ uses the framework of Algorithm 4, adapted to the setting of the classical attack (Algorithm 6). We will first explain its time complexity, and show how it goes below Grover's exhaustive search in time 2^{64}. Recall that the time complexity is counted in number of evaluations of SAFER++.

Algorithm 7 Attack on 5-round SAFER++.

Input: superposition access to E and E^{-1}
Output: the full key
1: **Repeat** 2^{26} **times**
2: Sample a guess for $K_0^4, K_0^{11}, K_0^{14}, K_0^{15}, K_0^1, K_{0'}^2, K_0^5, K_{0'}^6$ using:
3: **Repeat** 2^{18} **times** ▷ a valid quartet yields a key guess with prob. 2^{-18}
4: Find a valid quartet
▷ Either using Ambainis' algorithm, or quantum search
▷ We need the same number of quartets as classically: 2^{78} plaintexts in total
5: **Repeat** 2^{32} **times** ▷ at most one solution
6: Pick a key guess $K_0^4, K_0^{11}, K_0^{14}, K_0^{15}$
7: Check if this key guess yields the good difference $(x, -4x, x, -x)$ for both pairs of the quartet
8: **EndRepeat**
9: If a solution was found, return it (and the quartet)
10: **EndRepeat**
11: Search exhaustively for $K_0^1, K_{0'}^2$ and $K_{0'}^6$ (keep at most one value)
▷ there is on average one value; sometimes there can be more; but we only need this to work for the actual boomerang quartet
12: Given the current guess of 8 bytes of key, complete the key by Grover search
13: **EndRepeat**
14: Measure and return the full key

Assuming that we use Grover search at Step 4 (instead of collision search), the attack *would* run in approximately:

$$\sqrt{2^{26}}\left(\sqrt{2^{18}}\left(\underbrace{\sqrt{2^{80}}}_{\text{Step 4}}+\sqrt{2^{32}}+\underbrace{2\cdot 2^{16}}_{\text{Step 11}}\right)+\underbrace{\sqrt{2^{64}}}_{\text{Step 12}}\right) \tag{9}$$

calls to SAFER++ and its inverse. The dominating term comes from the search for quartets satisfying the partial collision condition on 10 bytes. Indeed, the classical attack can sample these quartets very easily using structures. If we use classical search in Algorithm 7, we can obtain an attack running with polynomial memory in little less time than 2^{128}. However, in the quantum setting, additional factors from Amplitude Amplification will prevent that, so we must resort to quantum collision search in Step 4.

Since a structure is of size 2^{48} and we need 80 bits of collision, we can use Ambainis' algorithm to produce a superposition of colliding pairs (thus valid quartets) in about $(\pi/2)^2 2^{80/3} \simeq 2^{28.3}$ calls to E and E^{-1} in superposition.

Assuming that the numbers of iterations are exact, but putting the additional factors of quantum search, we obtain:

$$2\left\lfloor\frac{\pi}{4}2^{13}\right\rfloor\left(2\left\lfloor\frac{\pi}{4}2^{9}\right\rfloor\left(\underbrace{2^{28.3}\times 4}_{\text{Step 4}}+\underbrace{2\left\lfloor\frac{\pi}{4}2^{16}\right\rfloor}_{\text{Step 11}}+4\cdot 2^{16}\right)+\underbrace{\left\lfloor\frac{\pi}{4}2^{32}\right\rfloor\times 4}_{\text{Step 12}}\right)$$
$$\simeq 2^{13.65}\left(2^{9.65}(2^{30.3}+2^{16.65}+2^{19})+2^{33.65}\right)\simeq 2^{53.6}<2^{64} \quad (10)$$

where the time is counted in computations of SAFER++ and its inverse, and we assume that a black-box (quantum) query costs the same. In the last step (quantum exhaustive search over 8 key bytes), we match against two given plaintext-ciphertext pairs.

