

# Quantum Boomerang Attacks and Some Applications

Paul Frixons<sup>1,2</sup>, María Naya-Plasencia<sup>2</sup>, and André Schrottenloher<sup>3</sup>

<sup>1</sup> Orange Labs, Caen, France

<sup>2</sup> Inria, Paris, France (`firstname.lastname@inria.fr`)

<sup>3</sup> 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 [28]. 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 [22,23,21], while others have studied the security of actual designs [7,20,5]. Many have proposed quantum versions of popular classical attacks, like for instance [21,6].

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 [19] 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 [28]. 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 [7,21], 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 the full version of the paper [18]. We conclude the paper in Section 6.

## 2 Preliminaries

In this section, we introduce the classical Boomerang attack from [28] 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  $\Delta_{\text{in}} \rightarrow \Delta_{\text{out}}$  is a *differential* for  $E$  of probability:

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

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 [28] and the last-round key-recovery attack that can be based on it. The notation that we introduce here  $(E, p, q, \alpha, \dots)$  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:  $\alpha \rightarrow \beta$  for  $E_1$  and  $\delta \rightarrow \gamma$  for  $E_2^{-1}$ .

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

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”

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

figures/boomerang.png

**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  $D_{\text{fin}}$  denote the set of differences that can be obtained from  $\delta$  after the additional rounds,  $p_{\text{out}}$  the probability that we get  $\delta$  back from an element in  $D_{\text{fin}}$  by computing the last rounds backwards, and  $k_{\text{out}}$  the number of key bits involved in these additional rounds. We need  $D_{\text{fin}}$  to be a vector space. This is usually the case as most of the time, non-zero components of  $\delta$  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  $D_{\text{fin}}$ :  $\forall \Delta_1, \Delta_2, (X \oplus \Delta_1) \oplus (X \oplus \Delta_2) = \Delta_1 \oplus \Delta_2 \in D_{\text{fin}}$ .

To simplify for now, we assume that  $\frac{1}{\sqrt{p_{\text{out}} p q}} < |D_{\text{fin}}|$ , in which case a single structure is needed. Otherwise when  $\frac{1}{\sqrt{p_{\text{out}} p q}} \geq |D_{\text{fin}}|$ , 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  $k_{\text{out}}$  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  $\frac{1}{\sqrt{p_{\text{out}} p q}}$  ciphertexts with differences in  $D_{\text{fin}}$ . This first step costs a time and memory:  $\frac{1}{\sqrt{p_{\text{out}} p q}}$ .

Now we will partially decrypt these ciphertexts and obtain pairs with difference  $\delta$ . 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  $K_{\text{out}}$  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  $D_{\text{fin}}$

---

**Algorithm 2** Boomerang last-round attack
 

---

**Input:** oracle access to  $E = R \circ E_2 \circ E_1$   
**Output:** guess of partial key  $K_{\text{out}}$  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:      $L_{\text{pairs}} \leftarrow L_{\text{pairs}} \cup \{(C_i, C_j, C'_i, C'_j)\}$   
 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:     **ForEach** possible value  $K_{\text{out}}$  of the  $k_{\text{out}}$  bits of key ▷  $2^{k_{\text{out}}}$  trials,  $2^{k_{\text{out}}} p_{\text{out}}^2$  solutions  
 10:         If both  $(C_i, C_j)$  and  $(C'_i, C'_j)$  lead to a difference  $\delta$  through  $R^{-1}$  under  $K_{\text{out}}$   
 11:          $L_{\text{triplets}} \leftarrow L_{\text{triplets}} \cup \{(C_i, C_j, K_{\text{out}})\}$   
 12:     **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  $K_{\text{out}}$  in the triplet list

---

with probability 1. Obtaining the list  $L_{\text{pairs}}$  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  $D_{\text{fin}}$  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  $L_{\text{pairs}}$  to ensure that, for the good key guess of  $K_{\text{out}}$ , one of them is an actual boomerang quartet. Let  $C_{\text{out}}$  be the time complexity to obtain all valid keys  $K_{\text{out}}$  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  $L_{\text{pairs}}$  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  $K_{\text{out}}$  such that  $(C_i, C_j)$  and  $(C'_i, C'_j)$  lead to a difference  $\delta$  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  $K_{\text{out}}$  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  $K_{\text{out}}$ . 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:

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

in number of queries to  $E$  and  $E^{-1}$  or evaluations of the cipher. Note that  $C_{\text{out}}$  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.* 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  $K_{\text{out}}$ , as the only one that appears in two (or more) triples in the list  $L_{\text{triplets}}$ .

