Automatic Search for Differential Trails in ARX Ciphers

Alex Biryukov and Vesselin Velichkov

Laboratory of Algorithmics, Cryptology and Security (LACS)
University of Luxembourg
{Alex.Biryukov,Vesselin.Velichkov}@uni.lu

Abstract. We propose a tool[1] for automatic search for differential trails in ARX ciphers. By introducing the concept of a *partial difference distribution table* (pDDT) we extend Matsui's algorithm, originally proposed for DES-like ciphers, to the class of ARX ciphers. To the best of our knowledge this is the first application of Matsui's algorithm to ciphers that do not have S-boxes. The tool is applied to the block ciphers TEA, XTEA, SPECK and RAIDEN. For RAIDEN we find an iterative characteristic on all 32 rounds that can be used to break the full cipher using standard differential cryptanalysis. This is the first cryptanalysis of the cipher in a non-related key setting. Differential trails on 9, 10 and 13 rounds are found for SPECK32, SPECK48 and SPECK64 respectively. The 13 round trail covers half of the total number of rounds. These are the first public results on the security analysis of SPECK. For TEA multiple full (i.e. not truncated) differential trails are reported for the first time, while for XTEA we confirm the previous best known trail reported by Hong et al.. We also show closed formulas for computing the exact additive differential probabilities of the left and right shift operations.

Keywords: symmetric-key, differential trail, tools for cryptanalysis, automatic search, ARX, TEA, XTEA, SPECK, RAIDEN.

1 Introduction

A broad class of symmetric-key cryptographic algorithms are designed by combining a small set of simple operations such as modular addition, bit rotation, bit shift and `XOR`. Although such designs have been proposed as early as the 1980s, only recently the term *ARX* (from *Addition, Rotation, XOR*) was adopted in reference to them.

Some of the more notable examples of ARX algorithms, ordered chronologically by the year of proposal are: the block cipher FEAL [37] (1987), the hash functions MD4 [34] (1990) and MD5 [35] (1992), the block ciphers TEA [40] (1994), RC5 [36] (1994), XTEA [30] (1997), XXTEA [31] (1998) and HIGHT [15] (2006), the stream cipher Salsa20 [4] (2008), the SHA-3 [28] finalists Skein [13]

[1] The source code of the tool is made publicly available as part of a larger toolkit for the analysis of ARX at the following address: `https://github.com/vesselinux/yaarx` .

J. Benaloh (Ed.): CT-RSA 2014, LNCS 8366, pp. 227–250, 2014.

and BLAKE [2] (2011) and the recently proposed hash function for short messages SipHash [1] (2012).

By combining linear (XOR, bit shift, bit rotation) and non-linear (modular addition) operations, and iterating them over multiple rounds, ARX algorithms achieve strong resistance against standard cryptanalysis techniques such as linear [24] and differential [5] cryptanalysis. Additionally, due to the simplicity of the underlying operations, they are typically very fast in software.

Although ARX designs have many advantages and have been widely used for many years now, the methods for their rigorous security analysis are lagging behind. This is especially true when compared to algorithms such as AES [9] and DES [29]. The latter were designed using fundamentally different principles, based on the combination of linear transformations and non-linear substitution tables or S-boxes.

Since a typical S-box operates on 8 or 4-bit words, it is easy to efficiently evaluate its differential (resp. linear) properties by computing its difference distribution table (DDT) (resp. linear approximation table (LAT)). In contrast, ARX algorithms use modular addition as a source of non-linearity, rather than S-boxes. Constructing a DDT or a LAT for this operation for n-bit words would require $2^{3n} \times 4$ bytes of memory and would clearly be infeasible for a typical word size of 32 bits.

In this paper we demonstrate that although the computation of a full DDT for ARX is infeasible, it is still possible to efficiently compute a *partial DDT* containing (a fraction of) all differentials that have probability above a fixed threshold. This is possible due to the fact that the probabilities of XOR (resp. ADD) differentials through the modular addition (resp. XOR) operation are monotonously decreasing with the bit size of the word.

Based on the concept of partial DDT-s we develop a method for automatic search for differential trails in ARX ciphers. It is based on Matsui's branch-and-bound algorithm [23], originally proposed for S-box based ciphers. While other methods for automatic search for differential trails in ARX designs exist in literature [12,25,20] they have been exclusively applied to the analysis of hash functions where the key (the message) is known and can be freely chosen. With the proposed algorithm we address the more general setting of searching for trails in block ciphers, where the key is fixed and unknown to the attacker.

Beside the idea of using partial DDT-s another fundamental concept at the heart of the proposed algorithm is what we refer to as *the highways and country roads analogy*. If we liken the problem of finding high probability differential trails in a cipher to the problem of finding fast routes between two cities on a road map, then differentials that have high probability (w.r.t. a fixed threshold) can be thought of as *highways* and conversely differentials with low probability can be viewed as slow roads or *country roads*. To further extend the analogy, a differential trail for n rounds represents a route between points 1 and n composed of some number of highways and country roads. A search for high probability trails is analogous to searching for a route in which the number of highways is maximized while the number of country roads is minimized.

The differentials from the pDDT are the highways on the road map from the above analogy. Beside those highways, the proposed search algorithm explores also a certain number of country roads (low probability differentials). While the list of highways is computed offline prior to the start of the search, the list of country roads is computed on-demand for each input difference to an intermediate round that is encountered during the search. Of all possible country roads that can be taken at a given point (note that there may be a huge number of them), the algorithm considers only the ones that lead back on a highway. If such are not found, then the shortest country road is taken (resp. the maximum probability transition). This strategy prevents the number of explored routes from exploding and at the same time keeps the total probability of the resulting trail high.

Due to the fact that it uses a partial, rather than the full DDT, our algorithm is not guaranteed to find the best differential trail. However experiments[2] on small word sizes of 11, 14 and 16 bits show that the probabilities of the found trails are within a factor of at most 2^{-3} from the probability of the best one.

We demonstrate the proposed tool on block ciphers TEA [40], XTEA [30], SPECK [3] and RAIDEN [32]. Beside being good representatives of the ARX class of algorithms, these ciphers are of interest also due to the fact that results on full (i.e. not truncated) differential trails on them either do not exist (as is the case for TEA, RAIDEN and SPECK) or are scarce (in the case of XTEA). For TEA specifically, in [16, Sect. 1] the authors admit that *it is difficult to find a good differential characteristic.*

By applying our tool, we are able to find multiple differential characteristics for TEA. They cover between 15 and 18 rounds, depending on the value of the key and have probabilities $\approx 2^{-60}$. The 18 round trail, in particular, has probability $\approx 2^{-63}$ for approx. 2^{116} ($\approx 0.1\%$) of all keys. To put those results in perspective, we note that the best differential attack on TEA covers 17 rounds and is based on an impossible differential [8] while the best attack overall applies zero-correlation cryptanalysis and is on 23 rounds but requires the full codebook [6]. For XTEA, we confirm the best previously known full differential trail based on `XOR` differences [16], but this time it was found in a fully automatic way.

For RAIDEN an iterative characteristic on 3 rounds with probability 2^{-4} is reported. When iterated over all 32 rounds a characteristic with probability 2^{-42} on the full cipher is constructed that can be used to fully break RAIDEN using standard differential cryptanalysis. This is the first analysis of the cipher in a non-related key setting.

We also present results on versions of the recently proposed block cipher SPECK [3] with word sizes 16, 24 and 32 bits resp. SPECK32, SPECK48 and SPECK64. For SPECK64 the best trail found by the tool covers half of the total number of rounds (13 out of 26) and has probability 2^{-58}. The best found trails for 16 and 24 bits cover resp. 9 and 10 rounds out of 22/23 with probabilities resp. 2^{-31} and 2^{-45}.

---

[2] For 11 and 14 bits 50 experiments were performed, while for 16 bits 20 experiments were performed. In each experiment a new fixed key was chosen uniformly at random. More details are provided in Appendix C.1.

Table 1. Maximum number of rounds covered by single (truncated) differential trails used in existing differential attacks on TEA, XTEA, SPECK and RAIDEN compared to the best found trails reported in this paper

Cipher	**Type of Trail**	**#Rounds Covered**	**#Rounds Total**	**Ref.**
TEA	Trunc.	5	64	[26]
	Trunc.	7		[8]
	Trunc.	8		[16,6]
	Full	**18**		**Sect. 6**
XTEA	Trunc.	6	64	[26]
	Trunc.	7		[8]
	Trunc.	8		[16,6]
	Full	14		[16]
	Full	**14**		**Sect. 6**
SPECK32	**Full**	**9**	22	**Sect. 6**
SPECK48	**Full**	**10**	22/23	**Sect. 6**
SPECK64	**Full**	**13**	26/27	**Sect. 6**
RAIDEN	**Full**	**32**	32	**Sect. 6**

In Table 1 we provide a comparison between the number of rounds covered by single (truncated) differential trails used in existing attacks (where applicable) on TEA, XTEA, SPECK and RAIDEN to the number of rounds covered by the trails found with the tool.

An additional contribution is that the paper is the first to report closed formulas for computing the exact additive differential probabilities of the left and right shift operations. These formulas are derived in a similar way as the ones for computing the DP of left and right rotation reported by Daum [11, Sect. 4.1.3]. Note that Fouque et al. [14] have previously analyzed the propagation of additive differences through the shift operations, but not the corresponding differential probabilities.