In Step 12, we know that there is exactly one solution (or none) in a search space of size 2^{64}. Thus, we can run *exact* Amplitude Amplification with $\left\lfloor\frac{\pi}{4}2^{32}\right\rfloor$ iterations and succeed with probability 1 (either we find the solution, or prove that there is none). For the other loops, we can ensure a high probability of success by making a few assumptions on the *classical* attack: • the number of valid quartets for each structure is close to the average (this ensures that Ambainis' algorithm works as intended; if necessary we make multiple copies of it); • the actual boomerang pair yields only a single value at Step 11, and our algorithm does not lose it; • the probability, for a given guess of 8 key bytes sampled at Step 2, to be the good one, is close to 2^{-26}.

The attack requires 2^{27} registers of 256 qubits to store the values of P_i and $E^{-1}(E(P_i)\oplus\delta)$ for each P_i, and $2^{53.6}$ superposition queries to E and E^{-1}.

5 Application to Related-key AES

In Appendix B , we detail the quantum version of a classical related-key attack on AES-256 from [4]. The classical attack recovers the key in time $2^{99.5}$ using a tuple of 4 related keys. With the same configuration, our quantum attack runs in time $2^{96.4}$ (counted in evaluations of the cipher, forwards and backwards), which is rather a minor improvement compared to what we could do with SAFER++.

The reason for this is that the structure of the attack is different from the framework of Section 3. Here, we are looking among the quartets (and all the key guesses that they generate) for a key guess that appears at least three times. With high probability, there is only one such guess, which corresponds to the three valid boomerang quartets that should exist in the search space.

Classically, it takes time N to examine N objects and find the one that appears three times. This *3-distinctness* problem can be solved quantumly with a generalization of Ambainis' element distinctness algorithm [2], but it will take time $\mathcal{O}\left(N^{3/4}\right)$, which is far from a square-root speedup. This is the core reason why this attack does not perform well.

In the quantum setting, it is a much better strategy to discriminate the right key guess by checking exhaustively the remaining bits of key (e.g., Step 12 in Algorithm 4). In the attack on AES-256, using the path of [4], this is not possible because there are too many bits remaining. There probably exists a more suitable path, which would yield a more efficient quantum procedure. We leave this as an open question.

6 Conclusion

In this paper we proposed an efficient quantum boomerang attack, as well as a variant for mixing boomerang attacks. We proposed several improvements for different cases, as reducing the quantum RAM need or making Q2 attacks work in Q1 under certain circumstances. In some cases, our attacks reach a quadratic speed up with respect to classical attacks. This shows that boomerang attacks are also performant cryptanalysis tools for the post-quantum world, that will be needed for correctly determining the best security margins.

Acknowledgments. This work has been supported by the European Union's H2020 project No. 714294 (QUASYModo) and by the ERC Advanced Grant ALGSTRONGCRYPTO (project 740972).

References

1. Aaronson, S., Shi, Y.: Quantum lower bounds for the collision and the element distinctness problems. J. ACM 51(4), 595–605 (2004)
2. Ambainis, A.: Quantum walk algorithm for element distinctness. SIAM J. Comput. 37(1), 210–239 (2007)
3. Biryukov, A., Cannière, C.D., Dellkrantz, G.: Cryptanalysis of SAFER++. In: CRYPTO. Lecture Notes in Computer Science, vol. 2729, pp. 195–211. Springer (2003)
4. Biryukov, A., Khovratovich, D.: Related-key cryptanalysis of the full AES-192 and AES-256. In: ASIACRYPT. Lecture Notes in Computer Science, vol. 5912, pp. 1–18. Springer (2009)
5. Bonnetain, X., Jaques, S.: Quantum period finding against symmetric primitives in practice. IACR Cryptol. ePrint Arch. 2020, 1418 (2020), https://eprint.iacr.org/2020/1418
6. Bonnetain, X., Naya-Plasencia, M., Schrottenloher, A.: On quantum slide attacks. In: SAC. Lecture Notes in Computer Science, vol. 11959, pp. 492–519. Springer (2019)
7. Bonnetain, X., Naya-Plasencia, M., Schrottenloher, A.: Quantum security analysis of AES. IACR Trans. Symmetric Cryptol. 2019(2), 55–93 (2019)
8. Brassard, G., Hoyer, P., Mosca, M., Tapp, A.: Quantum amplitude amplification and estimation. Contemporary Mathematics 305, 53–74 (2002)
9. 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)