## 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  $\alpha \rightarrow \beta$  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:

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

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  $p_{\uparrow}$ ,  $\beta$  gets mapped to  $\alpha$ . 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  $D_{\text{start}}$  denote the set of differentials that can lead to  $\alpha$  for the additional rounds  $R$ ,  $p_{\text{in}}$  the probability to get  $\alpha$  from an element of  $D_{\text{start}}$  and  $k_{\text{in}}$  the number of bits of the key involved in the additional rounds.

figures/mixingboomerang.png

**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  $D_{\text{start}}$ . 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:

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

For each pair, we have  $R(P_1) \oplus R(P_2) = \alpha$  with probability  $p_{\text{in}}$ . 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  $D_{\text{start}}$ .

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  $K_{\text{in}}$  (the  $k_{\text{in}}$  bits of key that intervene in  $R$ ) such that  $P_1, P_2$  and  $P_3, P_4$  both lead to a difference  $\alpha$  through  $R$ . The average expected number of possible values for  $K_{\text{in}}$  for each pair is thus  $2^{k_{\text{in}}}(p_{\text{in}})^2$ . We let  $C_{\text{in}}$  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:

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

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

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

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

### 3 Quantum Boomerang Attacks

In this section, we adapt the attacks presented in [Section 2](#) to the quantum setting.

---

**Algorithm 3** Quantum Boomerang Distinguisher

---

**Input:** *superposition* oracle access to  $E = E_2 \circ E_1$ , or a random permutation

**Output:** “E” or “Random”

- 1: **Repeat**  $\frac{\pi}{4}(pq)^{-1}$  **times**
  - 2:     Select  $P_1$  at random
  - 3:     **If**  $E^{-1}(E(P_1) \oplus \delta) = E^{-1}(E(P_1 \oplus \alpha) \oplus \delta) \oplus \alpha$  **then**
  - 4:         the state  $(P_1)$  is “good” **Else** the state  $(P_1)$  is “bad”
  - 5: **EndRepeat**
  - 6: Measure and check if a solution was found
  - 7: **If** it was found **return** “E” **Else Return** “Random”
- 

### 3.1 Quantum Boomerang Distinguisher

Similarly to the differential attacks from [21], 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 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  $k_{\text{out}}$  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  $k_{\text{out}}$  bits of key that the Boomerang property gives. If we can produce the superposition of valid  $K_{\text{out}}$  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  $K_{\text{out}}$ , then tries to complete the key in time  $2^{(k-k_{\text{out}})/2}$ .

In order to produce a possible value of  $K_{\text{out}}$ , 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  $K_{\text{out}}$  such that  $(C_1, C_2)$  and  $(C'_1, C'_2)$  decrypt to the difference  $\delta$  through  $R^{-1}$  under  $K_{\text{out}}$ .

Finding such a valid pair  $C_1, C_2$  is a collision search problem, solved with BHT algorithm. 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](#).



---

**Algorithm 4** Quantum Last-rounds Key-recovery

---

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

- 1: **Amplify**  $(\frac{|D_{\text{fin}}|}{2^n} \frac{2^{k_{\text{out}}} p_{\text{out}}}{(pq)^2})$
- 2: Sample a subkey guess  $K_{\text{out}}$  using:
- 3: **Amplify**  $(\frac{1}{2^{k_{\text{out}}} p_{\text{out}}^2})$
- 4: Use BHT algorithm to find a valid pair  $(C_1, C_2, C'_1, C'_2)$
- 5: **Amplify**  $2^{k_{\text{out}}}$
- 6: Sample a guess of  $K_{\text{out}}$
- 7: **If**  $R^{-1}(C_1) \oplus R^{-1}(C_2) = \delta = R^{-1}(C'_1) \oplus R^{-1}(C'_2)$  then  $K_{\text{out}}$  is “good”
- 8: **EndAmplify**
- 9: **If** we obtained a valid  $K_{\text{out}}$  then  $(C_1, C_2, C'_1, C'_2)$  is “good”
- 10: **EndAmplify**
- 11: Given this subkey guess  $K_{\text{out}}$ :
- 12: **Amplify**  $2^{(k-k_{\text{out}})}$
- 13: Sample a choice of the  $k - k_{\text{out}}$  other key bits, check if the full  $K$  matches
- 14: **EndAmplify**
- 15: **If** we obtained a valid full key  $K$  then  $K_{\text{out}}$  is “good”
- 16: **EndAmplify**
- 17: Measure  $K_{\text{out}}$ , recompute  $K$  and return it

---

**Step 1:** the number of iterations to perform depends on the probability that a subkey guess  $K_{\text{out}}$ , 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  $K_{\text{out}}$  from one of these 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  $K_{\text{out}}$ , 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:

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

The algorithm requires Q2 queries to  $E$  and  $E^{-1}$ , and uses  $(2^n/|D_{\text{fin}}|)^{1/3}$  qRAM due to the use of Ambainis' algorithm (or BHT) to find valid pairs. However, if we can replace the factor  $(2^n/|D_{\text{fin}}|)^{1/3}$  by  $(2^n/|D_{\text{fin}}|)^{1/2}$ , 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  $E(E^{-1}(C) \oplus \alpha) := f(C)$ . 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  $K_{\text{in}}$  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  $K_{\text{in}}$  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.

