



ICT-2007-216676

ECRYPT II

European Network of Excellence in Cryptology II

Network of Excellence

Information and Communication Technologies

**D.SYM.6**

**New developments in symmetric key cryptanalysis**

Due date of deliverable: 30. June 2010

Actual submission date: 30. June 2010

Start date of project: 1 August 2008

Duration: 4 years

Lead contractor: Katholieke Universiteit Leuven (KUL)

Revision 1.0

Project co-funded by the European Commission within the 7th Framework Programme		
Dissemination Level		
<b>PU</b>	Public	X
<b>PP</b>	Restricted to other programme participants (including the Commission services)	
<b>RE</b>	Restricted to a group specified by the consortium (including the Commission services)	
<b>CO</b>	Confidential, only for members of the consortium (including the Commission services)	



# New developments in symmetric key cryptanalysis

## Editor

Svetla Nikova (K.U.Leuven)

## Contributors

Jonathan Etrog (Orange Labs)  
Dmitry Khovratovich (University of Luxemburg)  
Willi Meier (FHNW),  
Nicky Mouha (K.U. Leuven),  
Jorge Nakahara Jr. (EPFL),  
Andrea Röck (Aalto),  
Vincent Rijmen (K.U. Leuven),  
Christian Rechberger (K.U. Leuven),  
Martin Schläffer (TU Graz),  
Vesselin Velichkov, (K.U. Leuven)

30. June 2010

Revision 1.0

The work described in this report has in part been supported by the Commission of the European Communities through the ICT program under contract ICT-2007-216676. The information in this document is provided as is, and no warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.



# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Cube Testers and Key Recovery Attacks</b>	<b>5</b>
2.1	Cube Attacks . . . . .	5
2.2	Cube Testers . . . . .	7
2.2.1	Definitions . . . . .	8
2.2.2	Examples . . . . .	9
2.2.3	Building on Property Testers . . . . .	10
2.2.4	Examples of Testable Properties . . . . .	10
2.3	Cube Testers on Trivium . . . . .	12
2.4	Cube attacks and cube testers on other cryptographic schemes . . . . .	13
<b>3</b>	<b>Distinguishing and Initialization-Phase Attacks</b>	<b>15</b>
<b>4</b>	<b>Trade-offs in generic attacks on stream ciphers</b>	<b>19</b>
4.1	Exhaustive attacks . . . . .	19
4.2	Time-Memory-Data trade-offs . . . . .	20
4.3	Time-Memory-Key trade-offs . . . . .	20
<b>5</b>	<b>Implications of Related-Key Attacks</b>	<b>23</b>
5.1	Related-key attacks in theory . . . . .	23
5.2	Related-key attacks in practice . . . . .	23
5.3	Related-key attacks and security claims . . . . .	24
5.4	Outline of recent attacks . . . . .	24
5.4.1	Attacks on AES . . . . .	24
5.4.2	Attacks on other ciphers . . . . .	27
<b>6</b>	<b>Linear Cryptanalysis</b>	<b>29</b>
6.1	Piling-up lemma . . . . .	30
6.2	Matsui Algorithm 1 . . . . .	30
6.3	Matsui Algorithm 2 . . . . .	31
6.3.1	Probability of success of Algorithm 2 . . . . .	32
6.4	Linear hull . . . . .	32
6.5	Deriving methods . . . . .	33

<b>7</b>	<b>Multiple-Linear and Multidimensional Linear Cryptanalysis</b>	<b>35</b>
7.1	Multiple Linear Cryptanalysis (Biryukov Method) . . . . .	35
7.2	Multidimensional Linear Cryptanalysis . . . . .	36
<b>8</b>	<b>Cryptanalysis of ARX Structures</b>	<b>41</b>
8.1	Rotational Cryptanalysis . . . . .	42
8.2	Toolkit for Cryptanalysis of ARX . . . . .	43
<b>9</b>	<b>Analysis of AES-based Hash Functions</b>	<b>45</b>
<b>10</b>	<b>Meet-in-the-middle preimage attacks on hash functions</b>	<b>47</b>
10.1	The basic approach . . . . .	47
10.2	Refinements and improvements . . . . .	48
10.3	Discussion of results . . . . .	49

## **Executive Summary**

Symmetric cryptography addresses the problem of protecting the secret information using a shared secret key. The message is transformed in such a way that it cannot be recovered without this key. Algorithms which are used for symmetric encryption are known as symmetric ciphers: block ciphers and stream ciphers. The security of symmetric encryption algorithms can in general not be proved (with the exception of the one-time pad). Instead, the trust in a symmetric cipher is based on the fact that no weaknesses have been found after a long and thorough evaluation phase. The field which focuses on methods to defeat the secrecy of the information, i.e. which aims at breaking the ciphers is called **cryptanalysis**.

In this deliverable we provide a state of the art survey of recent developments in symmetric cryptanalysis. We summarize the best to our knowledge techniques developed for cryptanalysis of block ciphers, stream ciphers and hash functions. This includes general attacks that apply in the same way to all symmetric cryptographic algorithms of given dimensions, as well as specific shortcut attacks, that exploit weaknesses in the designs. We also discuss the practical aspects of the presented attacks and how they affect the security of the ciphers.



# Chapter 1

## Introduction

In this deliverable we have collected the most important recent developments in symmetric key cryptanalysis.

We start with **cube attacks** - a new type of algebraic attacks for key recovery, which has been introduced in 2008. Cube attacks are applicable to cryptographic functions that have a public parameter and that are inherently of a low algebraic degree. A variant of cube attacks are **cube testers**, based on efficient property-testing algorithms. Attacks related to cube attacks and cube testers have been applied to the stream ciphers Trivium and Grain-128, and to the SHA-3 candidate MD6. In chapter 2 we summarize the main results.

The second type of attacks that we discuss is the **distinguishing attack**, where the adversary tries to determine if a given bit stream comes from a known cipher or from a purely random source. Although often weaker than a key-recovery attack, the existence of a distinguisher of high probability may lead to information about unknown plaintext or the internal state can leak through the ciphertext. Distinguishing attacks become quite conventional in the cryptographic literature, and many examples can be found, some of which are listed in Chapter 3.

**Trade-offs** in generic attacks on stream ciphers are discussed in Chapter 4. We briefly describe three types of generic attacks: exhaustive attacks, time-memory-data trade-offs, time-memory-key trade-offs.

A series of **related-key** attacks on AES drawn much attention in 2009. For those general related-key attacks there is no protocol or application known where the attacks could be considered to be practical. In Chapter 5 we summarize the results and discuss the main ideas. We also briefly mention related-key attacks applied to other ciphers.

**Linear cryptanalysis** is one of the most powerful statistical cryptanalysis tools using known plaintext/ciphertext pairs and it is based on probabilistic linear approximations. While the idea to use probabilistic approximations has already appeared in 1992 applied to FEAL, the theory of linear cryptanalysis was described and experimented on DES by Matsui in 1994. More recent results will be discussed in Chapter 6.

Basic linear cryptanalysis uses one linear approximation, but can we improve the attacks by using several approximations? Answer to this question is given in Chapter 7, where new developments in **multiple-linear** and **multidimensional linear** cryptanalysis are presented.

Increasingly, cryptographic primitives use operations such as addition modulo  $2^n$ , rotation and exclusive ORs (ARX), as well as bitwise Boolean functions. In NIST's SHA-3 hash function competition, this applies to 6 out of the 14 second-round candidates. An overview of

recent developments in the field of **ARX-based cryptography** are presented in Chapter 8.

The submission of many **AES-based hash** function to the NIST SHA-3 competition has initiated new developments in the analysis of AES as a building block in hash functions. Two different directions of analyzing AES based hash functions have been recently proposed - an attack applied to AES in hashing mode, which uses local collisions to construct high-probability differential paths and the rebound attack, which uses the available degrees of freedom to find pairs for the (multiple) expansive parts of a path more efficiently. Both approaches are discussed in Chapter 9.

**Meet-in-the-middle** attacks on the preimage resistance of hash functions recently received renewed attention. This is both in the context of MD2 by Knudsen et al., attacks on round-1 SHA-3 candidates by Khovratovic et al., as well as a series of results on members of the MD4 family of hash functions by Aoki, Sasaki and others and Tiger. In Chapter 10, we describe them in a way to fit into the meet-in-the-middle (MITM) framework of Aoki and Sasaki.

## Chapter 2

# Cube Testers and Key Recovery Attacks

At CRYPTO 2008 a type of algebraic attacks for key recovery, called cube attacks, has been introduced. Cube attacks are applicable to cryptographic functions that have a public parameter and that are inherently of a low algebraic degree. A variant of cube attacks are cube testers, based on efficient property-testing algorithms. Unlike cube attacks, cube testers detect nonrandom behavior rather than performing key extraction, but they can also attack cryptographic schemes described by nonrandom polynomials of relatively high degree. Attacks related to cube attacks and cube testers have been applied to the stream ciphers Trivium and Grain-128, and to the SHA-3 candidate MD6.