10. Childs, A.M., Eisenberg, J.M.: Quantum algorithms for subset finding. Quantum Inf. Comput. 5(7), 593–604 (2005)
11. Cid, C., Huang, T., Peyrin, T., Sasaki, Y., Song, L.: Boomerang connectivity table: A new cryptanalysis tool. In: EUROCRYPT (2). Lecture Notes in Computer Science, vol. 10821, pp. 683–714. Springer (2018)
12. Daemen, J., Rijmen, V.: AES proposal: Rijndael. Submission to NIST AES competition (1999)
13. Demirci, H., Selçuk, A.A.: A meet-in-the-middle attack on 8-round AES. In: FSE. Lecture Notes in Computer Science, vol. 5086, pp. 116–126. Springer (2008)
14. Derbez, P., Fouque, P., Jean, J.: Improved key recovery attacks on reduced-round AES in the single-key setting. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 7881, pp. 371–387. Springer (2013)
15. Dunkelman, O., Keller, N., Ronen, E., Shamir, A.: The retracing boomerang attack. In: EUROCRYPT (1). Lecture Notes in Computer Science, vol. 12105, pp. 280–309. Springer (2020)
16. Dunkelman, O., Keller, N., Shamir, A.: A practical-time related-key attack on the KASUMI cryptosystem used in GSM and 3g telephony. vol. 27, pp. 824–849 (2014)
17. Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D.A., Whiting, D.: Improved cryptanalysis of rijndael. In: FSE. Lecture Notes in Computer Science, vol. 1978, pp. 213–230. Springer (2000)
18. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: STOC. pp. 212–219. ACM (1996)
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)
20. Kaplan, M., Leurent, G., Leverrier, A., Naya-Plasencia, M.: Quantum differential and linear cryptanalysis. IACR Trans. Symmetric Cryptol. 2016(1), 71–94 (2016)
21. Kuwakado, H., Morii, M.: Quantum distinguisher between the 3-round feistel cipher and the random permutation. In: ISIT. pp. 2682–2685. IEEE (2010)
22. Kuwakado, H., Morii, M.: Security on the quantum-type even-mansour cipher. In: ISITA. pp. 312–316. IEEE (2012)
23. Massey, J.L.: SAFER K-64: A byte-oriented block-ciphering algorithm. In: FSE. Lecture Notes in Computer Science, vol. 809, pp. 1–17. Springer (1993)
24. Massey, J.L., Khachatrian, G.H., Kuregian, M.K.: Nomination of SAFER+ as candidate algorithm for the advanced encryption standard (AES) (1998)
25. Massey, J.L., Khachatrian, G.H., Kuregian, M.K.: Nomination of SAFER++ as candidate algorithm for the new european schemes for signatures, integrity, and encryption (NESSIE) (2000)
26. National Institute of Standards and Technology (NIST): Advanced encryption standard. FIPS Publication 197 (2001), https://csrc.nist.gov/csrc/media/publications/fips/197/final/documents/fips-197.pdf
27. Nielsen, M.A., Chuang, I.: Quantum computation and quantum information (2002)
28. Rötteler, M., Steinwandt, R.: A note on quantum related-key attacks. Inf. Process. Lett. 115(1), 40–44 (2015)
29. Wagner, D.A.: The boomerang attack. In: FSE. Lecture Notes in Computer Science, vol. 1636, pp. 156–170. Springer (1999)
30. Zhandry, M.: A note on the quantum collision and set equality problems. Quantum Inf. Comput. 15(7&8), 557–567 (2015), http://www.rintonpress.com/xxqic15/qic-15-78/0557-0567.pdf