The outline is as follows. In Sect. 2 we define partial difference distribution tables (pDDT) and present an efficient method for their computation. Our extension of Matsui's algorithm using pDDT, referred to as *threshold search*, is presented in Sect. 3. It is followed by the description of a general methodology for automatic search for differential trails in ARX ciphers with Feistel structure in Sect. 4. A brief description of block ciphers TEA, XTEA, SPECK and RAIDEN is given in Sect. 5. In Sect. 6 we apply our methods to search for differential trails in the studied ciphers and we show the most relevant experimental results. Finally, in Sect. 7 are discussed general problems and limitations arising when studying differential trails in ARX ciphers. Sect. 8 concludes the paper. Proofs of all theorems and propositions and more experimental results are provided in Appendix.

A few words on notation: with $x[i]$ is denoted the i-th bit of x; $x[i:j]$ represents the sequence of bits $x[j], x[j+1], \ldots, x[i] : j \leq i$ where $x[0]$ is the least-significant bit (LSB); x_n denotes the n-bit word x (equivalent to $x[n-1:0]$, but more concise); $\#A$ denotes the number of elements in the set A and $x|y$ is the concatenation of the bit strings x and y.

2 Partial Difference Distribution Tables

In this section as well as in the rest of the paper with xdp^{+} and $\mathrm{adp}^{\oplus}$ are denoted respectively the XOR differential probability (DP) of addition modulo 2^n and the additive DP of XOR. Similarly, the additive differential probability of the operations right bit shift (RSH) and left bit shift (LSH) are denoted resp. with $\mathrm{adp}^{\gg r}$ and $\mathrm{adp}^{\ll r}$. Due to space constrains the formal definition and details on the efficient computation of those probabilities are given in Appendix A and Appendix B.

Definition 1. *A* **partial difference distribution table (pDDT)** *D for the ADD (resp. XOR) operation is a DDT that contains all XOR (resp. ADD) differentials $(\alpha, \beta \rightarrow \gamma)$ whose probabilities are larger than or equal to a pre-defined threshold* $\mathbf{p}_{\mathrm{thres}}$*:*

$$(\alpha, \beta, \gamma) \in D \Longleftrightarrow \mathrm{DP}(\alpha, \beta \rightarrow \gamma) \geq \mathbf{p}_{\mathrm{thres}} \ . \qquad (1)$$

If a DDT contains only a fraction of all differentials that have probability above a pre-defined threshold, it is an **incomplete pDDT**.

The following proposition is crucial for the efficient computation of a pDDT:

Proposition 1. *The DP of ADD and XOR (resp.* xdp^{+} *and* $\mathrm{adp}^{\oplus}$*) are monotonously decreasing with the word size n of the differences α, β, γ:*

$$p_n \leq \ldots \leq p_k \leq p_{k-1} \leq \ldots \leq p_1 \leq p_0 \ , \qquad (2)$$

where $p_k = \mathrm{DP}(\alpha_k, \beta_k \rightarrow \gamma_k)$*,* $n \geq k \geq 1$*,* $p_0 = 1$*, and x_k denotes the k LSB-s of the difference x i.e.* $x_k = x[k-1:0]$*.*

Proof. Appendix D.1.

For xdp^{+}, the proposition follows from the following result by Lipmaa et al. [21]: $\mathrm{xdp}^{+}(\alpha, \beta \rightarrow \gamma) = 2^{-\sum_{i=0}^{n-2} \neg \mathrm{eq}(\alpha[i], \beta[i], \gamma[i])}$, where $\mathrm{eq}(\alpha[i], \beta[i], \gamma[i]) = 1 \Longleftrightarrow \alpha[i] = \beta[i] = \gamma[i]$. Proposition 1 is also true for $\mathrm{adp}^{\oplus}$.

Due to Proposition 1 a recursive procedure for computing a pDDT for a given probability threshold p_{thres} can be defined as follows. Starting at the least-significant (LS) bit position $k = 0$ recursively assign values to bits $\alpha[k]$, $\beta[k]$ and $\gamma[k]$. At every bit position $k : n > k \geq 0$ check if the probability of the partially constructed $(k+1)$-bit differential is still bigger than the threshold i.e. check if $p_k = \mathrm{DP}(\alpha_k, \beta_k \rightarrow \gamma_k) \geq p_{\mathrm{thres}}$ holds. If yes, then proceed to the next bit position, otherwise backtrack and assign other values to $(\alpha[k], \beta[k], \gamma[k])$. This process is repeated recursively until $k = n$, at which point the differential $(\alpha_n, \beta_n \rightarrow \gamma_n)$ is added to the pDDT together with its probability p_n. A pseudo-code of the described procedure is listed in Algorithm 1. The initial values are: $k = 0$, $p_0 = 1$ and $\alpha_0 = \beta_0 = \gamma_0 = \emptyset$.

Algorithm 1. Computation of a pDDT for ADD and XOR.

Input: n, p_{thres}, k, p_k, α_k, β_k, γ_k.
Output: pDDT D: $(\alpha, \beta, \gamma) \in D : \mathrm{DP}(\alpha, \beta \to \gamma) \geq p_{\text{thres}}$.

procedure compute_pddt$(n, p_{\text{thres}}, k, p_k, \alpha_k, \beta_k, \gamma_k)$ **do**
    if $n = k$ **then**
        Add $(\alpha, \beta, \gamma) \leftarrow (\alpha_k, \beta_k, \gamma_k)$ to D
        return
    for $x, y, z \in \{0, 1\}$ **do**
        $\alpha_{k+1} \leftarrow x|\alpha_k, \quad \beta_{k+1} \leftarrow y|\beta_k, \quad \gamma_{k+1} \leftarrow z|\gamma_k \quad .$
        $p_{k+1} = \mathrm{DP}(\alpha_{k+1}, \beta_{k+1} \to \gamma_{k+1})$
        if $p_{k+1} \geq p_{\text{thres}}$ **then**
            compute_pddt$(n, p_{\text{thres}}, k + 1, p_{k+1}, \alpha_{k+1}, \beta_{k+1}, \gamma_{k+1})$

The correctness of Algorithm 1 follows directly from Proposition 1. After successful termination the computed pDDT contains all differentials with probability equal to or larger than the threshold. The complexity of Algorithm 1 depends on the value of the threshold p_{thres}. Some timings for both ADD and XOR differences for different thresholds are provided in Table 2. As can be seen from the data in the table it is infeasible to compute pDDT-s for XOR differences for values of the threshold $p_{\text{thres}} \leq 0.01 = 2^{-6.64}$, while for ADD differences this is still possible, but requires significant time (more than 17 hours).

Table 2. Timings on the computation of pDDT for ADD and XOR on 32-bit words using Algorithm 1. Target machine: Intel® Core™ i7-2600, 3.40GHz CPU, 8GB RAM.

	ADD		XOR	
$\mathbf{p}_{\text{thres}}$	**#elements in pDDT**	**Time**	**#elements in pDDT**	**Time**
0.1	252 940	36 *sec.*	3 951 388	1.23 *min.*
0.07	361 420	37 *sec.*	3 951 388	2.29 *min.*
0.05	3 038 668	5.35 *min.*	167 065 948	44.36 *min.*
0.01	2 715 532 204	17.46 *hours*	$\geq$ 72 589 325 174	$\geq$ 29 *days*

3 Threshold Search

In his paper from 1994 [23] Matsui proposed a practical algorithm for searching for the best differential trail (and linear approximation) for the DES block cipher. The algorithm performs a recursive search for differential trails over a given number of rounds $n \geq 1$. From knowledge of the best probabilities $B_1, B_2, \ldots, B_{n-1}$ for the first $(n - 1)$ rounds and an initial estimate $\overline{B}_n$ for the probability for n rounds it derives the best probability B_n for n rounds. For the estimate the following must hold: $\overline{B}_n \leq B_n$. As already noted, Matsui's algorithm is applicable to block ciphers that have S-boxes. In this section we extend it to the case of ciphers without S-boxes such as ARX by applying the concept of pDDT.

We describe the extended algorithm next. Its description in pseudo-code is listed in Algorithm 2.

In addition to Matsui's notation for the probability of the best n-round trail B_n and of its estimate $\overline{B}_n$ we introduce $\widehat{B}_n$ to denote the probability of *the best found* trail for n rounds: $\overline{B}_n \leq \widehat{B}_n \leq B_n$. Given a pDDT H of size m, an estimation for the best n-round probability $\overline{B}_n$ with its corresponding n-round differential trail $\overline{T}$ and the probabilities $\widehat{B}_1, \widehat{B}_2, \ldots, \widehat{B}_{n-1}$ of the best found trails for the first $n-1$ rounds, Algorithm 2 outputs an n-round trail $\widehat{T}$ that has probability $\widehat{B}_n \geq \overline{B}_n$.

Similarly to Matsui's algorithm, Algorithm 2 operates by recursively extending a trail for i rounds to $(i+1)$ rounds, beginning with $i = 1$ and terminating at $i = n$. The recursion at level i continues to level $(i+1)$ only if the probability of the constructed i-round trail multiplied by the probability of the best found trail for $(n-i)$ rounds is at least $\overline{B}_n$ i.e. if $p_1 p_2 \ldots p_i \widehat{B}_{n-i} \geq \overline{B}_n$. For $i = n$ the last equation is equivalent to: $p_1 p_2 \ldots p_n = \widehat{B}_n \geq \overline{B}_n$. If the latter holds, the initial estimate is updated with the new: $\overline{B}_n \leftarrow \widehat{B}_n$ and the corresponding trail is also updated accordingly: $\overline{T}_n \leftarrow \widehat{T}_n$.