### 2.1 Cube Attacks

Cube attacks [Sha,DS09] are a particular type of algebraic cryptanalysis that exploit implicit low-degree equations in cryptographic algorithms. Cube attacks only require black box access to the target primitive, and were successfully applied to reduced versions of the stream cipher Trivium [CP08] in [DS09]. Roughly speaking, a cryptographic function is vulnerable to cube attacks if its implicit algebraic normal form over  $\mathbb{GF}(2)$  has degree at most  $d$ , provided that  $2^d$  computations of the function is feasible. Cube attacks recover a secret key through queries to a *black box polynomial with tweakable public variables* (e.g. chosen plaintext or IV bits), followed by solving a linear system of equations in the secret key variables. A one time preprocessing phase is required to determine which queries should be made to the black box during the on-line phase of the attack. Low-degree implicit equations were previously exploited in [Fil02, Saa06, O’N07, EJT07] to construct distinguishers, and in [Vie07, FKM08, KM08] for key recovery. Cube attacks are related to saturation attacks [Luc01] and to high order differential cryptanalysis [Knu94].

**The idea** Let  $\mathcal{F}_n$  be the set of all functions mapping  $\{0, 1\}^n$  to  $\{0, 1\}$ ,  $n > 0$ , and let  $f \in \mathcal{F}_n$ . The *algebraic normal form* (ANF) of  $f$  is the polynomial  $p$  over  $\mathbb{GF}(2)$  in variables  $x_1, \dots, x_n$  such that evaluating  $p$  on  $x \in \{0, 1\}^n$  is equivalent to computing  $f(x)$ , and such that it is of

the form<sup>1</sup>

$$\sum_{i=0}^{2^n-1} a_i \cdot x_1^{i_1} x_2^{i_2} \cdots x_{n-1}^{i_{n-1}} x_n^{i_n}$$

for some  $(a_0, \dots, a_{2^n-1}) \in \{0, 1\}^{2^n}$ , and where  $i_j$  denotes the  $j$ -th digit of the binary encoding of  $i$  (and so the sum spans all monomials in  $x_1, \dots, x_n$ ). A key observation regarding cube attacks is that for any function  $f : \{0, 1\}^n \mapsto \{0, 1\}$ , the sum (XOR) of all entries in the truth table

$$\sum_{x \in \{0,1\}^n} f(x)$$

equals the coefficient of the highest degree monomial  $x_1 \cdots x_n$  in the algebraic normal form (ANF) of  $f$ . For example, let  $n = 4$  and  $f$  be defined as

$$f(x_1, x_2, x_3, x_4) = x_1 + x_1 x_2 x_3 + x_1 x_2 x_4 + x_3 .$$

Then summing  $f(x_1, x_2, x_3, x_4)$  over all 16 distinct inputs makes all monomials vanish and yields zero, i.e. the coefficient of the monomial  $x_1 x_2 x_3 x_4$ . Instead, cube attacks sum over a *subset* of the inputs; for example summing over the four possible values of  $(x_1, x_2)$  gives

$$f(0, 0, x_3, x_4) + f(0, 1, x_3, x_4) + f(1, 0, x_3, x_4) + f(1, 1, x_3, x_4) = x_3 + x_4 ,$$

where  $(x_3 + x_4)$  is the polynomial that multiplies  $x_1 x_2$  in  $f$ :

$$f(x_1, x_2, x_3, x_4) = x_1 + x_1 x_2 (x_3 + x_4) + x_3 .$$

Generalizing, given an index set  $I \subsetneq \{1, \dots, n\}$ , any function in  $\mathcal{F}_n$  can be represented algebraically under the form

$$f(x_1, \dots, x_n) = t_I \cdot p(\cdots) + q(x_1, \dots, x_n)$$

where  $t_I$  is the monomial containing all the  $x_i$ 's with  $i \in I$ ,  $p$  is a polynomial that has no variable in common with  $t_I$ , and such that no monomial in the polynomial  $q$  contains  $t_I$  (that is, we factored  $f$  by the monomial  $t_I$ ). Summing  $f$  over the *cube*  $t_I$  for other variables fixed, one gets

$$\sum_I t_I \cdot p(\cdots) + q(x_1, \dots, x_n) = \sum_I t_I \cdot p(\cdots) = p(\cdots),$$

that is, the evaluation of  $p$  for the chosen fixed variables. Following the terminology of [DS09],  $p$  is called the *superpoly* of  $I$  in  $f$ . A *cube*  $t_I$  is called a *maxterm* if and only if its superpoly  $p$  has degree 1 (i.e., is linear but not a constant). The polynomial  $f$  is called the *master polynomial*.

Given access to a cryptographic function with public and secret variables, the attacker has to recover the secret key variables. Key recovery is achieved in two steps, a preprocessing and an online phase, which are described below.

<sup>1</sup>The ANF of any  $f \in \mathcal{F}_n$  has degree at most  $n$ , since  $x_i^d = x_i$ , for  $x_i \in \text{GF}(2)$ ,  $d > 0$ .

**Preprocessing** One first finds sufficiently many maxterms  $t_I$  of the master polynomial. For each maxterm, one computes the coefficients of the secret variables in the symbolic representation of the linear superpoly  $p$ . That is, one *reconstructs* the ANF of the superpoly of each  $t_I$ . Reconstruction is achieved via probabilistic linearity tests [BLR90], to check that a superpoly is linear, and to identify which variables it contains. The maxterms and superpolys are not key-dependent, thus they need to be computed only once per master polynomial.

The main challenge of the cube attack is to find maxterms. We propose the following simple preprocessing heuristic: one randomly chooses a subset  $I$  of  $k$  public variables. Thereafter one uses a linearity test to check whether  $p$  is linear. If the subset  $I$  is too small, the corresponding superpoly  $p$  is likely to be a nonlinear function in the secret variables, and in this case the attacker adds a public variable to  $I$  and repeats the process. If  $I$  is too large, the sum will be a constant function, and in this case he drops one of the public variables from  $I$  and repeats the process. The correct choice of  $I$  is the borderline between these cases, and if it does not exist the attacker retries with a different initial  $I$ .

**Online Phase** Once sufficiently many maxterms and the ANF of their superpolys are found, preprocessing is finished and one performs the online phase. Now the secret variables are fixed: one evaluates the superpoly's  $p$  by summing  $f(x)$  over all the values of the corresponding maxterm, and gets as a result a linear combination of the key bits (because the superpolys are linear). The public variables that are not in the maxterm should be set to a fixed value, and to the same value as set in the preprocessing phase.

Assuming that the degree of the master polynomial is  $d$ , each sum requires at most  $2^{d-1}$  evaluations of the derived polynomials (which the attacker obtains via a chosen plaintext attack). Once enough linear superpolys are found, the key can be recovered by simple linear algebra techniques.

**Trivium** The stream cipher Trivium was designed by De Cannière and Preneel [CP08] and submitted as a candidate to the eSTREAM project in 2005. Trivium was eventually chosen as one of the four hardware ciphers in the eSTREAM portofolio<sup>2</sup>. Reduced variants of Trivium underwent several attacks [Rad06, MB07, MCP07, TK07, Vie07, EJT07, FKM08, Pas08], including cube attacks [DS09].

Trivium takes as input a 80-bit key and a 80-bit IV, and produces a keystream after 1152 rounds of initialization. Each round corresponds to clocking three feedback shift registers, each one having a quadratic feedback polynomial. The best result on Trivium is a cube attack [DS09] on a reduced version with 767 initialization rounds instead of 1152.

## 2.2 Cube Testers

Unlike cube attacks, cube testers detect a nonrandom behaviour without recovering a secret key. Cube testers and its predecessors have been introduced in [ADMS09, Fil02, Saa06, O'N07, EJT07]

---

<sup>2</sup>See <http://www.ecrypt.eu.org/stream/>

### 2.2.1 Definitions

Recall that  $\mathcal{F}_n$  denotes the set of all functions mapping  $\{0, 1\}^n$  to  $\{0, 1\}$ ,  $n > 0$ . For a given  $n$ , a *random function* is a random element of  $\mathcal{F}_n$  (we have  $|\mathcal{F}_n| = 2^{2^n}$ ). In the ANF of a random function, each monomial (and in particular, the highest degree monomial  $x_1 \cdots x_n$ ) appears with probability  $1/2$ , hence a random function has maximal degree of  $n$  with probability  $1/2$ . Similarly, it has degree  $(n - 2)$  or less with probability  $1/2^{n+1}$ . Note that the explicit description of a random function can be directly expressed as a circuit with, in average,  $2^{n-1}$  gates (AND and XOR), or as a string of  $2^n$  bits where each bit is the coefficient of a monomial (encoding the truth table also requires  $2^n$  bits, but hides the algebraic structure).