Appendix

A Round Function of SAFER++

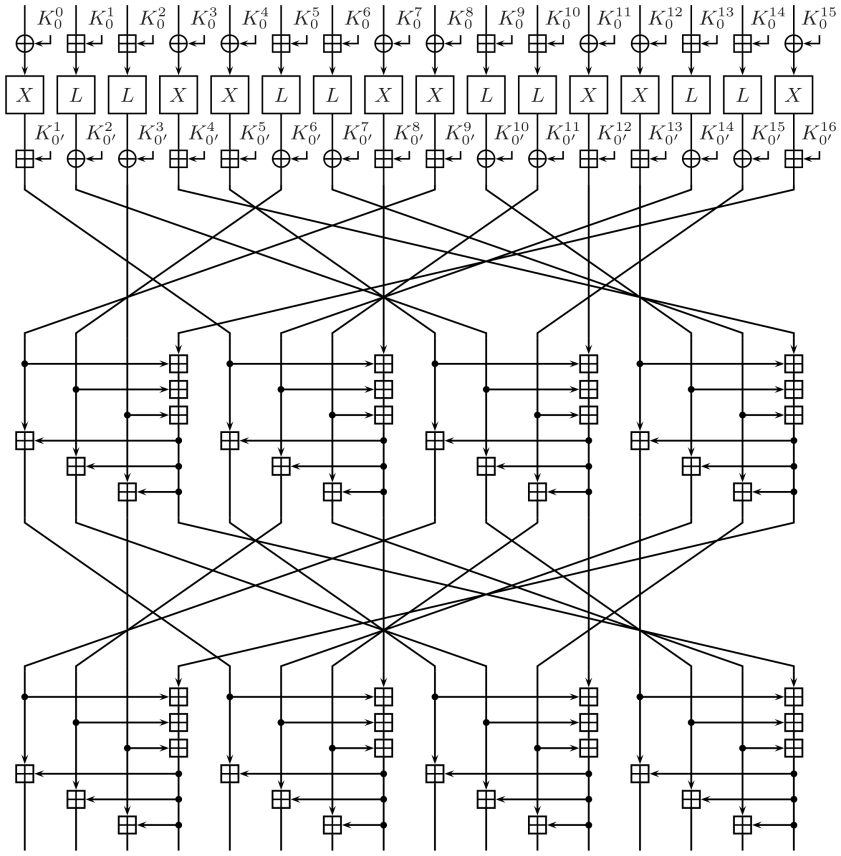

Fig. 4. Round function of SAFER++ (Fig. 1 in [3]).

B Application to Related-key AES

In this section, we present the boomerang related-key attack on AES-256 from [4] and study a corresponding quantum attack. We will see that the classical attack path is actually not adapted to the quantum setting; the quantum attack represents a relatively minor improvement on the classical one, and it can only be considered valid under a restriction on the number of related keys.

Note that the attack uses a tuple of classically related keys (the same as in the classical attack). In this situation, there is no trivial quantum attack such as the one of [28] (which needs a quantum superposition of related keys).

B.1 Description of AES

The AES [26], designed by Daemen and Rijmen [12], was the winner of the AES competition organized by the NIST. It is arguably one of the most analyzed ciphers to date. Similarly to SAFER++, it has blocks of 128 bits, divided in 16 bytes numbered from 0 to 1. The state of AES is represented as a square and we will follow the byte numbering of [4]. Thus, in a plaintext P or a subkey K, the byte $P_{i,j}$ or $K_{i,j}$ is located at row i and column j.

0,0	0,1	0,2	0,3
1,0	1,1	1,2	1,3
2,0	2,1	2,2	2,3
3,0	3,1	3,2	3,3

Fig. 5. AES byte numbering

The round function of AES applies 4 operations: • **AddRoundKey** (ARK), which adds the current round key (as described by the AES key-schedule) and a round constant; • **SubBytes** (SB), which applies the AES S-Box to each byte; • **ShiftRows** (SR), which shifts the bytes in row i of the square by i positions to the left; • **MixColumns** (MC), which applies the (aligned) AES MDS matrix to the columns of the state.

We consider here the related-key attack of [4] on full-round AES-256 (14 rounds in total, numbered from 1 to 14). This key-recovery has a data and time complexity of $2^{99.5}$.