During the search process Algorithm 2 explores multiple differential trails. It is important to stress that the differentials that compose those trails are not restricted to the entries from the initial pDDT H. The latter represent only the starting point of the first two rounds of the search, as in those rounds both the input and the output differences of the round transformation can be freely chosen (due to the specifics of the Feistel structure). From the third round onwards, excluding the last round, beside the entries in H the algorithm explores also an additional set of low-probability differentials stored in a temporary pDDT C and sharing the same input difference.

The table C is computed on demand for each input difference to an intermediate round (any round other than the first two and the last) encountered during the search. All entries in C additionally satisfy the following two conditions: (1) Their probabilities are such that they can still improve the probability of the best found trail for the given number of rounds i.e. if (α_r, β_r, p_r) is an entry in C for round r, then $p_r \geq \overline{B}_n / (p_1 p_2 \cdots p_{r-1} \widehat{B}_{n-r})$; (2) Their structure is such that they guarantee that the input difference for the next round $\alpha_{r+1} = \alpha_{r-1} + \beta_r$ will have a matching entry in H. While the need for condition (1) is self-evident, condition (2) is necessary in order to prevent the exploding of the size of C while at the same time keeping the probability of the resulting trail high. The meaning of the tables H and C is further clarified with the following analogy.

Example 1 (The Highways and Country Roads Analogy). The two tables H and C employed in the search performed by Algorithm 2 can be thought of as lists of highways and country roads on a map. The differentials contained in H have high probabilities w.r.t. to the fixed probability threshold and correspond therefore to fast roads such as *highways*. Analogously, the differentials in C have low probabilities and can be seen as slow roads or *country roads*. To continue this analogy, the problem of finding a high probability differential trail for n rounds can be seen as a problem of finding a fast route between points 1 and n on the

Algorithm 2. Matsui Search for Differential Trails Using pDDT (Threshold Search).

Input: $\mathbf{n}$: number of rounds; $\mathbf{r}$: current round; $\mathbf{H}$: pDDT; $\widehat{\mathbf{B}} = (\widehat{\mathbf{B}}_{\mathbf{1}}, \widehat{\mathbf{B}}_{\mathbf{2}}, \ldots, \widehat{\mathbf{B}}_{\mathbf{n-1}})$: probs. of best found trails for the first $(n-1)$ rounds; $\overline{\mathbf{B}}_{\mathbf{n}} \leq \mathbf{B}_{\mathbf{n}}$: initial estimate; $\overline{\mathbf{T}} = (\overline{\mathbf{T}}_{\mathbf{1}}, \ldots, \overline{\mathbf{T}}_{\mathbf{n}})$: trail for n rounds with prob. $\overline{B}_n$; $\mathbf{p}_{\text{thres}}$: probability threshold.
Output: $\widehat{\mathbf{B}}_{\mathbf{n}}$, $\widehat{\mathbf{T}} = (\widehat{\mathbf{T}}_{\mathbf{1}}, \ldots, \widehat{\mathbf{T}}_{\mathbf{n}})$: trail for n rounds with prob. $\widehat{B}_n : \overline{B}_n \leq \widehat{B}_n \leq B_n$.

procedure threshold_search$(n, r, H, \widehat{B}, \overline{B}_n, \overline{T})$ **do**
  // Process rounds 1 and 2
  if $((r = 1) \vee (r = 2)) \wedge (r \neq n)$ **then**
    for all (α, β, p) in H **do**
      $p_r \leftarrow p, \quad \widehat{B}_n \leftarrow p_1 \cdots p_r \widehat{B}_{n-r}$
      if $\widehat{B}_n \geq \overline{B}_n$ **then**
        $\alpha_r \leftarrow \alpha, \quad \beta_r \leftarrow \beta,$ **add** $\widehat{T}_r \leftarrow (\alpha_r, \beta_r, p_r)$ to $\widehat{T}$
        call **threshold_search**$(n, r+1, H, \widehat{B}, \overline{B}_n, \widehat{T})$
  // Process intermediate rounds
  if $(r > 2) \wedge (r \neq n)$ **then**
    $\alpha_r \leftarrow (\alpha_{r-2} + \beta_{r-1})$; $p_{r,\min} \leftarrow \overline{B}_n / (p_1 p_2 \cdots p_{r-1} \widehat{B}_{n-r})$
    $C \leftarrow \emptyset$ // Initialize the country roads table
    for all $\beta_r : (p_r(\alpha_r \to \beta_r) \geq p_{r,\min}) \wedge ((\alpha_{r-1} + \beta_r) = \gamma \in H)$ **do**
      add (α_r, β_r, p_r) to C // Update country roads table
    if $C = \emptyset$ **then**
      $(\beta_r, p_r) \leftarrow p_r = \max_\beta p(\alpha_r \to \beta)$; **add** (α_r, β_r, p_r) to C
    for all $(\alpha, \beta, p) : \alpha = \alpha_r$ in H and all $(\alpha, \beta, p) \in C$ **do**
      $p_r \leftarrow p, \quad \widehat{B}_n \leftarrow p_1 p_2 \ldots p_r \widehat{B}_{n-r}$
      if $\widehat{B}_n \geq \overline{B}_n$ **then**
        $\beta_r \leftarrow \beta,$ **add** $\widehat{T}_r \leftarrow (\alpha_r, \beta_r, p_r)$ to $\widehat{T}$
        call **threshold_search**$(n, r+1, H, \widehat{B}, \overline{B}_n, \widehat{T})$
  // Process last round
  if $(r = n)$ **then**
    $\alpha_r \leftarrow (\alpha_{r-2} + \beta_{r-1})$
    if $(\alpha_r$ in $H)$ **then**
      $(\beta_r, p_r) \leftarrow p_r = \max_{\beta \in H} p(\alpha_r \to \beta)$ // Select the max. from the highway table
    else
      $(\beta_r, p_r) \leftarrow p_r = \max_\beta p(\alpha_r \to \beta)$ // Compute the max.
      if $p_r \geq p_{\text{thres}}$ **then**
        add (α_r, β_r, p_r) to H
    $p_n \leftarrow p_r, \quad \widehat{B}_n \leftarrow p_1 p_2 \ldots p_n$
    if $\widehat{B}_n \geq \overline{B}_n$ **then**
      $\alpha_n \leftarrow \alpha_r, \quad \beta_n \leftarrow \beta,$ **add** $\widehat{T}_n \leftarrow (\alpha_n, \beta_n, p_n)$ to $\widehat{T}$
      $\overline{B}_n \leftarrow \widehat{B}_n, \quad \overline{T} \leftarrow \widehat{T}$
  $\widehat{B}_n \leftarrow \overline{B}_n, \quad \widehat{T} \leftarrow \overline{T}$ // Update the target bound and the best found trail
  return $\widehat{B}_n, \widehat{T}$

map. Clearly such a route must be composed of as many highways as possible. Condition (2), mentioned above, essentially guarantees that any country road that we may take in our search for a fast route will bring us back on a highway. Note that it is possible that the fastest route contains two or more country roads in sequence. While such a case will be missed by Algorithm 2, it may be accounted for by lowering the initial probability threshold.

Algorithm 2 terminates when the initial estimate $\overline{B}_n$ can not be further improved. The complexity of Algorithm 2 depends on the following factors: (1) the closeness of the best found probabilities $\widehat{B}_1, \widehat{B}_2, \ldots, \widehat{B}_{n-1}$ for the first $(n-1)$ rounds to the actual best probabilities, (2) the tightness of the initial estimate $\overline{B}_n$ and (3) the number of elements m in H. The latter is determined by the probability threshold used to compute H.

4 General Methodology for Automatic Search for Differential Trails in ARX

We describe a general methodology for the automatic search for differential trails in ARX algorithms. In our analysis we restrict ourselves to Feistel ciphers, although the proposed method is applicable to other ARX designs as well.

Let F be the round function (the F-function) of a Feistel cipher E, designed by combining a number of ARX operations, such as `XOR`, `ADD`, bit shift and bit rotation. To search for differential trails for multiple rounds of E perform the following steps:

1. Derive an expression for computing the differential probability (DP) of F for given input and output difference. The computation may be an approximation obtained as the multiplication of the DP of the components of F.
2. Compute a pDDT for F. It can be an incomplete pDDT obtained e.g. by merging the separate pDDT-s of the different components of F.
3. Execute the threshold search algorithm described in Sect.3 with the (incomplete) pDDT computed in Step. 2 as input.

In the following sections we apply the proposed methodology to automatically search for differential trails in several ARX-based block ciphers.