---

**Algorithm 5** Quantum first-rounds key-recovery
 

---

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

- 1: **Amplify**  $\frac{|D_{\text{start}}|}{2^n} \frac{1}{p_{\uparrow} p_{\downarrow}} 2^{k_{\text{in}}} p_{\text{in}}$
- 2: Sample a subkey guess  $K_{\text{in}}$  using:
- 3: **Amplify**  $1/(2^{k_{\text{in}}} p_{\text{in}}^2)$
- 4: Sample a valid  $(P_1, P_2)$  using:
- 5: **Amplify**  $\frac{2^n}{|D_{\text{start}}|}$
- 6: Sample a pair  $P_1, P_2 = P_1 \oplus \alpha$ , compute  $P_3, P_4$  as in the distinguisher
- 7: **If**  $P_3 \oplus P_4 = \alpha$ , then  $(P_1, P_2)$  is “good”
- 8: **EndAmplify**
- 9: **Amplify**  $2^{k_{\text{in}}}$
- 10: Sample a guess of  $K_{\text{in}}$
- 11: **If**  $R(P_1) \oplus R(P_2) = \alpha = R(P_3) \oplus R(P_4)$ , then  $K_{\text{in}}$  is “good”
- 12: **EndAmplify**
- 13: **If** there is a valid  $K_{\text{in}}$ , then  $(P_1, P_2, K_{\text{in}})$  is “good”
- 14: **EndAmplify**
- 15: Given this subkey guess  $K_{\text{in}}$ :
- 16: **Amplify**  $2^{k-k_{\text{in}}}$
- 17: Sample a choice of the  $k - k_{\text{in}}$  other key bits, check if the full  $K$  matches
- 18: **EndAmplify**
- 19: If the valid full key  $K$  was obtained,  $K_{\text{in}}$  is good
- 20: **EndAmplify**
- 21: Measure  $K_{\text{in}}$ , recompute the full  $K$  and return it

---

**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:

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

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 [24]. 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 [25]. Finally, another variant, SAFER++, was submitted to the NESSIE project [26]. 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 (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  $L$ :

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

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++:

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

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

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

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  $(0, 0, 0, 0, 0, x, -x, 0, \dots)$  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  $X(P), 1 \boxplus X(P)$ , 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.

*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  $\delta$  (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  $(0, x, 0, 0, x, x, 0, 0, 0, 0, 0, 0, -4x, 0, 0, x, -x)$
  - 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: **Amplify**  $2^{26}$
- 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:   **Amplify**  $2^{18}$                    ▷ 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:   **Amplify**  $2^{32}$                    ▷ 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:   **EndAmplify**
- 9:   If a solution was found, return it (and the quartet)
- 10: **EndAmplify**
- 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: **EndAmplify**
- 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:

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

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:

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

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  $\lfloor \frac{\pi}{4} 2^{32} \rfloor$  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 the full version of the paper [18], 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  $\mathcal{O}(N^{3/4})$ , 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. Frixons, P., Naya-Plasencia, M., Schrottenloher, A.: Quantum boomerang attacks and some applications. *IACR Cryptol. ePrint Arch.* p. 060 (2022), <https://eprint.iacr.org/2022/060>
19. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: STOC. pp. 212–219. ACM (1996)
20. 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)
21. Kaplan, M., Leurent, G., Leverrier, A., Naya-Plasencia, M.: Quantum differential and linear cryptanalysis. *IACR Trans. Symmetric Cryptol.* 2016(1), 71–94 (2016)
22. Kuwakado, H., Morii, M.: Quantum distinguisher between the 3-round feistel cipher and the random permutation. In: ISIT. pp. 2682–2685. IEEE (2010)
23. Kuwakado, H., Morii, M.: Security on the quantum-type even-mansour cipher. In: ISITA. pp. 312–316. IEEE (2012)
24. 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)
25. Massey, J.L., Khachatrian, G.H., Kuregian, M.K.: Nomination of SAFER+ as candidate algorithm for the advanced encryption standard (AES) (1998)
26. 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)
27. Nielsen, M.A., Chuang, I.: Quantum computation and quantum information (2002)
28. Wagner, D.A.: The boomerang attack. In: FSE. Lecture Notes in Computer Science, vol. 1636, pp. 156–170. Springer (1999)
29. 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>