Informally, a *distinguisher* for a family  $\mathcal{F} \subsetneq \mathcal{F}_n$  is a procedure that, given a function  $f$  randomly sampled from  $\mathcal{F}^* \in \{\mathcal{F}, \mathcal{F}_n\}$ , efficiently determines which one of these two families was chosen as  $\mathcal{F}^*$ . A family  $\mathcal{F}$  is *pseudorandom* if and only if there exists no efficient distinguisher for it. In practice, e.g. for hash functions or ciphers, a family of functions is defined by a  $k$ -bit parameter of the function, randomly chosen and unknown to the adversary, and the function is considered broken (or, at least, “nonrandom”) if there exists a distinguisher making significantly less than  $2^k$  queries to the function. Note that a distinguisher that runs in exponential time in the key may be considered as “efficient” in practice, e.g.  $2^{k-10}$ .

We would like to stress the terminology difference between a *distinguisher* and the more general detection of *pseudorandomness*, when speaking about cryptographic algorithms; the former denotes a distinguisher (as defined above) where the parameter of the family of functions is the cipher’s *key*, and thus can’t be modified by the adversary through its queries; the latter considers part of the key as a public input, and assumes as secret an arbitrary subset of the input (including the input bits that are normally public, like IV bits). The detection of nonrandomness thus does not necessarily correspond to a realistic scenario. Note that related-key attacks are captured by neither one of those scenarios.

To distinguish  $\mathcal{F} \subsetneq \mathcal{F}_n$  from  $\mathcal{F}_n$ , cube testers partition the set of public variables  $\{x_1, \dots, x_n\}$  into two complementary subsets:

- *cube variables* (CV)
- *superpoly variables* (SV)

We illustrate these notions with the example from §2.1: recall that, given

$$f(x_1, x_2, x_3, x_4) = x_1 + x_1x_2x_3 + x_1x_2x_4 + x_3 ,$$

we considered the *cube*  $x_1x_2$  and called  $(x_3 + x_4)$  its *superpoly*, because

$$f(x_1, x_2, x_3, x_4) = x_1 + x_1x_2(x_3 + x_4) + x_3 .$$

Here the cube variables (CV) are  $x_1$  and  $x_2$ , and the superpoly variables (SV) are  $x_3$  and  $x_4$ . Therefore, by setting a value to  $x_3$  and  $x_4$ , e.g.  $x_3 = 0$ ,  $x_4 = 1$ , one can compute  $(x_3 + x_4) = 1$  by summing  $f(x_1, x_2, x_3, x_4)$  for all possible choices of  $(x_1, x_2)$ . Note that it is not required for a SV to actually appear in the superpoly of the maxterm. For example, if  $f(x_1, x_2, x_3, x_4) = x_1 + x_1x_2x_3$ , then the superpoly of  $x_1x_2$  is  $x_3$ , but the SV’s are both  $x_3$  and  $x_4$ .

**Remark.** When  $f$  is, for example, a hash function, not all inputs should be considered as variables, and not all Boolean components should be considered as outputs, for the sake of efficiency. For example if  $f$  maps 1024 bits to 256 bits, one may choose 20 CV and 10 SV and set a fixed value to the other inputs. These fixed inputs determine the coefficient of each monomial in the ANF with CV and SV as variables. This is similar to the preprocessing phase of key-recovery cube attacks, where one has access to all the input variables. Finally, for the sake of efficiency, one may only evaluate the superpolys for 32 of the 256 Boolean components of the output.

## 2.2.2 Examples

Cube testers distinguish a family of functions from random functions by testing a property of the superpoly for a specific choice of CV and SV. This section introduces this idea with simple examples. Consider

$$f(x_1, x_2, x_3, x_4) = x_1 + x_1x_2x_3 + x_1x_2x_4 + x_3$$

and suppose we choose CV  $x_3$  and  $x_4$  and SV  $x_1$  and  $x_2$ , and evaluate the superpoly of  $x_3x_4$ :

$$f(x_1, x_2, 0, 0) + f(x_1, x_2, 0, 1) + f(x_1, x_2, 1, 0) + f(x_1, x_2, 1, 1) = 0 ,$$

This yields zero for any  $(x_1, x_2) \in \{0, 1\}^2$ , i.e. the superpoly of  $x_3x_4$  is zero, i.e. none of the monomials  $x_3x_4$ ,  $x_1x_3x_4$ ,  $x_2x_3x_4$ , or  $x_1x_2x_3x_4$  appears in  $f$ . In comparison, in a random function the superpoly of  $x_3x_4$  is null with probability only 1/16, which suggests that  $f$  was not chosen at random (indeed, we chose it particularly sparse, for clarity). Generalizing the idea, one can deterministically test whether the superpoly of a given maxterm is constant, and return “random function” if and only if the superpoly is not constant. This is similar to the test used in [EJT07].

Let  $f \in \mathcal{F}_n$ ,  $n > 4$ . We present a probabilistic test that detects the presence of monomials of the form  $x_1x_2x_3x_i \dots x_j$  (e.g.  $x_1x_2x_3$ ,  $x_1x_2x_3x_n$ , etc.):

1. choose a random value of  $(x_4, \dots, x_n) \in \{0, 1\}^{n-4}$
2. sum  $f(x_1, \dots, x_n)$  over all values of  $(x_1, x_2, x_3)$ , to get

$$\sum_{(x_1, x_2, x_3) \in \{0, 1\}^3} f(x_1, \dots, x_n) = p(x_4, \dots, x_n)$$

where  $p$  is a polynomial such that

$$f(x_1, \dots, x_n) = x_1x_2x_3 \cdot p(x_4, \dots, x_n) + q(x_1, \dots, x_n)$$

where the polynomial  $q$  contains no monomial with  $x_1x_2x_3$  as a factor in its ANF

3. repeat the two previous steps  $N$  times, recording the values of  $p(x_4, \dots, x_n)$

If  $f$  were a random function, it would contain at least one monomial of the form  $x_1x_2x_3x_i \dots x_j$  with high probability; hence, for a large enough number of repetitions  $N$ , one would record at least one nonzero  $p(x_4, \dots, x_n)$  with high probability. However, if no monomial of the form  $x_1x_2x_3x_i \dots x_j$  appears in the ANF,  $p(x_4, \dots, x_n)$  always evaluates to zero.

### 2.2.3 Building on Property Testers

Cube testers combine an efficient property tester on the superpoly, which is viewed either as a polynomial or as a mapping, with a statistical decision rule. This section gives a general informal definition of cube testers, starting with basic definitions. A *family tester* for a family of functions  $\mathcal{F}$  takes as input a function  $f$  of same domain  $\mathcal{D}$  and tests if  $f$  is close to  $\mathcal{F}$ , with respect to a bound  $\epsilon$  on the distance

$$\delta(f, \mathcal{F}) = \min_{g \in \mathcal{F}} \frac{|\{x \in \mathcal{D}, f(x) \neq g(x)\}|}{|\mathcal{D}|}.$$

The tester accepts if  $\delta(f, \mathcal{F}) = 0$ , rejects with high probability if  $f$  and  $\mathcal{F}$  are not  $\epsilon$ -close, and behaves arbitrarily otherwise. Such a test captures the notion of property-testing, when a property is defined by belonging to a family of functions  $\mathcal{P}$ ; a *property tester* is thus a family tester for a property  $\mathcal{P}$ .

Suppose one wishes to distinguish a family  $\mathcal{F} \subsetneq \mathcal{F}_n$  from  $\mathcal{F}_n$ , i.e., given a random  $f \in \mathcal{F}^*$ , to determine whether  $\mathcal{F}^*$  is  $\mathcal{F}$  or  $\mathcal{F}_n$  (for example, in Trivium,  $\mathcal{F}$  may be a superpoly with respect to CV and SV in the IV bits, such that each  $f \in \mathcal{F}$  is computed with a distinct key). Then if  $\mathcal{F}$  is efficiently testable (see [RS96, KS08]), then one can use directly a family tester for  $\mathcal{F}$  on  $f$  to distinguish it from a random function.

Cube testers detect nonrandomness by applying property testers to superpolys: informally, as soon as a superpoly has some “unexpected” property (that is, is anomalously structured) it is identified as nonrandom. Given a testable property  $\mathcal{P} \subsetneq \mathcal{F}_n$ , cube testers run a tester for  $\mathcal{P}$  on the superpoly function  $f$ , and use a statistical decision rule to return either “random” or “nonrandom”. The decision rule depends on the probabilities  $|\mathcal{P}|/|\mathcal{F}_n|$  and  $|\mathcal{P} \cap \mathcal{F}|/|\mathcal{F}|$  and on a margin of error chosen by the attacker. Roughly speaking, a family  $\mathcal{F}$  will be distinguishable from  $\mathcal{F}_n$  using the property  $\mathcal{P}$  if

$$\left| \frac{|\mathcal{P}|}{|\mathcal{F}_n|} - \frac{|\mathcal{P} \cap \mathcal{F}|}{|\mathcal{F}|} \right|$$

is non-negligible. That is, the tester will determine whether  $f$  is significantly closer to  $\mathcal{P}$  than a random function. Note that the dichotomy between structure (e.g. testable properties) and randomness has been studied in [Tao06].