B.2 Classical Attack

There are 4 related keys K_A, K_B, K_C, K_D in the attack of [4], where K_B, K_C, K_D are computed from K_A. These related keys are used in the four branches of the quartet. We refer to [4] for the definition of their relations. The path of the boomerang is represented on Figure 6 and Figure 7 from Appendix C. The key-schedule of AES-256 has an internal state of 256 bits in which the full key is first loaded, then updated every two rounds (the first half is used for even rounds, the second half for odd rounds). We let $K_0, K_1, \ldots$ denote these successive 256-bit round keys, where $K_0 = K$ is the master key.

The main requirement for this attack are the differences `1f` in the output of the active S-Boxes. With a difference `01` in the input, the output difference is `1f` with probability 2^{-6}. Thus, assuming that we can start from a pair that satisfies the path of Figure 6 at Round 1, it passes the top differential to the beginning of

round 9 with probability $2^{-6\times5} = 2^{-30}$. Then we use the *ladder switch* technique, which means that we consider the S-Boxes on the bytes $(0,0),(0,1),(0,2),(0,3)$ to belong to the top part of the boomerang and the other ones to belong to the bottom part of the boomerang, this way there is no differential to pay in either parts of the boomerang for this round. In the bottom part of the boomerang, there remains 3 S-Boxes to pass, with a probability 2^{-18}. Thus the total probability is $2^{-30\times2-18\times2} = 2^{-96}$.

We focus now on the key-recovery attack of [4]. We start from $2^{25.5}$ structures of plaintexts that take all possible values in the 9 bytes $(0,0),(1,0),(2,0),(3,0),(1,1),(1,2),(1,3),(2,2),(3,3)$ (that includes the first column, the second row and the first diagonal) and leave the other constant. For each structure:

1. The plaintexts are encrypted under K_A and K_B and the resulting ciphertexts stored in sets S_A and S_B.
2. A (carefully chosen) difference Δ_C is XORed to S_A and the resulting ciphertexts are decrypted under K_C, yielding a set S_C of plaintexts.
3. The same Δ_C is XORed to S_B and the resulting ciphertexts are decrypted under K_D, yielding a set S_D of plaintexts.
4. S_C and S_D are then filtered for valid quartets: we find all the pairs $P_C \in S_C, P_D \in S_D$ such that $P_C \oplus P_D$ is 0 in the 7 inactive bytes of the structure $((0,1),(0,2),(0,3),(2,1),(3,1),(3,2),(2,3))$.
5. Two filtering steps for these quartets remains. We check whether $P_{A,i,0} \oplus P_{B,i,0} = P_{C,i,0} \oplus P_{D,i,0} (= \Delta K^0_{i,0}), i > 1$ because $\Delta K^0_{i,0}$ is equal for the key pairs (K_A, K_B) and (K_C, K_D).
6. Another filtering allows to reduce the number of quartets by a factor 2^6.

There were 2^{72} plaintexts in each structure, for a total of $2^{97.5}$ plaintexts. There are 2^{144} pairs per structure and a 9-byte condition is required to pass the first round; thus, we can expect $2^{144-8\times9-96+25.5} \simeq 3$ boomerang pairs among them. At the first filtering step, we filter on 7 byte-conditions, in the second, there are two byte-conditions, then a 6-bit condition. Thus there are $2^{144-8\times9-6+25.5} = 2^{91.5}$ quartets at this point.

We can then begin to sieve some subkey bytes. It is shown in [4] that by performing a few trials, one can reduce the number of quartets by a factor 2^{19} and, for each of the remaining quartets, find 2^6 candidates for a total of 15 key bytes. Notably, 85 bits of K_A are found. This makes for a total of $2^{78.5}$ choices for these key bytes. Since there are three boomerang quartets, one of these choices occurs three times (other collisions are unlikely).

B.3 Quantum Attack

The attack of [4] runs in time $2^{99.5}$, makes the same number of queries to E and E^{-1}, and uses a memory of size $2^{77.5}$ to recover the good guess for some of the key bytes. It falls already below the complexity of Grover search for a single secret-key AES-256.