5 Description of TEA, XTEA, SPECK and RAIDEN

The Tiny Encryption Algorithm (TEA) is a block cipher designed by Wheeler and Needham and presented at FSE 1994 [40]. It has a Feistel structure composed of 64 rounds. Each round operates on 64-bit blocks divided into two 32-bit words $L_i, R_i : 0 \leq i \leq 64$, so that $P = L_0|R_0$ is the plaintext and $C = L_{64}|R_{64}$ is the ciphertext. TEA has 128-bit key K composed of four 32-bit words: $K = K_3|K_2|K_1|K_0$. The key schedule is such that the same two key words are used at every second round i.e. K_0, K_1 are used in all odd rounds and K_2, K_3 are used in all even rounds. Additionally, thirty-two 32-bit constants $\delta_r : 1 \leq r < 32$ (the

δ constants) are defined. A different δ constant is used at every second round. The round function F of TEA takes as input a 32-bit value x, two 32-bit key words k_0, k_1 and a round constant δ and produces a 32-bit output $F(x)$. For fixed δ, k_0 and k_1, F is defined as:

$$(\delta, k_0, k_1) : F(x) = ((x \ll 4) + k_0) \oplus (x + \delta) \oplus ((x \gg 5) + k_1) \ . \qquad (3)$$

For fixed round keys $K_j, K_{j+1} : j \in \{0, 2\}$ and round constant δ_r, round i of TEA $(1 \leq i < 64)$ is described as: $L_{i+1} = R_i$, $R_{i+1} = L_i + F(R_i)$.

XTEA is an extended version of TEA proposed in [30] by the same designers. It was designed in order to address two weaknesses of TEA pointed by Kelsey et al. [18]: (1) a related-key attack on the full TEA and (2) the fact that the effective key size of TEA is 126, rather than 128 bits. The structure of XTEA is very similar to the one of TEA: 64-round Feistel network operating on 64-bit blocks using a 128-bit key. The main difference is in the key schedule: at every round XTEA uses one rather than two 32-bit key words from the original key according to a new non-periodic key schedule. Additionally, the number of δ constants is increased from 32 to 64 and thus a different constant is used at every round. The F-function of XTEA is also slightly modified and for a fixed round key k and round constant δ is defined as:

$$(\delta, k) : F(x) = (\delta + k) \oplus (x + ((x \ll 4) \oplus (x \gg 5))) \ . \qquad (4)$$

The F-functions of TEA and XTEA are depicted in Fig. 1.

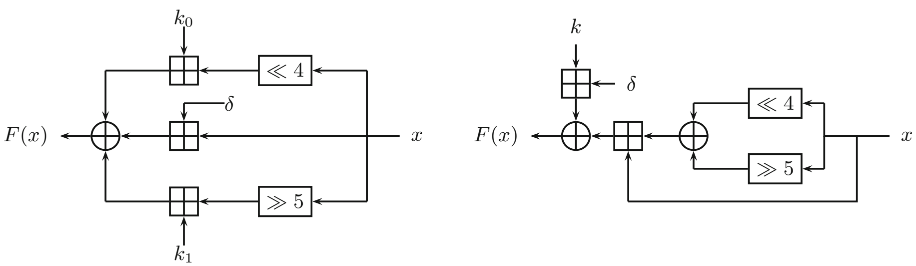

Fig. 1. The F-functions of TEA (left) and XTEA (right)

In [32] Polimón et al. have proposed a variant of TEA called RAIDEN. It has been designed by applying genetic programming algorithms to automatically evolve a highly non-linear round function. The latter is composed of the same operations as TEA (arranged in different order) but is more efficient and has better mixing properties as measured by its avalanche effect. As a result RAIDEN is claimed to be competitive to TEA in terms of security. It has 32 rounds and its round function is:

$$F_k(x) = ((k + x) \ll 9) \oplus (k - x) \oplus ((k + x) \gg 14) \ . \qquad (5)$$

The key k in (5) is updated every second round according to a new key schedule and therefore every two consecutive rounds use the same key. The main differences with TEA are that in (5) the round constant δ is discarded, the shift constants are changed and the shift operations are moved *after* the key addition (see Fig. 2, left). For more details on the RAIDEN cipher we refer the reader to [32]. The only previous security result for RAIDEN is a related-key attack reported in [17].

Most recently, in June 2013, a new family of ARX-based lightweight block ciphers SPECK [3] was proposed by researchers from the National Security Agency (NSA) of the USA. Its design bears strong similarity to Threefish – the block cipher used in the hash function Skein [13]. The round function of SPECK under a fixed round key k is defined on inputs x and y as

$$F_k(x, y) = (f_k(x, y),\ f_k(x, y) \oplus (y \lll t_2)) \ , \qquad (6)$$

where the function $f_k(\cdot, \cdot)$ is defined as $f_k(x, y) = ((x \ggg t_1) + y) \oplus k$. The rotation constants t_1 and t_2 are equal to 7 and 2 resp. for word size $n = 16$ bits and to 8 and 3 for all other word sizes: $24, 32, 48$ and 64. Note that although SPECK is not a Feistel cipher itself, it can be represented as a composition of two Feistel maps as described in [3]. At the time of this writing we are not aware of any published results on the security analysis of SPECK. The round functions of RAIDEN and SPECK are shown in Fig. 2.

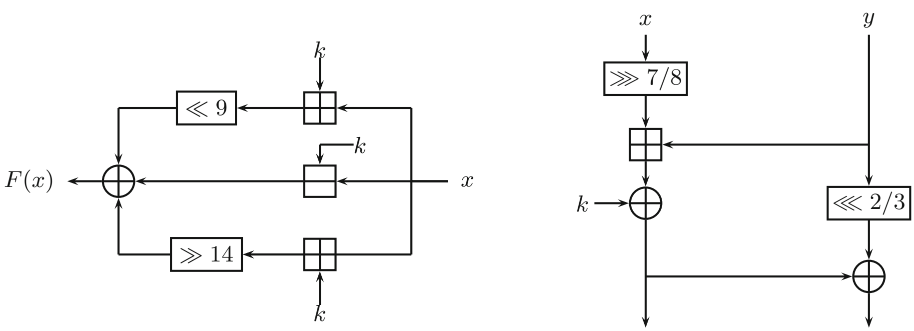

Fig. 2. The F-functions of RAIDEN (left) and SPECK (right)

In Table 1 are listed the maximum number of rounds covered by differential trail/s used in published differential attacks on TEA, XTEA, RAIDEN and SPECK. These results are compared with the best trails found using our method.

6 Automatic Search for Differential Trails

We apply the steps from Sect. 4 to search for differential trails for multiple rounds of the block ciphers described in Sect. 5. We analyze TEA, RAIDEN and SPECK w.r.t. ADD differences and XTEA w.r.t. XOR differences. Additive differences are more appropriate for the differential analysis of the former (as opposed to XOR differences) due to two reasons. First, the round keys and round constants are ADD-ed. Second, the number of ADD vs. XOR operations in one round is larger and therefore more components are linear w.r.t. ADD than to XOR. Similarly, XTEA is more suitably analyzed with XOR differences since the round keys are XOR-ed.

In Table 3 (left) is shown the best found ADD differential trail for 18 rounds of TEA with probability $2^{-62.6}$ and on the right side is shown the best found XOR trail for 14 rounds of XTEA with probability $2^{-60.76}$ confirming a previous result by Hong et al. [16]. Note that while the rule that a country road must be followed by a highway is strictly respected in the trail for TEA, this is not the case for XTEA. For example transitions 6 and 7 in the trail for XTEA have prob. resp. $2^{-5.35}$ and $2^{-5.36}$ both of which are below the threshold $p_{\text{thres}} = 2^{-4.32}$. In those cases no country road that leads back on a highway was found and so the shortest country road was taken (resp. the maximum probability transition for the given input difference was computed: lines 15–16 of Algorithm 2).

The top line of Table 3 shows the fixed values of the keys for which the two trails were found and for which their probabilities were experimentally verified.

Table 3. Differential trails for TEA and XTEA. The leftmost key word is K_0, the next is K_1, etc. #hways lists the number of elements in the pDDT (the highways).

	TEA				XTEA			
key	11CAD84E 96168E6B 704A8B1C 57BBE5D3				E15C838 DC8DBE76 B3BB0110 FFBB0440			
r	β		α	$\log_2 p$	β		α	$\log_2 p$
1	F	←	FFFFFFFF	−3.62	0	←	80402010	−4.61
2	0	←	0	−0.00	80402010	←	0	−3.01
3	F	←	FFFFFFFF	−2.87	80402010	←	80402010	−5.48
4	0	←	F	−7.90	0	←	80402010	−3.30
5	FFFFFFF1	←	FFFFFFFF	−3.60	80402010	←	0	−3.01
6	0	←	0	−0.00	80402010	←	80402010	−5.35
7	FFFFFFF1	←	FFFFFFFF	−2.78	0	←	80402010	−5.36
8	2	←	FFFFFFF1	−8.66	80402010	←	0	−2.99
9	F	←	1	−3.57	80402010	←	80402010	−5.45
10	0	←	0	−0.00	0	←	80402010	−5.42
11	FFFFFFF1	←	1	−2.87	80402010	←	0	−2.99
12	FFFFFFFE	←	FFFFFFF1	−7.90	80402010	←	80402010	−5.38
13	F	←	FFFFFFFF	−3.59	0	←	80402010	−5.40
14	0	←	0	−0.00	80402010	←	0	−2.99
15	11	←	FFFFFFFF	−2.79				
16	0	←	11	−8.83				
17	FFFFFFEF	←	FFFFFFFF	−3.61				
18	0	←	0	−0.00				
$\sum_r \log_2 p_r$				−62.6				−60.76
$\log_2 p_{\text{thres}}$				−4.32				−4.32
#hways				68				474
Time:				21.36 min.				315 min.