### 2.2.4 Examples of Testable Properties

Below, we give examples of efficiently testable properties of the superpoly, which can be used to build cube testers (see [KS08] for a general characterization of efficiently testable properties). We let  $C$  be the size of CV, and  $S$  be the size of SV; the complexity is given as the number of evaluations of the tested function  $f$ . Note that each query of the tester to the superpoly requires  $2^C$  queries to the target cryptographic function. The complexity of any property tester is thus, even in the best case, exponential in the number of CV.

**Imbalance.** A random function is expected to contain as many zeroes as ones in its truth table. Superpolys that have a strongly imbalanced truth table can thus be distinguished from random polynomials, by testing whether it evaluates as often to one as to zero, either deterministically (by evaluating the superpoly for each possible input), or probabilistically (over some random subset of the SV). For example, if CV are  $x_1, \dots, x_C$  and SV are  $x_{C+1}, \dots, x_n$ , the deterministic imbalance test is

1.  $c \leftarrow 0$

2. **for** all values of  $(x_{C+1}, \dots, x_n)$

3.     compute

$$p(x_{C+1}, \dots, x_n) = \sum_{(x_1, \dots, x_C)} f(x_1, \dots, x_n) \in \{0, 1\}$$

4.      $c \leftarrow c + p(x_{C+1}, \dots, x_n)$

5. **return**  $\mathcal{D}(c) \in \{0, 1\}$

where  $\mathcal{D}$  is some decision rule. A probabilistic version of the test makes  $N < 2^S$  iterations, for random distinct values of  $(x_{C+1}, \dots, x_n)$ . Complexity is respectively  $2^n$  and  $N \cdot 2^C$ .

**Constantness.** A particular case of balance test considers the “constantness” property, i.e. whether the superpoly defines a constant function; that is, it detects either that  $f$  has maximal degree strictly less than  $C$  (null superpoly), or that  $f$  has maximal degree exactly  $C$  (superpoly equals the constant 1), or that  $f$  has degree strictly greater than  $C$  (non-constant superpoly). This is equivalent to the maximal degree monomial test used in [EJT07], used to detect nonrandomness in 736-round Trivium.

**Low Degree.** A random superpoly has degree at least  $(S - 1)$  with high probability. Cryptographic functions that rely on a low-degree function, however, are likely to have superpolys of low degree. Because it closely relates to probabilistically checkable proofs and to error-correcting codes, low-degree testing has been well studied; the most relevant results to our concerns are the tests for Boolean functions in [AKK<sup>+</sup>03, Sam07]. The test by Alon et al. [AKK<sup>+</sup>03], for a given degree  $d$ , queries the function at about  $d \cdot 4^d$  points and always accepts if the ANF of the function has degree at most  $k$ , otherwise it rejects with some bounded error probability. Note that, contrary to the method of ANF reconstruction (exponential in  $S$ ), the complexity of this algorithm is *independent of the number of variables*. Hence, cube testers based on this low-degree test have complexity which is independent of the number of SV’s.

**Presence of Linear Variables.** This is a particular case of the low-degree test, for degree  $d = 1$  and a single variable. Indeed, the ANF of a random function contains a given variable in at least one monomial of degree at least two with probability close to 1. One can thus test whether a given superpoly variable appears only linearly in the superpoly, e.g. for  $x_1$  using the following test similar to that introduced in [BLR90]:

1. pick random  $(x_2, \dots, x_S)$

2. **if**  $p(0, x_2, \dots, x_S) = p(1, x_2, \dots, x_S)$

3.     **return** nonlinear

4. repeat steps 1 to 3  $N$  times

5. **return** linear

This test answers correctly with probability about  $1 - 2^{-N}$ , and computes  $N \cdot 2^{C+1}$  times the function  $f$ . If, say, a stream cipher is shown to have an IV bit linear with respect to a set of CV in the IV, independently of the choice of the key, then it directly gives a distinguisher.

**Presence of Neutral Variables.** Dually to the above linearity test, one can test whether a SV is neutral in the superpoly, that is, whether it appears in at least one monomial. For example, the following algorithm tests the neutrality of  $x_1$ , for  $N \leq 2^{S-1}$ :

1. pick random  $(x_2, \dots, x_S)$
2. **if**  $p(0, x_2, \dots, x_S) \neq p(1, x_2, \dots, x_S)$
3.     **return** not neutral
4. repeat steps 1 to 3  $N$  times
5. **return** neutral

This test answers correctly with probability about  $1 - 2^{-N}$  and runs in time  $N \cdot 2^C$ . For example, if  $x_1, x_2, x_3$  are the CV and  $x_4, x_5, x_6$  the SV, then  $x_6$  is neutral with respect to  $x_1 x_2 x_3$  if the superpoly  $p(x_4, x_5, x_6)$  satisfies  $p(x_4, x_5, 0) = p(x_4, x_5, 1)$  for all values of  $(x_4, x_5)$ . A similar test was implicitly used in [FKM08], via the computation of a *neutrality measure*.

**Remarks.** Except low degree and constantness, the above properties do not require the superpoly to have a low degree to be tested. For example if the maxterm  $x_1 x_2$  has the degree-5 superpoly

$$x_3 x_5 x_6 + x_3 x_5 x_6 x_7 x_8 + x_5 x_8 + x_9 ,$$

then one can distinguish this superpoly from a random one either by detecting the linearity of  $x_9$  or the neutrality of  $x_4$ , with a cost independent on the degree. In comparison, the cube tester suggested in [DS09] required the degree to be bounded by  $d$  such that  $2^d$  is feasible.

Note that the cost of detecting the property during the preprocessing is larger than the cost of the on-line phase of the attack, given the knowledge of the property. For example, testing that  $x_1$  is a neutral variable requires about  $N \cdot 2^C$  queries to the function, but once this property is known,  $2^C$  queries are sufficient to distinguish the function from a random one with high probability.

Finally, note that tests based on the nonrandom distribution of the monomials [Fil02, Saa06, O’N07] are not captured by this definition of cube testers, which focus on high-degree terms. Although, in principle, there exist cases where the former tests would succeed while cube testers would fail, in practice a weak distribution of lower-degree monomials rarely comes with a good distribution of high-degree ones, as results in [EJT07] and of ourselves suggest.

## 2.3 Cube Testers on Trivium

Observations in [DS09, Tables 1,2,3] suggest nonrandomness properties detectable in time about  $2^{12}$  after 684 rounds, in time  $2^{24}$  after 747 rounds, and in time  $2^{30}$  after 774 rounds. However, a distinguisher cannot be directly derived because the SV used are in the key, and thus cannot be chosen by the attacker in an attack where the key is fixed. For distinguishing

reduced-round Trivium from random, in [ADMS09] cube testers that test for neutrality of a set of SV's in the IV lead to a distinguisher on 790 rounds with  $2^{30}$  complexity. Moreover, nonrandomness over 885 rounds was detected in  $2^{27}$  complexity.

## 2.4 Cube attacks and cube testers on other cryptographic schemes

Cube attacks and cube testers have been applied in [ADMS09] to reduced-round versions of the SHA-3 candidate hash function MD6, [RAB<sup>+</sup>]. In a similar spirit as reported for Trivium, extensive hardware assisted cube testers on the eSTREAM candidate stream cipher Grain-128 <sup>3</sup> have been carried out in [ADH<sup>+</sup>]. A statistical framework related to cube attacks, based on probabilistic neutral bits, has been independently developed and applied in [FKM08] for reduced complexity key recovery in Grain-128 for up to 180 of its 256 rounds of initialization. A similar concept has successfully been applied in [KM08] to a T-function based self-synchronizing stream cipher proposed by Klimov and Shamir, [KS05]. This is interesting as the cryptanalysed cipher uses word-wise operations in the T-function approach, rather than low-degree boolean functions, as is the case for most other primitives cryptanalysed thus far by cube attacks. Finally, the zero-sum distinguishers as considered in [AM] are influenced by high-order differential cryptanalysis and by cube distinguishers.

**Conclusions** Cube attacks and cube testers can serve as a benchmark for evaluating the algebraic strength of primitives based on low degree components, and a reference for choosing the number of rounds.

The results on cube attacks and cube tests leave several issues open:

1. So far cube attacks have resulted from empirical observations, so that one could only assess the existence of feasible attacks. However, if one could upper-bound the degree of some Boolean component (e.g. of Trivium) after a higher number of rounds, then one could predict the existence of observable nonrandomness (and one may build distinguishers based on low-degree tests [AKK<sup>+</sup>03]). The problem is closely related to that of bounding the degree of a nonlinear recursive Boolean sequence which, to the best of our knowledge, has remained unsolved.
2. Low-degree tests may be used for purposes other than detecting nonrandomness. For example, key-recovery cube attacks may be optimized by exploiting low-degree tests, to discover low-degree superpolys, and then reconstruct them. Also, low-degree tests for general fields [KR04] may be applicable to hash functions based on multivariate systems [BRP07], which remain unbroken over fields larger than GF(2) [AM07].