Generic Attack. In the related-key setting, and with an unbounded number of related keys, it is possible to reduce this to $2^{256/3} = 2^{85.3}$ queries and time using the same number of related keys ($2^{85.3}$). This is the quantum counterpart of the classical collision attack using 2^{128} related keys, which would simply encrypt a small set of plaintexts P_i under all the keys $K, K \oplus 1, \ldots, K \oplus 2^{128}$ and find another key K' such that $E_{K'}(P_i)$ collides with one of the $E_{K \oplus j}(P_i)$. The quantum attack uses a smaller set of related keys and uses a Grover search in the second step; the complexities of both steps are balanced at roughly $2^{85.3}$.

More precisely, the quantum search step needs two plaintexts (hence four encryptions per iteration). The first step performs $2 \cdot 2^t$ queries and the second $\frac{\pi}{4} 2^{(256-t)/2}$ iterations, for a total of $\frac{\pi}{4} 2^{(256-t)/2+2} = 2^{128-t/2+2.65}$ encryptions. This is minimized for $t + 1 = 128 - t/2 + 2.65$, i.e., $t = 86.4$ and the total is $2^{87.4}$ encryptions.

Thus, we can choose to compare our attack complexity either with the generic bound of $2^{87.4}$, if we assume an unbounded number of classically related keys, *or* with a bound of 2^{127} if we assume that only 4 related keys are available (in which case the best procedure is the same as above, but limited by the number of keys that we have). In the former case, the attack will not be valid. That is, the complexity will go above $2^{87.4}$. It will only be valid for the second definition.

Attack Procedure. We now detail the quantum version of the previous attack. We use the same differential path and the same key relations. Contrary to the attack against SAFER++, we will not combine the boomerang with a complete recovery of the key. Since only 85 bits of the master key are obtained, it may take actually too much time to continue a Grover search on the remaining key bits; we would then need a more involved procedure to get as many key bits as possible within the allowed time. Instead, we will define a function that produces deterministically a quartet and a 85-bit key guess. Roughly speaking, this function starts from a set of plaintext values (some substructure), performs a search of valid quartets on this space, sieves them to obtain a single (expected) quartet leading to 2^6 guesses of key bits, and chooses one of these guesses. It will output $2^{78.5}$ key guesses of 85 bits; we expect one of these guesses to appear more often, due to the boomerang pairs. Thus we can use Ambainis' element distinctness algorithm [2]. However, we must recognize the only guess that appears three times, not two, so we must use the *3-distinctness* algorithm which runs in generic complexity $\mathcal{O}\left(N^{3/4}\right)$ for N elements (also optimal). The constant in the $\mathcal{O}$ is the same as in the 2-distinctness algorithm.

We compute the time complexity of Algorithm 8 as follows. It is dominated by the $2^{3 \times 78.5/4} \simeq 2^{58.9}$ calls to the function F that we defined. Calling F requires two loops, one of which corresponds actually to a quantum collision search for quartets matching the 9-byte condition. Note that the computations required to find if there are key guesses, and to select one of them, are negligible with

Algorithm 8 Related-key attack on AES-256 (as a classical combination of **Repeat** loops).

Input: access in superposition to encryptions and decryptions under K_A, K_B, K_C, K_D

function $F(i, b)$ ▷ i is a 72.5-bit value, b is a 6-bit value
  Depending on i, choose a set of 2^{19} subsets of $2^{72/3}$ plaintexts P_A
  Repeat 2^{19} **times**
    Choose one of the 2^{19} subsets for P_A
    Create a superposition of quartets passing the filter of Step 4 (56 bits), as follows:
    Create the list of $2^{72/3}$ plaintexts P_A
    By encrypting under K_A and decrypting under K_C, create the set S_C
    Repeat $2^{2\times 72/3}$ **times**
      Create a single plaintext P_B
        ▷ P_B belongs to a space of size $2^{7\times 9} = 2^{72}$, by the choice of P_A
      Encrypt under K_B, decrypt under K_D, obtain P_D
      Check if there is $P_C \in S_C$ such that: $P_C \oplus P_D$ is 0 in the 7 inactive bytes wanted (7 bytes), and the condition at Step 5 is also satisfied (2 bytes)
      If there is, return P_D
    EndRepeat
    Find P_D
    Check if the quartet P_A, P_B, P_C, P_D yields key guesses ▷ probability 2^{-19}
    If it does, return it
  EndRepeat
  At this point, we have a (single) quartet returning key guesses: choose one of them depending on b
end function
Apply Ambainis' 3-distinctness algorithm to F
Measure and return the single 3-collision among the key outputs of F