The reason to perform the search for a fixed key rather than averaged over all keys is the fact that for TEA the assumption of independent round keys, commonly made in differential cryptanalysis, does not hold. This is a consequence of the simple key schedule of the cipher according to which the same round keys are re-used every second round. Thus a trail that has very good probability computed as an average over all keys, may in fact have zero probability for many or even all keys. This problem is further discussed in Sect. 7.

The mentioned effect is not so strong for XTEA due to the slightly more complex key schedule of the latter. In XTEA, the round keys are re-used according to a non-periodic schedule and, more importantly, a round constant that is different for every round, is added to the key before it is applied to the state (see Fig. 1). In this way the round keys are randomized in every round and thus the traditional differential analysis with probabilities computed as an average over all keys is more appropriate for XTEA.

A major consequence of the key dependency effect discussed above is that while the 14 round trail for XTEA from Table 3 can directly be used in a key-recovery attack, as has indeed been already done in [16], it is not straightforward to do so for the 18 round trail for TEA. The reason is that this trail is valid only for a fraction of all keys. We have estimated the size of this fraction to be approx. $0.098\% \approx 0.1\%$, which is equal to 2^{116} weak keys (note that the effective key size of TEA is 126 bits [18]). The size of the weak key class was computed by observing that only the 9 LS bits of K_2 and the 3 LS bits of K_3 influence the probability of the trail. By fixing those 12 bits to the corresponding bits of the key values in Table 3 (resp. `0x11C` and `0x3`), we have experimentally verified that for any assignment of the remaning 116 bits of the key the 18 round trail has probability $\approx 2^{-63}$. Note that other assignments of the relevant 12 bits may also be possible and therefore the size of the weak key class may be actually bigger.

While the fixed-key trails for TEA found by the threshold search algorithm may have limited use for an attacker due to the reasons discussed above, they already provide very useful information for a designer. By running Algorithm 2 for many fixed keys we saw that the best found trails typically cover between 15 and 17 rounds and in more rare cases 18 rounds. If this information has been available to the designers of TEA at the time of the design, they may have considered reducing the total number of rounds from 64 to 32 or less. Similarly, the threshold search algorithm can be used in order to estimate the security of new ARX designs and to help to select the appropriate number of rounds accordingly.

Comparisons of the trails found with the tool to the actual best trails on TEA with reduced word size of 11 and 16 bits are shown in Appendix C.1.

After applying the threshold search to RAIDEN the best characteristic that was found is iterative with period 3 with probability 2^{-4} (shown in Table 4). By iterating it for 32 rounds we construct a charactersistic with probability 2^{-42}. The latter can be used in a standard differential attack on the full cipher under a non related-key setting. Note that in contrast to TEA, the probabilities of the

Table 4. Three round iterative characteristic for RAIDEN beginning at round i

r	β		α	$\log_2 p$
i	0	$\leftarrow$	0	-0
$i+1$	7FFFFF00	$\leftarrow$	7FFFFF00	-2
$i+2$	80000100	$\leftarrow$	7FFFFF00	-2
...	...	$\leftarrow$	0	-0
$\sum_r \log_2 p_r$				-4

reported differentials for RAIDEN are independent of the round key due to the fact that the shift operations are moved *after* the key addition.

We applied the threshold search algorithm using XOR differences to three instances of block cipher SPECK with 16, 24 and 32 bit word sizes respectively. The best trail found for the 32-bit version covers half of the rounds (13 out of 26) and has probability 2^{-58} while the best found trails for 16 and 24 bits cover resp. 9 and 10 rounds out of 22/23 and have probabilities resp. 2^{-31} and 2^{-45}. All trails are shown in Table 5.

Table 5. Differential trails for Speck32, Speck48 and Speck64. #hways lists the number of elements in the pDDT (the highways).

	Speck32			Speck48			Speck64		
r	Δ_L	Δ_R	$\log_2 p$	Δ_L	Δ_R	$\log_2 p$	Δ_L	Δ_R	$\log_2 p$
0	A60	4205	-0	88A	484008	-0	802490	10800004	-0
1	211	A04	-5	424000	4042	-5	80808020	4808000	-5
2	2800	10	-4	202	20012	-4	24000080	40080	-5
3	40	0	-2	10	100080	-3	80200080	80000480	-3
4	8000	8000	-0	80	800480	-2	802480	800084	-4
5	8100	8102	-1	480	2084	-2	808080A0	84808480	-5
6	8000	840A	-2	802080	8124A0	-3	24000400	42004	-6
7	850A	9520	-4	A480	98184	-6	202000	12020	-4
8	802A	D4A8	-6	888020	C48C00	-7	10000	80100	-3
9	A8	520B	-7	240480	6486	-7	80000	480800	-2
10				800082	8324B2	-6	480000	2084000	-3
11							2080800	124A0800	-4
12							12480008	80184008	-7
13							880A0808	88C8084C	-7
$\sum_r \log_2 p_r$			-31			-45			-58
$\log_2 p_{\text{thres}}$			-5.00			-5.00			-5.00
#hways			2^{30}			2^{30}			2^{30}
Time:			≈ 240 min.			≈ 400 min.			≈ 500 min.

7 Difficulties, Limitations and Common Problems

In this section we discuss the common problems and difficulties encountered when studying differential trails in ARX ciphers. This discussion is also naturally related to the limitations of the methodology proposed in Sect. 4. Although below we often use the TEA block cipher as an example, our observations are general and are therefore applicable to a broader class of ARX algorithms.

Accuracy of the Approximation of the DP of F. The first step in the methodology presented in Sect. 4 is to derive an expression for computing the DP of the F-function of the target cipher. Since it is often difficult to efficiently compute the exact probability, this expression would usually be an approximation obtained as the multiplication of the DP of the separate components of F. The probability computed in this way will often deviate from the actual value due to the dependency between the inputs of the different components. Indeed, this phenomenon is well-known and has been studied before e.g. in [38]. The mentioned problem can be addressed with experimental *re-adjustment* of the probability by evaluating the F-function over a number of random chosen input pairs satisfying the input difference.

Dependency of the DP of F on the Round Keys. Another difficulty arises from the fact that in some cases the DP of the F-function is dependent on the value of the round key(s). Ciphers for which this is the case are *not* key-alternating ciphers (cf. [10, Definition 2]) and are typically harder to analyze. The block cipher TEA is an example of a non-key-alternating cipher. The DP of its F-function is key-dependent w.r.t. both `XOR` and `ADD` differences. A solution to the problem of key-dependency of the DP of the F-function is to search for differential trails with probabilities computed for (multiple) fixed keys rather than for trails with probabilities averaged over all keys. As discussed in Sect. 6, this is the approach that we took in the analysis of TEA.

Dependency Between the Round Keys. In differential cryptanalysis of keyed primitives it is common practice to assume that the round keys are independent [19]. This is known as *making the hypothesis of independent round keys* [10]. In ciphers with weak key schedule such as TEA the hypothesis of independent round keys does not hold. As a consequence, obtaining an accurate estimation of the expected probabilities of differential trails in such ciphers is difficult. A possible solution to the dependent round keys problem is to analyze the cipher with respect to a set of randomly chosen fixed keys and consider the minimum probability, among all keys within the set (rather than the expected probabilities averaged over all keys). The reason to select the minimum probability is to guarantee that the resulting differential trail is possible (i.e. has non-zero probability) for every key in the set.

Influence of the Round Constants. Fixed constants are commonly used in the design of symmetric-key primitives in order to destroy similarities between the rounds. Since they are typically added to the state by applying the same operation as for the round keys, it is generally assumed that constants influence neither the probabilities nor the structure of differential trails and hence can be safely ignored. Surprisingly, this assumption does not hold for TEA and possibly for other ARX constructions as well. After modifying TEA to use the same δ constant at every round, for many keys the best found trail after several rounds eventually becomes iterative with period 2 and of the form $(\alpha \to 0), (0 \to 0), (\alpha \to 0), \ldots$. The difference that maximizes the probability of the differential $(\alpha \to 0)$ is $\alpha = \texttt{0xF}$ and has probability 2^{-8} for exactly $6 \cdot 2^{59} \approx 2^{61.6}$ keys (approx. 10% of all keys). We use the two-round iterative trail $(\texttt{0xF} \to 0), (0 \to 0)$ to construct a trail over 15 rounds with probability 2^{-56}. We also found a 4-round iterative pattern with probability $< 2^{-15}$ which holds for a smaller number of key and is used to construct a trail with probability $2^{-61.36}$ on 18 rounds of the modified TEA.

8 Conclusions and Future Work

In this paper we proposed the first extension of Matsui's algorithm for automatic search for differential trails, originally proposed for S-box based ciphers, to the class of ARX ciphers. We used the block ciphers TEA, XTEA, RAIDEN and SPECK as a testbed for demonstrating the practical application of this method.

Using the proposed algorithm, the first full (i.e. not truncated) differential trails for block cipher TEA were found. The best one covers 18 rounds which is one round more than the best differential attack on TEA (17 rounds) and significantly improves the best previously known truncated trail which is on 8 rounds. Trails on 9, 10 and 13 rounds of SPECK32, SPECK48 and SPECK64 resp. were also reported. They represent the first public security analysis of the cipher. For RAIDEN, a trail on all 32 rounds was shown that can be used to break the full cipher. The best trail for XTEA found by the tool confirms the previous known best trail, but this time it was found in a fully automatic way.