**Acknowledgment** Thanks go to Jean-Philippe Aumasson for helpful inputs and comments.

---

<sup>3</sup>See <http://www.ecrypt.eu.org/stream/>



## Chapter 3

# Distinguishing and Initialization-Phase Attacks

In a distinguishing attack, the adversary tries to determine if a given bit stream comes from a known cipher or from a purely random source (a bit stream chosen according to the uniform distribution). In this case, the distinguisher is a tool that takes a bit stream as input and outputs either **1** (cipher) or **0** (random). Additionally, such attacks may allow an adversary to make predictions about upcoming portions of the keystream sequence. This is a generic kind of attack on stream ciphers. Although often weaker than a key-recovery attack, the existence of a distinguisher of high probability may lead to information leakage, that is, information about unknown plaintext or the internal state can leak through the ciphertext. Distinguishing attacks become quite conventional in the cryptographic literature, and many examples can be found, some of which are listed below.

Initialization or resynchronization mechanisms are sensitive components of synchronous stream ciphers, and have been targeted for a long time in attacks, such as [DGV94, Mul04a]. Recovery from loss of synchronization (or resynchronization) can be performed by the receiver without intervention by the sender if the plaintext has enough redundancy, or can be mitigated by inserting synchronization patterns in the ciphertext at regular intervals. The aim is to guarantee that sender and receiver keep the same internal state during transmission, that is, they operate in sync.

Retransmitting state information in clear will compromise security and could be too costly or ineffective due to the same problems that caused loss of synchronization in the first place. More effective solutions involve defining a new internal state based on information already known by sender and receiver: a secret key and fresh IVs. This setting of new IVs (resynchronization) becomes a source of potential attacks, when the adversary has control over the selection of IVs.

In [RH02], Rose and Hawkes argue that the existence of distinguishing attacks is not a threat to stream ciphers in practice. In particular, the data complexity of these attacks is unrelated to the key length and is often much beyond the amount that can be obtained in real applications.

In [EHJ], Englund *et al.* describe generic distinguishing attacks on block ciphers in stream modes such as OFB and CTR, with complexity  $2^{n/2}$  encryptions, with  $n$  the bit size of a text block. This observation implied an upper bound of  $2^{k/2}$  on the number of keystream bits generated (or plaintext bits encrypted) by these modes before the key (of size  $k$  bits) is

changed [ZB05]. Englund *et al.* also describe a new resynchronization attack on LEX stream cipher requiring  $2^{65.66}$  keystream bits from a single IV and the first 132 bits from  $2^{65.66}$  messages initialized with different IVs. In [DK08], Dunkelman and Keller present a key-recovery attack on the updated (tweaked version of) LEX requiring  $2^{36.3}$  key stream bytes (possibly under different IVs) and time complexity  $2^{112}$  operations.

In [FM00], Fluhrer and McGrew describe an attack that distinguish RC4 from a random stream using  $2^{30.6}$  output key bytes. This attack works even if a few key bytes are discarded (blank rounds), which is a typical countermeasure against attacks that exploit biases in the distribution of the first key bytes output by RC4.

In [Max05], Maximov propose attacks that distinguish the VMPC stream cipher [Zol04] from random using  $2^{54}$  bytes of the output key stream, and the RC4A stream cipher [PP04], an improved variant of RC4, with  $2^{58}$  key stream bytes. This attack on VMPC violated a design criteria of the cipher that explicitly claimed resistance against distinguishing attacks. An improved attack on VMPC was reported in [TSK<sup>+</sup>05] requiring  $2^{38}$  key stream bytes, and on RC4A using  $2^{23}$  key bytes.

In [PPS06], Paul *et al.* presented distinguishing attacks on the Py stream cipher exploiting a statistical bias in the distribution of its output words in the first and third rounds. More specifically, the weakness originates from the non-uniform distribution of carry bits in the modular addition in Py. This attack requires  $2^{84.7}$  random key/IV's, plus the first 24 bytes of the keystream. The time complexity is  $t_{\text{ini}} \cdot 2^{84.7}$ , where  $t_{\text{ini}}$  is the running time of a single key/IV setup in Py. In [IOKM06], Isobe *et al.* report on key-recovery attacks on Py and Pypy, under a chosen-IV setting. Their attack recover the 128-bit key with a time complexity of  $2^{48}$ , under approximately  $2^{24}$  chosen IVs.

In [WP06a], Wu and Preneel described flaws in the IV setup of Py and Pypy stream ciphers. For IVs with special differences, two keystreams can be identical for every  $2^{16}$  IVs. In [WP06b], Wu and Preneel presented key-recovery attacks on Py and Pypy based on weaknesses in the IV setup algorithms of these two ciphers. More specifically, the problem is that two keystreams generated from the chosen IVs can be identical with high probability. In their attack, if the IV size is more than 10 bytes,  $\text{IVsizeb} - 9$  bytes of the key can be recovered with  $(\text{IVsizeb} - 4) \cdot 2^{19}$  chosen IVs, where  $\text{IVsizeb}$  is the IV size in bytes. For key and IV of 128 bits each, the effective key length is reduced to 72 bits with approximately  $2^{24}$  chosen IVs.

Another chosen-IV attack, this time on the VEST family of stream ciphers, was presented by Joux and Reinhard [JR07]. Their attack exploit collisions in the counter (diffusor) used during the IV setup phase. As a consequence, they can recover 53 bits of the internal cipher state reducing the effort of exhaustive search by the same number of bits.

In [NP07], Naya-Plasencia reported on key-recovery/state-recovery and distinguishing attacks on Achterbahn-128/80, improving on previous attacks by Hell and Johansson [HJ06]. The distinguishing part of her attack on Achterbahn-80 has complexity  $2^{74}$  operations and requires  $2^{61}$  key stream bits. As for the (distinguishing part of the) attack on Achterbahn-128, it requires  $2^{80.4}$  operations and  $2^{60}$  key stream bits.

In [CP07], Cho and Pieprzyk presented a linear distinguisher for the Dragon stream cipher. The distinguisher exploits biases of two S-boxes and the modular addition. This attack allows to distinguish Dragon from a random bitstream using  $2^{150.6}$  keystream bytes (which is, nonetheless, larger than the maximum amount of keystream available) and memory complexity  $2^{59}$ .

In [WP07], Wu and Preneel describe a differential-linear attack against the Phelix stream cipher, using  $2^{34}$  chosen nonces and  $2^{37}$  chosen plaintext words. The attack complexity is  $2^{41.5}$

operations.



## Chapter 4

# Trade-offs in generic attacks on stream ciphers

The literature describes several brute-force generic attacks on stream ciphers<sup>1</sup>. From a high-level point of view, the attacks can be described as a sequence of two phases:

**Precomputation phase:** The results of the computations performed in this phase can be reused for several iterations of the attack. Typically, the results are some tables which are used to speed up the last phase of the attack. We denote the binary logarithm of the computational complexity of this phase by  $P$ .

**Online phase:** The attacker collects data, which usually consists of ciphertexts and corresponding known or chosen plaintexts. We count the amount of data collected in units of  $k$  bits, and denote the binary logarithm of this quantity by  $D$ . This data is combined with the results of the precomputation phase to recover the key. We denote the binary logarithm of the computational complexity of this phase by  $T$ .

We denote the binary logarithm of the sum of the memory requirements of the precomputation phase and the online phase of the attack by  $M$ .

### 4.1 Exhaustive attacks

For a straightforward exhaustive key search, there is no precomputation, the attacker collects a negligible amount of known plaintexts, and uses a negligible amount of memory ( $P = D = M \approx 0$ ). The online computational complexity is given by  $T = k$ .

On the other hand, we can also imagine an attack scenario where the attacker precomputes the ciphertext for a given chosen plaintext under all possible keys, and stores the result in a table. This gives  $P = M = k$ . The online phase of the attack consists of obtaining the ciphertext corresponding to the chosen plaintext and looking up the key in the table ( $D \approx T = 0$ ).

**Notes:**

1. For an SSC a chosen-plaintext attack can always be replaced by a known-plaintext attack with the same complexities.

---

<sup>1</sup>The content of this chapter was published in [Rij10]

2. Some authors argue that it is not realistic to say that the time to look up data in a table doesn't depend on the size of the table [Ber05]. All authors agree that memory cost and computation cost are very different things. Hence, when comparing various trade-off curves or points on a given trade-off curve, one has to be careful. A point with much higher  $T$ , but slightly decreased  $M$  compared to its alternatives, might well be the best choice.

Between the two extreme cases formed by exhaustive key search and a full table based attack, several trade-off scenarios can be defined. They are discussed next. We denote the key length (in bits) by  $k$ . The classical time-memory trade-off described by Hellman [Hel80] has the following complexities:

$$T = M = 2k/3, P = k \text{ and } D = 0,$$

which can be generalized to

$$T + 2M = 2k, P = k \text{ and } D = 0.$$

This trade-off works for any one-way function, hence also for stream ciphers. The attack recovers the secret key.