respect to the other factors.

$$2 \cdot 2^{58.9} \left(2 \left\lfloor \frac{\pi}{4}\sqrt{2^{19}} \right\rfloor \left(2 \cdot 2^{72/3} + 4 \left\lfloor \frac{\pi}{4} 2^{72/3} \right\rfloor + \text{negl.}\right) + \text{negl.}\right) \simeq 2^{59.9} \left(2^{10.15}(2^{25} + 2^{25.65})\right) \simeq 2^{96.4} \quad . \quad (11)$$

The complexity that we obtain is slightly below the classical attack complexity, but not significantly. It is also above the quantum bound of $2^{87.4}$ encryptions. We will use also the same number of superposition queries, and about $2^{58.9}$ quantum random-access memory. Our main issue with this method is that the quantum time complexity of 3-distinctness is different than that of 2-distinctness, which is not the case classically. If the good key was the only one appearing twice, instead of three times, we could replace the term $2^{58.9}$ by $2^{52.3}$ and go more significantly below the classical complexity.

Furthermore, this method (using an element distinctness routine on top of the search of quartets) seems intrinsically less adapted than the direct combination with exhaustive search that we used against SAFER++. Thus, it seems necessary

to modify the path of the attack in order to find more key bytes with a valid quartet, and to finish faster the recovery of the key.