For future work, an important problem on the theoretical side would be to compute a bound on how far the probabilities of the best found trails can be from the actual best trail in terms of the fixed probability threshold. On the practical side it would be interesting to extend the algorithm to search for differentials rather than characteristics. Applying the tool to other ARX constructions is another natural direction for future work.

Acknowledgments. We thank our colleagues from the laboratory of algorithmics, cryptology and security (LACS) at the university of Luxembourg for the useful discussions, and especially Yann Le Corre for his help with visualizing the experimental data. We further thank the anonymous reviewers for their time and helpful comments. Some of the experiments presented in this paper were carried out using the HPC facility of the University of Luxembourg.

References

1. Aumasson, J.-P., Bernstein, D.J.: SipHash: a fast short-input PRF. IACR Cryptology ePrint Archive, 2012:351 (2012)
2. Aumasson, J.-P., Henzen, L., Meier, W., Phan, R.C.-W.: SHA-3 proposal BLAKE. Submission to the NIST SHA-3 Competition, Round 2 (2008)
3. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK Families of Lightweight Block Ciphers. Cryptology ePrint Archive, Report 2013/404 (2013), http://eprint.iacr.org/
4. Bernstein, D.J.: The Salsa20 Family of Stream Ciphers. In: Robshaw, M., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 84–97. Springer, Heidelberg (2008)
5. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. J. Cryptology 4(1), 3–72 (1991)
6. Bogdanov, A., Wang, M.: Zero Correlation Linear Cryptanalysis with Reduced Data Complexity. In: Canteaut [7], pp. 29–48
7. Canteaut, A. (ed.): FSE 2012. LNCS, vol. 7549. Springer, Heidelberg (2012)
8. Chen, J., Wang, M., Preneel, B.: Impossible Differential Cryptanalysis of the Lightweight Block Ciphers TEA, XTEA and HIGHT. In: Mitrokotsa, A., Vaudenay, S. (eds.) AFRICACRYPT 2012. LNCS, vol. 7374, pp. 117–137. Springer, Heidelberg (2012)
9. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer (2002)
10. Daemen, J., Rijmen, V.: Probability distributions of Correlation and Differentials in Block Ciphers. IACR Cryptology ePrint Archive, 2005:212 (2005)
11. Daum, M.: Cryptanalysis of Hash Functions of the MD4-Family. PhD thesis, Ruhr-Universität Bochum (2005)
12. De Cannière, C., Rechberger, C.: Finding SHA-1 Characteristics: General Results and Applications. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 1–20. Springer, Heidelberg (2006)
13. Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., Walker, J.: The Skein Hash Function Family. Submission to the NIST SHA-3 Competition, Round 2 (2009)
14. Fouque, P.-A., Leurent, G., Nguyen, P.Q.: Automatic Search of Differential Path in MD4. IACR Cryptology ePrint Archive, 2007:206 (2007)
15. Hong, D., Sung, J., Hong, S., Lim, J., Lee, S., Koo, B., Lee, C., Chang, D., Lee, J., Jeong, K., Kim, H., Kim, J., Chee, S.: Hight: A new block cipher suitable for low-resource device. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 46–59. Springer, Heidelberg (2006)
16. Hong, S., Hong, D., Ko, Y., Chang, D., Lee, W., Lee, S.: Differential Cryptanalysis of TEA and XTEA. In: Lim, J.-I., Lee, D.-H. (eds.) ICISC 2003. LNCS, vol. 2971, pp. 402–417. Springer, Heidelberg (2004)
17. Karroumi, M., Malherbe, C.: Related-key cryptanalysis of raiden. In: International Conference on Network and Service Security, N2S 2009, pp. 1–5 (2009)
18. Kelsey, J., Schneier, B., Wagner, D.: Related-key cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA. In: Han, Y., Quing, S. (eds.) ICICS 1997. LNCS, vol. 1334, pp. 233–246. Springer, Heidelberg (1997)

19. Lai, X., Massey, J.L.: Markov ciphers and differential cryptanalysis. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 17–38. Springer, Heidelberg (1991)
20. Leurent, G.: Construction of Differential Characteristics in ARX Designs - Application to Skein. IACR Cryptology ePrint Archive, 2012:668 (2012)
21. Lipmaa, H., Moriai, S.: Efficient Algorithms for Computing Differential Properties of Addition. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 336–350. Springer, Heidelberg (2002)
22. Lipmaa, H., Wallén, J., Dumas, P.: On the Additive Differential Probability of Exclusive-Or. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 317–331. Springer, Heidelberg (2004)
23. Matsui, M.: On Correlation between the Order of S-Boxes and the Strength of DES. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 366–375. Springer, Heidelberg (1995)
24. Matsui, M., Yamagishi, A.: A New Method for Known Plaintext Attack of FEAL Cipher. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 81–91. Springer, Heidelberg (1993)
25. Mendel, F., Nad, T., Schläffer, M.: Finding SHA-2 Characteristics: Searching through a Minefield of Contradictions. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 288–307. Springer, Heidelberg (2011)
26. Moon, D., Hwang, K., Lee, W.I., Lee, S.-J., Lim, J.-I.: Impossible Differential Cryptanalysis of Reduced Round XTEA and TEA. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 49–60. Springer, Heidelberg (2002)
27. Mouha, N., Velichkov, V., De Cannière, C., Preneel, B.: The Differential Analysis of S-Functions. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2010. LNCS, vol. 6544, pp. 36–56. Springer, Heidelberg (2011)
28. National Institute of Standards and Technology. Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family. Federal Register 27(212), 62212–62220 (November 2007), `http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf` (October 17, 2008)
29. National Institute of Standards, U.S. Department of Commerce. FIPS 47: Data Encryption Standard (1977)
30. Needham, R.M., Wheeler, D.J.: TEA extensions. Computer Laboratory, Cambridge University, England (1997), `http://www.movable-type.co.uk/scripts/xtea.pdf`
31. Needham, R.M., Wheeler, D.J.: Correction to XTEA. Technical report, University of Cambridge (October 1998)
32. Polimón, J., Castro, J.C.H., Estévez-Tapiador, J.M., Ribagorda, A.: Automated design of a lightweight block cipher with Genetic Programming. KES Journal 12(1), 3–14 (2008)
33. Preneel, B. (ed.): FSE 1994. LNCS, vol. 1008. Springer, Heidelberg (1995)
34. Rivest, R.L.: The MD4 Message Digest Algorithm. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 303–311. Springer, Heidelberg (1991)
35. Rivest, R.L.: The MD5 Message-Digest Algorithm. RFC 1321 (April 1992)
36. Rivest, R.L.: The RC5 Encryption Algorithm. In: Preneel [33], pp. 86–96

37. Shimizu, A., Miyaguchi, S.: Fast Data Encipherment Algorithm FEAL. In: Price, W.L., Chaum, D. (eds.) EUROCRYPT 1987. LNCS, vol. 304, pp. 267–278. Springer, Heidelberg (1988)
38. Velichkov, V., Mouha, N., De Cannière, C., Preneel, B.: The Additive Differential Probability of ARX. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 342–358. Springer, Heidelberg (2011)
39. Velichkov, V., Mouha, N., Preneel, C.D.B.: UNAF: A Special Set of Additive Differences with Application to the Differential Analysis of ARX. In: Canteaut, [7] pp. 287–305
40. Wheeler, D.J., Needham, R.M.: TEA, a Tiny Encryption Algorithm. In: Preneel [33], pp. 363–366

A The Differential Probabilities of ADD and XOR

In this section we recall the definitions of the differential probabilities of the operations XOR and modular addition. Before we begin – a brief remark on notation: in the same way as XOR is used to denote both the XOR operation and an XOR difference, we use ADD to denote both the modular addition operation and an additive difference.

Definition 2. *Let α, β and γ be fixed n-bit XOR differences. The XOR differential probability (DP) of addition modulo 2^n (xdp^+) is the probability with which α and β propagate to γ through the ADD operation, computed over all pairs of n-bit inputs (x, y):*

$$\mathrm{xdp}^+(\alpha, \beta \to \gamma) = 2^{-2n} \cdot \#\{(x, y) : ((x \oplus \alpha) + (y \oplus \beta)) \oplus (x + y) = \gamma\} \ . \quad (7)$$

The dual of xdp^+ is the probability $\mathrm{adp}^\oplus$ and is defined analogously:

Definition 3. *Let α, β and γ be fixed n-bit ADD differences. The additive DP of XOR ($\mathrm{adp}^\oplus$) is the probability with which α and β propagate to γ through the XOR operation, computed over all pairs of n-bit inputs (x, y):*

$$\mathrm{adp}^\oplus(\alpha, \beta \to \gamma) = 2^{-2n} \cdot \#\{(x, y) : ((x + \alpha) \oplus (y + \beta)) - (x + y) = \gamma\} \ . \quad (8)$$

The probabilities xdp^+ and $\mathrm{adp}^\oplus$ have been studied in [21] and [22] respectively, where methods for their efficient computation have been proposed. In [21] is also described an efficient algorithm for the computation of xdp^+ maximized over all output differences: $\max_\gamma \mathrm{xdp}^+(\alpha, \beta \to \gamma)$. In [27] the methods for the computation of xdp^+ and $\mathrm{adp}^\oplus$ are further generalized using the concept of S-functions. Finally, in [39, Appendix C, Algorithm 1] a general algorithm for computing the maximum probability output difference for certain types of differences and operations is described. It is applicable to both $\max_\gamma \mathrm{xdp}^+(\alpha, \beta \to \gamma)$ and $\max_\gamma \mathrm{adp}^\oplus(\alpha, \beta \to \gamma)$.