## 4.2 Time-Memory-Data trade-offs

For stream ciphers with an unkeyed output transformation, it is possible to define trade-off attacks targeting the internal state instead of the the key. We denote the size of the internal state (in bits) by  $s$ . Babbage and Golić (BG) [Bab95, Gol97] describe time-memory-data trade-off attacks with:

$$T + M = s, D = T \text{ and } P = M. \quad (4.1)$$

Biryukov and Shamir (BS) [BS00] describe time-memory-data trade-off attacks targeting the internal state with:

$$T + 2M + 2D = 2s, T \geq 2D \text{ and } P + D = s. \quad (4.2)$$

In 'practice,' one usually chooses  $T = 2D$  for the Biryukov-Shamir trade-off. If the state size is at least twice the key length, then both the Babbage-Golić trade-offs and the Biryukov-Shamir trade-offs become less efficient than the previously described attacks.

## 4.3 Time-Memory-Key trade-offs

In [HS05] a new type of trade-off is considered: the time-memory-key trade-off. The attack works in a scenario where an attacker can obtain a large number of short key streams, produced with different keys, but the same IV. The attack recovers one of the keys.

We denote by  $K$  the binary logarithm of the number of keys that are attacked in parallel. The trade-offs (4.1) and (4.2) become now:

$$T + M = k, K = T \text{ and } P = M, \quad (4.3)$$

$$T + 2M + 2K = 2k, T \geq 2K \text{ and } P + K = k. \quad (4.4)$$

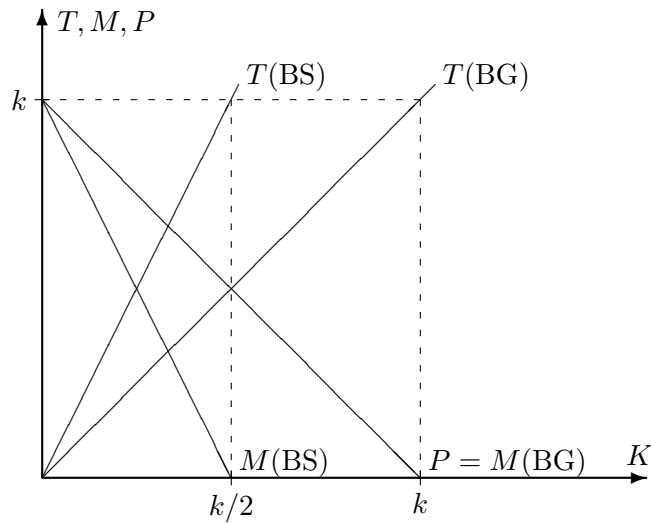


Figure 4.1: Time-Memory-Key trade-offs for the Babbage-Golić (BG) and for the Biryukov-Shamir (BS) curves. The precomputation ( $P$ ) curve is the same for both cases, and equal to the BG memory  $M$  curve.

The data complexity of the attacks equals 1  $k$ -bit string per key. The trade-off (4.4) is generalized in [BMS06] to:

$$T + 2M + 2K = (3 - \epsilon)k \text{ and } P + K = k \text{ with } 0 \leq \epsilon \leq 1.$$

Figure 4.1 compares the Babbage-Golić trade-offs and the Biryukov-Shamir trade-offs. For increasing  $K$ , the Biryukov-Shamir has a faster decreasing memory complexity, but a slower decreasing computational complexity. It follows that the Biryukov-Shamir approach is better adapted to the current state of computing technology, where computations are cheaper than memory.



## Chapter 5

# Implications of Related-Key Attacks

### 5.1 Related-key attacks in theory

Let  $B(k, x)$  denote a block cipher with key input  $k$  and plaintext input  $x$ .

The most traditional definition of security for a block cipher is the *Pseudo-Random Permutation (PRP)* model, where an adversary needs to distinguish the block cipher with a randomly selected key from a random permutation. In this model, an adversary can query the block cipher  $B$  output for different plaintext inputs  $x_i$  and one fixed key  $k$ :

$$y_i = B(k, x_i).$$

In the related-key attack model, the adversary can query the block cipher  $B$  output for different plaintext inputs  $x_i$  and different key inputs  $k_i$ . However, the key inputs are not under full control of the adversary. Instead, the adversary can choose a function  $f_i$  on the key space and obtain the ciphertexts:

$$y_i = B(f_i(k), x_i).$$

As has been observed in [BK03], in order to make possible a sound definition of security in this model, the set of admissible functions  $f_i$  has to be restricted. The conditions *output-inpredictability* and *collision resistance* are introduced in [BK03]. If these conditions are not satisfied, then security can't be defined.

Already in [Bih02] it was observed that if an adversary can query a block cipher under different keys, then, even if the keys are independently uniformly distributed, the adversary will need a lower number of queries than in a model where the key is fixed. This implies that we have to take special precautions when defining the advantage of an adversary in a related-key or any other multi-key model.

### 5.2 Related-key attacks in practice

IBM's key control vector mechanism and certain legacy banking protocols include functionality that can be abused to mount related-key attacks where the functions  $f_i$  are restricted to xoring with a known value

$$f_i(k_i) = k_i \oplus c_i.$$

Hence, this type of related-key attacks can be considered to be practical. Note also that the set of functions defined by letting the constants  $c_i$  take any value satisfies also the conditions output-inpredictability and collision resistance.

For the more general related-key attacks like the ones described in the next section, there is no protocol or application known where the attacks could be considered to be practical.

### 5.3 Related-key attacks and security claims

There exist a couple of methods to prove a design secure against differential and linear cryptanalysis, or at least basic versions of these attacks. Examples include provable security [NK95, Mat96], the wide trail design strategy [DR02], decorrelation [Vau03]. All of these methods assume a fixed key. Although many of the ciphers designed according to any of these methodologies include a key schedule that will provide some resistance against related-key attacks, the security claims and proofs are not valid in a related-key model.

## 5.4 Outline of recent attacks

### 5.4.1 Attacks on AES

A series of related-key attacks on AES [BKN09, BK09, BDK<sup>+</sup>10] drawn much attention in 2009. The authors heavily used the following new ideas

- The local collision technique from hash function cryptanalysis in order to get better differential trails.
- Use of non-trivial relations between the keys: difference in subkeys.
- Use of conforming pairs as a certificate of non-idealness of a cipher.
- Use of parallelism in round operations to get higher-probability boomerangs.

We describe them consecutively in the next paragraphs.

**Local collision.** A local collision is a short collision trail in the internal state, whose difference is controlled by the key schedule (message schedule in a hash function). The notion comes from the cryptanalysis of SHA-0, where a local collision is produced by injecting one-bit differences in particular positions of the six consecutive message blocks.

A local collision in AES-256 is produced as follows (Figure 5.1). The subkey has a difference in a single byte of the upper row, which is injected to a zero-difference internal state (*disturbance*). Then it is converted to an unknown difference by SubBytes, but is not touched by ShiftRows. Finally, the difference is expanded by MixColumns to a full column difference and is corrected by the next subkey addition. The appropriate column difference in the subkey can be predicted with probability up to  $2^{-6}$ , so the local collision trail has probability up to  $2^{-6}$ .

Due to the key schedule the differences spread to other subkeys thus forming the *key schedule difference*. The resulting key schedule difference can be viewed as a set of local collisions, where the expansion of the disturbance (also called *disturbance vector*) and the correction differences compensate each other. The probability of the full differential trail is



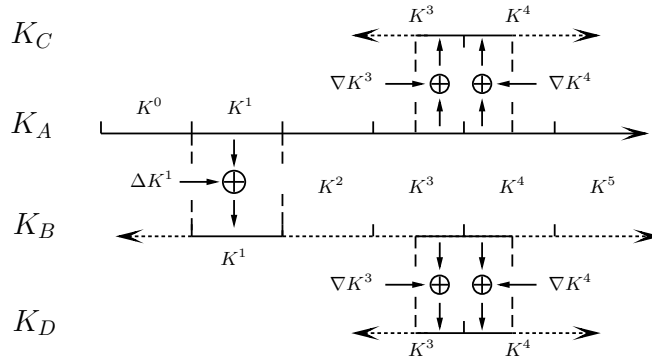


Figure 5.3: AES-256: Computing  $K_B$ ,  $K_C$ , and  $K_D$  from  $K_A$ .

demonstrated that AES actually exhibits properties that are unlikely to find in an ideal cipher with the same query complexity. This implies that an ideal cipher being instantiated with AES may have undesirable properties and not fulfill the randomness requirements.

The property that was exploited is called a differential  $q$ -multicollision, which is a set of  $q$  pairs of plaintexts conforming to the related-key differential trail. Such a set can be found for an ideal cipher with as many as  $O(q \cdot 2^{\frac{q-2}{q+2}n})$  queries, while for AES-256 it can be constructed with the complexity equivalent to  $q \cdot 2^{67}$  AES calls. A definition of a differential multicollision can be weakened to a so called partial multicollision, for which the complexities to be found in an ideal cipher and AES are  $2^{52}$  and  $2^{37}$ , respectively. This leads to a practical certificate of non-randomness. The complexity of all these attacks is significantly lower than the inverse of the probability of the respective differential trails due to the freedom in the choice of the key and the application of equation-solver tool in the most expensive part of a trail.