C Paths for the Related-Key Boomerang Attack on AES

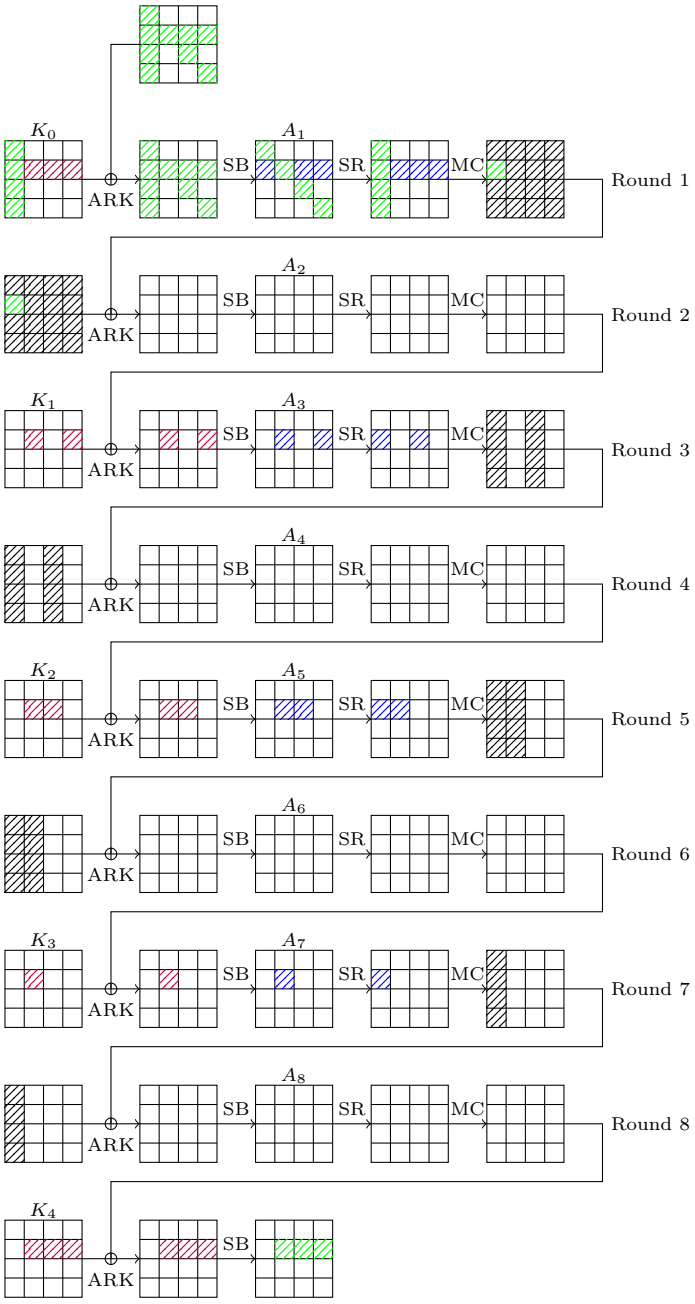

Fig. 6. Top part of the boomerang (see [4], Fig. 7). We use purple for the difference 01 and blue for the difference $1f$. The state after the SubBytes layer of round i is denoted A_i.

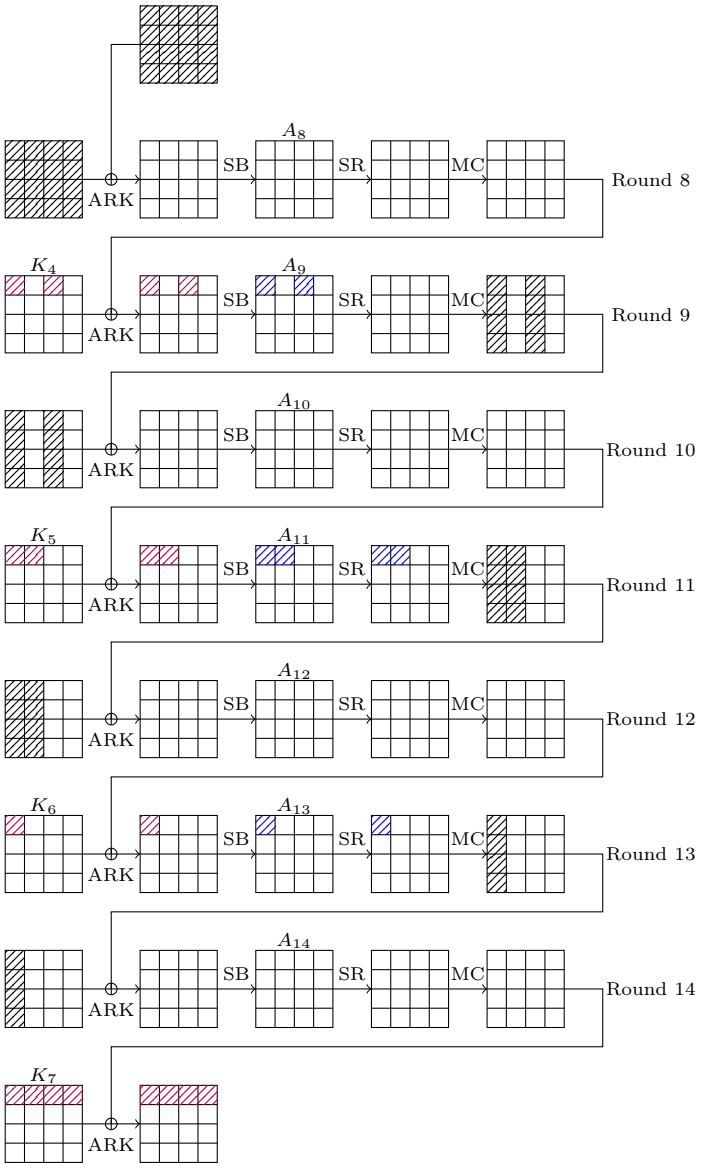

Fig. 7. Bottom part of the boomerang (see [4], Fig. 7). We use purple for the difference 01 and blue for the difference $1f$.