B The Additive DP of Left and Right Shift

Definition 4. *For fixed input and output* ADD *differences resp. α and β, the additive differential probability of the operation* **right bit shift** *(*RSH*) by r positions is defined over all n-bit $(n \geq r)$ inputs x as:*

$$\mathrm{adp}^{\gg r}(\alpha \to \beta) = 2^{-n} \cdot \#\{x : ((x+\alpha) \gg r) - (x \gg r) = \beta\} \ . \tag{9}$$

Analogously, the additive differential probability of the operation **left bit shift** *(*LSH*) by r positions is defined as in (9) after replacing $\gg r$ with $\ll r$.*

Theorem 1. *The* LSH *operation is linear with respect to* ADD *differences i.e. $((x+\alpha) \ll r) - (x \ll r) = (\alpha \ll r)$, where x, α and r are as in Definition 4. It follows that*

$$\mathrm{adp}^{\ll r}(\alpha \to \beta) = \begin{cases} 1 \ , & \text{if } (\beta = \alpha \ll r) \ , \\ 0 \ , & \text{otherwise} \ . \end{cases} \ . \tag{10}$$

Proof. Appendix D.2.

In contrast to LSH, the RSH operation is not linear w.r.t. ADD differences. The following theorem provides expressions for the computation of $\mathrm{adp}^{\gg r}$.

Theorem 2. *Let α be a fixed n-bit input* ADD *difference to an* RSH *operation with shift constant $r \leq n$. Then there are exactly four possibilities for the output difference β. The four differences together with their corresponding probabilities computed over all n-bit inputs are:*

$$\mathrm{adp}^{\gg r}(\alpha \to \beta) = \begin{cases} 2^{-n}(2^{n-r} - \alpha_{\mathrm{L}})(2^r - \alpha_{\mathrm{R}}) \ , & \beta = (\alpha \gg r) \ , \\ 2^{-n}\alpha_{\mathrm{L}}(2^r - \alpha_{\mathrm{R}}) \ , & \beta = (\alpha \gg r) - 2^{n-r} \ , \\ 2^{-n}\alpha_{\mathrm{R}}(2^{n-r} - \alpha_{\mathrm{L}} - 1) \ , & \beta = (\alpha \gg r) + 1 \ , \\ 2^{-n}(\alpha_{\mathrm{L}} + 1)\alpha_{\mathrm{R}} \ , & \beta = (\alpha \gg r) - 2^{n-r} + 1 \ . \end{cases} \ , \tag{11}$$

where α_{L} and α_{R} denote respectively the $(n-r)$ most-significant (MS) bits and the r least-significant (LS) bits of α so that: $\alpha = \alpha_{\mathrm{L}} 2^r + \alpha_{\mathrm{R}}$ and additions and subtractions are performed modulo 2^n. If $\alpha : \beta = \beta_i = \beta_j$ for some $0 \leq i \neq j < 4$ then $\mathrm{adp}^{\gg r}(\alpha \to \beta) = \mathrm{adp}^{\gg r}(\alpha \to \beta_i) + \mathrm{adp}^{\gg r}(\alpha \to \beta_j)$.

Proof. Appendix D.3.

C More Experimental results

C.1 Threshold Search on TEA with Reduced Word Size

In Fig. 3 and Fig. 4 are compared the probabilities of the best trails found by the threshold search algorithm using pDDT to the actual best trails found by applying Matsui's search using full DDT on TEA with word size reduced to 11 and 16 bits respectively. For 11 bits 50 experiments are performed and in

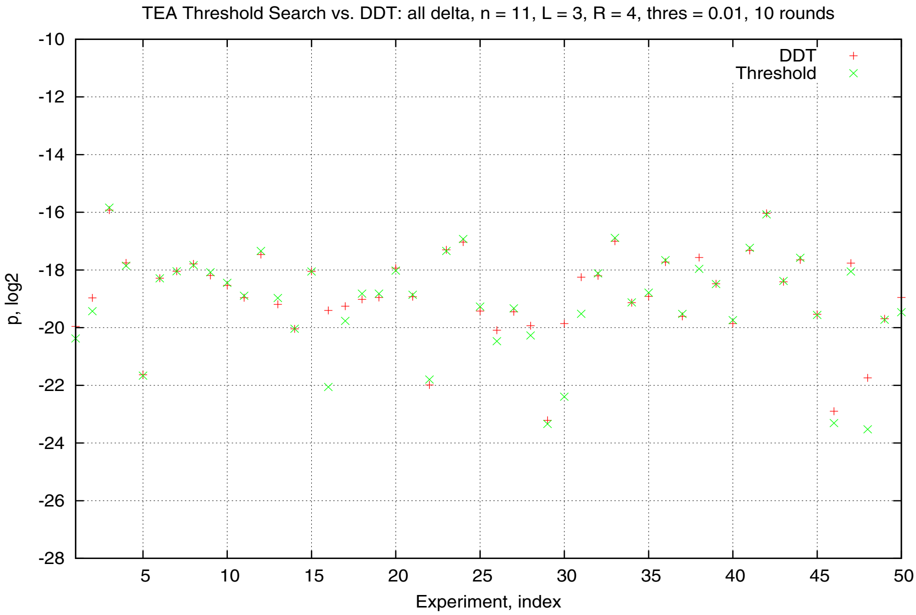

Fig. 3. Threshold Search vs. DDT Search: word size $n = 11$ bits

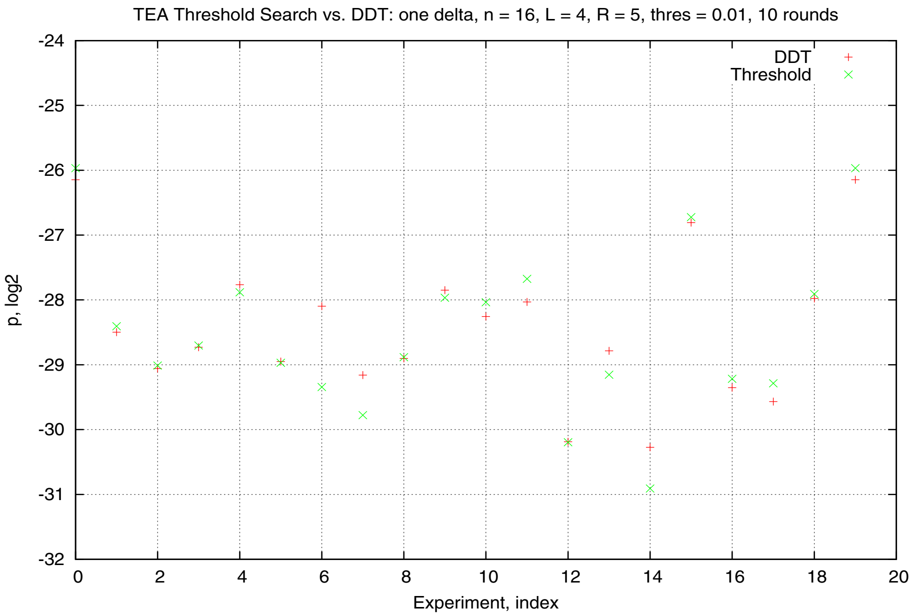

Fig. 4. Threshold Search vs. DDT Search: word size $n = 16$ bits; same δ is used in every round

each experiment a new fixed key is chosen uniformly at random. For 16 bits, the number of experiments is 20. In the experiments on 16 bits the same δ constant (equal to the initial value) was used in every round. The reason is that if different constants are used, then a separate DDT has to be computed for every round, which for more than a couple of rounds quickly becomes infeasible. Also note that for 16 bits it takes longer to compute the full DDT-s due to their larger size (compared to the 11 bit case). The memory consumption is also much bigger – 320 GB of RAM are required to store all DDT-s. Due to the mentioned limitations, less number of experiments on 16 bits were performed.

D Proofs

D.1 Proof of Proposition 1

Proof. We shall prove the proposition for $\mathrm{adp}^{\oplus}$. In this case α, β and γ are `ADD` differences propagating through the `XOR` operation. The proof for xdp^{+} is analogous.

We induct over the word size n. The proposition is trivially true for the base case $n = 1$: $p_1 \leq p_0 = 1$. Let $n = k > 1$. We have to prove that $p_k \leq p_{k-1}$.