**Improvements in the boomerang attack.** The paper on boomerang attacks [BK09] formally introduced several tricks that are aimed to reduce the probability of a boomerang distinguisher. The tricks deal with the switch phase of the boomerang attack, which is the transition point between sub-ciphers  $E_0$  and  $E_1$  and also a place all the four differences of a quartet of states are explicitly fixed. The authors noted that the differential trails for the boomerang attack should be constructed so that

- The number of S-boxes in the switch round that are active in both trails should be as low as possible.
- The S-boxes that are active in both trails should possess the same input or output differences.

**Summary.** These ideas resulted in the following advances in the cryptanalysis of AES:

- Related-key boomerang attacks on AES-192 and AES-256 in the secret-key model.
- Practical-complexity attacks on up to 10 rounds of AES-256 in the secret-key model.
- Practical-complexity attacks on AES-256 in the chosen-key model.

### 5.4.2 Attacks on other ciphers

#### Kasumi

Kasumi is a block cipher used in the third-generation GSM telephony. It is a modification of a block cipher Misty and operates on a 64-bit block and a 128-bit key. The attack on Kasumi [DKS] is a related-key boomerang attack with  $2^{32}$  time and  $2^{26}$  data complexity. The authors managed to verify the attack on a PC and.

The attack greatly benefits from a formal treatment of the boomerang tricks, first introduced in [BK09] and [BCD03], and now called a *sandwich attack*. The idea is to separate the switch phase of the boomerang attack into a single transformation  $M$ , so that the whole cipher  $E$  is decomposed as  $E_1 \circ M \circ E_0$ . The sub-ciphers  $E_0$  and  $E_1$  have the differential trails

$$\Delta \xrightarrow{E_0} \Delta'; \quad \nabla' \xrightarrow{E_1} \nabla.$$

The authors consider the probability  $P_M$  that in the return phase of the boomerang the backward iteration of  $E_0$  gets the proper difference  $\Delta'$  if both iterations of  $E_1^{-1}$  give  $\nabla'$ :

$$P_M = \mathbb{P} \left\{ \text{Input Diff}(E_0^{-1}) = \Delta' \mid \begin{array}{l} \xrightarrow{E_0} \Delta' \\ \xrightarrow{E_1^{-1}} \nabla' \\ \xrightarrow{E_1^{-1}} \nabla' \end{array} \right\}.$$

In the regular boomerang attacks and in [BK09]  $P_M$  is equal to 1. The KASUMI paper demonstrates how to exploit the case  $P_M \neq 1$  and formally introduced the framework for the precise estimation of the distinguisher probability.

#### SQUARE

SQUARE is a block cipher with a 128-bit state and a 128-bit key. SQUARE follows the SPN structure, which is quite similar to that of AES. However, key schedule of SQUARE is linear and admits linear relation between related keys. The attacker is able to choose the disturbance difference in the local collision so that it is converted to the correction difference by a round of the key schedule. As a result [KYS], one can get a differential trail with zero-difference rounds before and after the local collision round (Figure 5.4). However, the probability of the local collision drops to  $2^{-28}$ , because the disturbance difference at first expanded by the linear transformation and only afterwards passes the S-box layer.

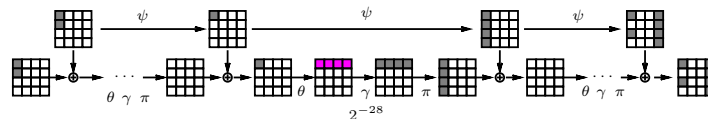


Figure 5.4: Related-key trail for SQUARE.

SQUARE has 8 rounds, so 4-round trails are long enough for the boomerang attack. The trails are produced from the 3-round trail in Figure 5.4, and the positions of the differences are carefully chosen in order to benefit from the tricks in the switch phase (similar to AES and KASUMI). The linearity of the key schedule also admitted a simple linear relation between keys.

**Other ciphers**

Several related-key attacks with little use of new techniques also appeared in 2009. The block cipher Threefish, the core of SHA-3 candidate Skein, the block cipher XTEA were analyzed with a simple boomerang attack with 33 and 36 rounds broken, respectively. The attacks use a linear relation between keys.

## Chapter 6

# Linear Cryptanalysis

Linear cryptanalysis is one of the most powerful statistical cryptanalysis tools using known plaintext/ciphertext pairs and it is based on probabilistic linear approximations. While the idea to use probabilistic approximations has already appeared in [TCG92] applied to FEAL [SM88], the theory of linear cryptanalysis was described and experimented on DES by Matsui [Mat94b, Mat94a].

**Notation.** In the following, we will denote random variables  $X, Y, \dots$  by capital letter and their realizations  $x \in \mathcal{X}, y \in \mathcal{Y}, \dots$  by small letters. The probability function of a variable  $X$  following a distribution  $D$  will be denoted  $\Pr_D[x]$ . If the distribution or the distribution and the random variable are clear from the context we can write  $\Pr_X[x]$  or  $\Pr[x]$ , respectively. A sequence  $X_1, X_2, \dots, X_N$  of independent and identically-distributed (i.i.d.) random variables of length  $N$  is denoted by  $\{X_i\}_{i=1}^N$ . For a realization of such a sequence we define by  $\hat{N}(\{x_i\}_{i=1}^N | \eta) = \#\{1 \leq i \leq N : x_i = \eta\}$  the relative frequency of  $\eta \in \mathcal{X}$ . If the sequence is clear from the context we can write only  $\hat{N}(\eta)$ . For  $\mathcal{X} = \mathbb{F}_2^n$ , we can represent  $x \in \mathcal{X}$  as a bit vector  $\mathbf{x}$ . A vector representation is displayed by bold or Greek letters.

Let us denote  $\mathbf{P}$ ,  $\mathbf{C}$ , and  $\mathbf{K}$  the plaintext, ciphertext and key, respectively. Given the bit masks  $\alpha$  and  $\gamma$ , we consider the following relation:

$$\alpha^T \begin{pmatrix} \mathbf{P} \\ \mathbf{C} \end{pmatrix} = \gamma^T \mathbf{K}.$$

We will use brackets to consider specific bits, e.g.  $\mathbf{P}[i]$  denotes the  $i$ th bit of  $\mathbf{P}$ . We use the DES notation for Feistel ciphers and note by  $\mathbf{P}_H$  the left block of the plaintext,  $\mathbf{P}_L$  the right block,  $\mathbf{C}_H$  the left block of the ciphertext and  $\mathbf{C}_L$  the right block. Let  $\{(\mathbf{p}_i, \mathbf{c}_i)\}_{i=1}^N$  be a sequence of plaintext/ciphertext pairs. We will use the following centered counters of the approximation:

$$\hat{u}_\alpha = \#\{(\mathbf{p}_i, \mathbf{c}_i) : \alpha^T \begin{pmatrix} \mathbf{p}_i \\ \mathbf{c}_i \end{pmatrix} = 0\} - \#\{(\mathbf{p}_i, \mathbf{c}_i) : \alpha^T \begin{pmatrix} \mathbf{p}_i \\ \mathbf{c}_i \end{pmatrix} = 1\}.$$

We note the distribution function of the standard normal distribution by  $\phi(x) = \int_{-\infty}^x \frac{e^{-t^2/2}}{\sqrt{2\pi}} dt$ .

**Principle.** Given a relation

$$\alpha^T \begin{pmatrix} \mathbf{P} \\ \mathbf{C} \end{pmatrix} = \gamma^T \mathbf{K}.$$

occurring with a probability  $p = \frac{1}{2} + s = \frac{1}{2}(1 + \epsilon)$  we denote  $s$  as the bias of the approximation and  $\epsilon$  as the correlation. The bias was the quantity used historically but it has been replaced by correlation and we will express everything by means of correlation in this summary. We present first the two algorithms invented by Matsui before presenting variants of these algorithms.

## 6.1 Piling-up lemma

To build linear approximations of a cipher we use the following theorem, called the Piling-up lemma:

**Theorem 6.1.1** *Let  $\{X_i\}_{i=1}^N$  be binary statistically independent variables such that the correlation of  $X_i$  is  $\epsilon_i$ . Then  $\oplus_i X_i$  is a random boolean variable with a correlation  $\prod_i \epsilon_i$ .*

Hence it is possible to concatenate linear approximations. This can be used for building a linear approximation of a cipher using approximations of components. Nevertheless, in practice the independence hypothesis is not always valid, so the estimation given should be checked on each algorithm. For example, for DES [FIP99], it gives precise results while for other algorithms, such as RC5 [Riv95], an inner dependency provokes wrong results applying the piling up lemma [Sel98]. This kind of problem exists particularly when there are several consecutive active rounds.