Let x and y be n-bit integers. Define L_i to be the set of i-bit pairs (x_i, y_i) that satisfy the differential $(\alpha_i, \beta_i \to \gamma_i)$ for the operation addition modulo 2^i:

$$L_i = \{(x_i, y_i) : ((x_i + \alpha_i) \oplus (y_i + \beta_i)) - (x_i + y_i) = \gamma_i\}, \quad n \geq i \geq 1 \ . \tag{12}$$

Let $l_i = \#L_i$. By definition $p_k = l_k/2^{2k}$ and $p_{k-1} = l_{k-1}/2^{2(k-1)}$ (cf. (8)). Note that every element of L_k can be obtained from an element $(x_{k-1},\ y_{k-1})$ of L_{k-1} by appending bits $x[k-1]$ and $y[k-1]$ to x_{k-1} and y_{k-1} respectively. Assume that this is not true i.e. assume:

$$\begin{aligned} \exists x_k, y_k : \quad & (x_k = x[k-1]|x_{k-1},\ y_k = y[k-1]|y_{k-1},\ (x_k, y_k) \in L_k) \wedge \\ & ((x_{k-1},\ y_{k-1}) \notin L_{k-1}) \ . \end{aligned} \tag{13}$$

If (13) is true then we can construct a new set $L^*_{k-1} = (x_{k-1}, y_{k-1}) \cup L_{k-1}$. Its size is $l^*_{k-1} = l_{k-1} + 1$ and so $p_{k-1} = l^*_{k-1}/2^{2(k-1)}$. The latter differs from the actual value of the probability $p_{k-1} = l_{k-1}/2^{2(k-1)}$ and therefore the assumption (13) is false. Thus $\forall (x_k,\ y_k) \in L_k : (x_{k-1},\ y_{k-1}) \in L_{k-1}$. Because $\#\{(x[k], y[k])\} = 2^2$, the size of L_k can be at most 2^2 times bigger than the size of L_{k-1}:

$$l_k \leq 2^2 l_{k-1} \Rightarrow \frac{l_k}{2^{2k}} \leq \frac{l_{k-1}}{2^{2(k-1)}} \Rightarrow p_k \leq p_{k-1} \ . \tag{14}$$

□

D.2 Proof of Theorem 1

Proof. Let x be an n-bit input to `LSH` with shift constant $r \leq n$. Let $x_{\mathrm{L}}, x_{\mathrm{R}}$: $x = x_{\mathrm{L}} 2^{n-r} + x_{\mathrm{R}}$. Then $(x \ll r) = x_{\mathrm{R}} 2^r$. Similarly, for the input `ADD` difference

α let $\alpha_{\mathrm{L}}, \alpha_{\mathrm{R}} : \alpha = \alpha_{\mathrm{L}} 2^{n-r} + \alpha_{\mathrm{R}}$ and thus $(\alpha \lll r) = \alpha_{\mathrm{R}} 2^r$. The sum $(x + \alpha)$ can then be represented as:

$$\begin{aligned}(x+\alpha) &= (x_{\mathrm{L}} + \alpha_{\mathrm{L}})2^{n-r} + (x_{\mathrm{R}} + \alpha_{\mathrm{R}}) \\ &= ((x_{\mathrm{L}} + \alpha_{\mathrm{L}} + c_{\mathrm{R}}) \mod 2^r)\, 2^{n-r} + ((x_{\mathrm{R}} + \alpha_{\mathrm{R}}) \mod 2^{n-r}) \ , \quad (15)\end{aligned}$$

where c_{R} is the carry generated from the addition $(x_{\mathrm{R}} + \alpha_{\mathrm{R}}) \mod 2^{n-r}$. From (15) follows that $(x + \alpha) \lll r = (x_{\mathrm{R}} + \alpha_{\mathrm{R}})2^r$. Thus for the output difference β we get:

$$\beta = ((x + \alpha) \lll r) - (x \lll r) = (x_{\mathrm{R}} + \alpha_{\mathrm{R}})2^r - x_{\mathrm{R}} 2^r = \alpha_{\mathrm{R}} 2^r = (\alpha \lll r) \ . \quad (16)$$

Note that (16) is independent of the input x and therefore holds with probability 1 over all values of x. From this the expression (10) for the probability $\mathrm{adp}^{\lll r}$ immediately follows. □

D.3 Proof of Theorem 2

Proof. Let x be an n-bit input to `RSH` with shift constant $r \leq n$. Let $x_{\mathrm{L}}, x_{\mathrm{R}} : x = x_{\mathrm{L}} 2^r + x_{\mathrm{R}}$. Then $(x \gg r) = x_{\mathrm{L}}$. Similarly, for the input `ADD` difference α let $\alpha_{\mathrm{L}}, \alpha_{\mathrm{R}} : \alpha = \alpha_{\mathrm{L}} 2^r + \alpha_{\mathrm{R}}$ and thus $(\alpha \gg r) = \alpha_{\mathrm{L}}$. Denote by c_{R} the carry generated from the addition $(a_{\mathrm{R}} + \alpha_{\mathrm{R}}) \mod 2^r$:

$$c_{\mathrm{R}} = \begin{cases} 0 \ , & \text{if } (x_{\mathrm{R}} + \alpha_{\mathrm{R}}) < 2^r \ , \\ 1 \ , & \text{otherwise} \ . \end{cases} \quad (17)$$

The sum $(x + \alpha)$ can then be represented as:

$$\begin{aligned}(x+\alpha) &= (x_{\mathrm{L}} + \alpha_{\mathrm{L}})2^r + (x_{\mathrm{R}} + \alpha_{\mathrm{R}}) \\ &= ((x_{\mathrm{L}} + \alpha_{\mathrm{L}} + c_{\mathrm{R}}) \mod 2^{n-r})\, 2^r + ((x_{\mathrm{R}} + \alpha_{\mathrm{R}}) \mod 2^r) \ . \quad (18)\end{aligned}$$

Therefore $(x + \alpha) \gg r = (x_{\mathrm{L}} + \alpha_{\mathrm{L}} + c_{\mathrm{R}}) \mod 2^{n-r}$ and for the output difference β we derive:

$$\begin{aligned}\beta &= ((x + \alpha) \gg r) - (x \gg r) = ((x_{\mathrm{L}} + \alpha_{\mathrm{L}} + c_{\mathrm{R}}) \mod 2^{n-r}) - x_{\mathrm{L}} \\ &= \alpha_{\mathrm{L}} - c_{\mathrm{L}} 2^{n-r} + c_{\mathrm{R}} \ , \quad (19)\end{aligned}$$

where

$$c_{\mathrm{L}} = \begin{cases} 0 \ , & \text{if } (x_{\mathrm{L}} + \alpha_{\mathrm{L}} + c_{\mathrm{R}}) < 2^{n-r} \ , \\ 1 \ , & \text{otherwise} \ . \end{cases} \quad (20)$$

The term $-c_{\mathrm{L}} 2^{n-r}$ in (19) is introduced in order to cancel the carry 2^{n-r} that is generated in the cases in which the sum $(x_{\mathrm{L}} + \alpha_{\mathrm{L}} + c_{\mathrm{R}})$ is bigger than $(2^{n-r} - 1)$. In such a case $c_{\mathrm{L}} = 1$ and $-c_{\mathrm{L}} 2^{n-r} + (x_{\mathrm{L}} + \alpha_{\mathrm{L}} + c_{\mathrm{R}}) = -2^{n-r} + 2^{n-r} + (x_{\mathrm{L}} + \alpha_{\mathrm{L}} + c_{\mathrm{R}}) \mod 2^{n-r} = (x_{\mathrm{L}} + \alpha_{\mathrm{L}} + c_{\mathrm{R}}) \mod 2^{n-r}$.

In the expression for β (19), for each distinct value of the tuple $(c_{\mathrm{L}}, c_{\mathrm{R}})$ we get one of the four possibilities for β:

$$\beta = \begin{cases} (\alpha \ggg r) \ , & c_{\mathrm{L}} = 0, c_{\mathrm{R}} = 0 \ , \\ (\alpha \ggg r) - 2^{n-r} \ , & c_{\mathrm{L}} = 1, c_{\mathrm{R}} = 0 \ , \\ (\alpha \ggg r) + 1 \ , & c_{\mathrm{L}} = 0, c_{\mathrm{R}} = 1 \ , \\ (\alpha \ggg r) - 2^{n-r} + 1 \ , & c_{\mathrm{L}} = 1, c_{\mathrm{R}} = 1 \ . \end{cases} \quad . \tag{21}$$

In order to compute the corresponding probabilities, we have to count the number of inputs x, that result in a given value for $(c_{\mathrm{L}}, c_{\mathrm{R}})$. Note that c_{L} and c_{R} depend on x and α, of which α is fixed and x can take on all values from 0 to $2^n - 1$. From (17) it is easy to compute that $c_{\mathrm{R}} = 0$ for exactly $(2^r - \alpha_{\mathrm{R}})$ values of x_{R} and therefore $c_{\mathrm{R}} = 1$ for the remaining $2^r - (2^r - \alpha_{\mathrm{R}}) = \alpha_{\mathrm{R}}$ values. Note that x_{R} is an r-bit word. Similarly, if $c_{\mathrm{R}} = 0$ then $c_{\mathrm{L}} = 0$ for $(2^{n-r} - \alpha_{\mathrm{L}})$ values of x_{L} and $c_{\mathrm{L}} = 1$ for the remaining α_{L} values. If $c_{\mathrm{R}} = 1$ then $c_{\mathrm{L}} = 0$ for $(2^{n-r} - \alpha_{\mathrm{L}} - 1)$ values and $c_{\mathrm{L}} = 1$ for the remaining $\alpha_{\mathrm{L}} + 1$ values. Therefore $(c_{\mathrm{L}}, c_{\mathrm{R}}) = (0, 0)$ for $(2^{n-r} - \alpha_{\mathrm{L}})(2^r - \alpha_{\mathrm{R}})$ values of x. Since the total number of values is 2^n we obtain the probability:

$$\mathrm{adp}^{\ggg r}(\alpha \to \beta = (\alpha \ggg r)) = 2^{-n}(2^{n-r} - \alpha_{\mathrm{L}})(2^r - \alpha_{\mathrm{R}}) \ . \tag{22}$$

The expressions for the remaining three probabilities are derived analogously.

□