## 6.2 Matsui Algorithm 1

The idea of Algorithm 1 is the following: if the absolute value of the correlation  $|\epsilon|$  is sufficiently large compared to the number of plaintext/ciphertext pairs, then the value of  $\gamma^T \mathbf{k}$  can be recovered. Thus we have found a bit of information of the key. More precisely, if  $\gamma^T \mathbf{k} = 0$  then  $\alpha^T \begin{pmatrix} \mathbf{P} \\ \mathbf{C} \end{pmatrix}$  (seen as a binary variable) will be distributed with a mean  $\frac{1}{2}(1 + \epsilon)$  and a variance  $\frac{1}{4} - 4\epsilon^2$  whereas if  $\gamma^T \mathbf{k} = 1$  then  $\alpha^T \begin{pmatrix} \mathbf{P} \\ \mathbf{C} \end{pmatrix}$  will be distributed with a mean  $\frac{1}{2}(1 - \epsilon)$  and a variance  $\frac{1}{4} - 4\epsilon^2$ . To distinguish the two distributions, Matsui's Algorithm 1 works as follows:

1. Take  $N$  plaintext/ciphertext pairs.
2. Compute the centered counter  $\hat{u}_\alpha$ .
3. If  $\hat{u}_\alpha > 0$  then guess  $\gamma^T \mathbf{k} = 0$  (if  $\epsilon > 0$ ) or  $\gamma^T \mathbf{k} = 1$  (if  $\epsilon < 0$ ) otherwise guess  $\gamma^T \mathbf{k} = 1$  (if  $\epsilon > 0$ ) or  $\gamma^T \mathbf{k} = 0$  (if  $\epsilon < 0$ ).

The probability of success of Algorithm 1 is given by the following theorem (with a numeric application in Table 6.1)

**Theorem 6.2.1** *If  $|\epsilon|$  is sufficiently small then the probability of success of Algorithm 1 is  $\phi(\sqrt{N}\epsilon)$ .*

$N$	$\epsilon^{-2}$	$2\epsilon^{-2}$	$4\epsilon^{-2}$	$8\epsilon^{-2}$
Probability of success	84.1%	92.1%	97.7%	99.8%

Table 6.1: Probability of success of Algorithm 1.

### 6.3 Matsui Algorithm 2

Algorithm 2 uses the approximation as a distinguisher on inner rounds to recover external key bits. If the guess on the external key bits is correct then we should see the expected correlation while if the guess is wrong then the linear approximation should occur with a probability  $\frac{1}{2}$  because the link between the plaintext and the ciphertext will be random. Therefore we should distinguish a distribution with a mean  $\frac{1}{2}(1 \pm \epsilon)$  among a lot of different distributions with a mean  $\frac{1}{2}$ . This attack recovers more key bits than Algorithm 1 and uses an approximation over fewer rounds so it gives more practical attacks. Algorithm 2 works as following:

1. Take  $N$  plaintext/ciphertext pairs.
2. For each possible external key  $\mathbf{k}$ , do a partial encryption/decryption over external rounds to compute the associated counter  $\hat{u}_\alpha^{\mathbf{k}}$ .
3. Guess the value of the key which gives the greatest absolute value of  $\hat{u}_\alpha^{\mathbf{k}}$ . The sign of  $\hat{u}_\alpha^{\mathbf{k}}$  gives also a guess on one internal key bit.

Other key bits can be recovered with a brute force attack. This algorithm gives an attack on the full DES using  $2^{47}$  plaintext/ciphertext pairs and a probability of success of 96.7% in [Mat94b] with a partial decryption on one round and the following linear approximation over 15 rounds with a correlation  $-2^{-21.75}$ :

$$\alpha^T \begin{pmatrix} \mathbf{P} \\ \mathbf{C} \end{pmatrix} = \mathbf{P}_H[7, 18, 24] \oplus \mathbf{P}_L[12, 16] \oplus \mathbf{C}_H[7, 18, 24, 29] \oplus \mathbf{C}_L[15] \text{ and}$$

$$\gamma^T \mathbf{K} = \mathbf{K}_1[19, 23] \oplus \mathbf{K}_3[22] \oplus \mathbf{K}_4[44] \oplus \mathbf{K}_5[22] \oplus \mathbf{K}_7[22] \oplus \mathbf{K}_8[44]$$

$$\oplus \mathbf{K}_9[22] \oplus \mathbf{K}_{11}[22] \oplus \mathbf{K}_{12}[44] \oplus \mathbf{K}_{13}[22] \oplus \mathbf{K}_{15}[22].$$

An important modification of this algorithm was introduced in [Mat94a], is named key ranking and consist in:

1. Taking  $N$  plaintext/ciphertext pairs.
2. For each possible external key  $\mathbf{k}$ , do a partial encryption/decryption over external turns to compute the associated counter  $\hat{u}_\alpha^{\mathbf{k}}$ .
3. Rank the counters  $\hat{u}_\alpha^{\mathbf{k}}$  with their absolute value.
4. Take the  $\hat{u}_\alpha^{\mathbf{k}}$  with the maximum absolute value and try to find with exhaustive search the rest of the key bits. If it is impossible, try again with the  $\hat{u}_\alpha^{\mathbf{k}}$  with the second greatest absolute value and so on.

An application of this attack was an attack over the full version of DES using  $2^{43}$  plaintext/ciphertext pairs and a probability of success of 85% in [Mat94a] using the two following linear approximations over 14 rounds (each of them gives different external key bits):

$$\begin{aligned} \alpha^T \begin{pmatrix} \mathbf{P} \\ \mathbf{C} \end{pmatrix} &= \mathbf{P}_L[7, 18, 24] \oplus \mathbf{C}_L[15] \oplus \mathbf{C}_H[7, 18, 24, 29] \text{ and} \\ \gamma^T \mathbf{K} &= \mathbf{K}_2[22] \oplus \mathbf{K}_3[44] \oplus \mathbf{K}_4[22] \oplus \mathbf{K}_6[22] \oplus \mathbf{K}_7[44] \oplus \mathbf{K}_8[22] \\ &\quad \oplus \mathbf{K}_{10}[22] \oplus \mathbf{K}_{11}[44] \oplus \mathbf{K}_{12}[22] \oplus \mathbf{K}_{14}[22] \text{ together with} \\ \alpha^T \begin{pmatrix} \mathbf{P} \\ \mathbf{C} \end{pmatrix} &= \mathbf{C}_L[7, 18, 24] \oplus \mathbf{P}_L[15] \oplus \mathbf{P}_H[7, 18, 24, 29] \text{ and} \\ \gamma^T \mathbf{K} &= \mathbf{K}_{13}[22] \oplus \mathbf{K}_{12}[44] \oplus \mathbf{K}_{11}[22] \oplus \mathbf{K}_9[22] \oplus \mathbf{K}_8[44] \\ &\quad \oplus \mathbf{K}_7[22] \oplus \mathbf{K}_5[22] \oplus \mathbf{K}_4[44] \oplus \mathbf{K}_3[22] \oplus \mathbf{K}_1[22]. \end{aligned}$$

In [Mat94a] Matsui estimates the complexity of this attack as  $2^{43}$  DES encryptions. This complexity is an overestimation as was indicated in [Jun01] by Junod who showed experimentally that it was  $2^{41}$  DES encryptions. An important optimization concerning the computation of experimental correlations based on the FFT was presented by Collard et al. in [CSQ07].

### 6.3.1 Probability of success of Algorithm 2

The computation of the probability of success of Algorithm 2 is much more complex and depends in the general case on the correlation and the number of external key bits recovered in the attack. A first estimate was given by Matsui [Mat94b], has then been extended by Junod [Jun01] and was finally given by Selçuk [Sel08]. The important notion to evaluate this probability is the notion of the  $a$ -bit advantage, defined as follows

**Definition 6.3.1** *If, after the last step of the algorithm, the correct key belongs to the first  $2^r$  keys among the  $2^m$  candidates, we say that we have an  $(m - r)$ -bit advantage.*

In [Sel08] Selçuk proves the following theorem:

**Theorem 6.3.1** *Let  $P$  be the probability that Matsui Algorithm 2, with an  $m$ -bit external key, a linear approximation with correlation  $\epsilon$ , and  $N$  known plaintext/ciphertext pairs, gives an  $a$ -bit advantage. Suppose that the linear approximations made by each key guess are statistically independent and their probability is  $\frac{1}{2}$  for each wrong key guess, then for sufficiently large  $m$  and  $N$*

$$P = \phi(\sqrt{N}\epsilon - \phi^{-1}(1 - 2^{-a-1})).$$

## 6.4 Linear hull

Introduced by Nyberg [Nyb95], the linear hull is an effect appearing when different linear trails with the same plaintext and ciphertext masks coexist. It influences the correlation of the binary variable  $\alpha^T \begin{pmatrix} \mathbf{P} \\ \mathbf{C} \end{pmatrix}$  and so linear hulls have been an important topic of research in linear cryptanalysis. A detailed survey and bibliography is given in [Kel03].

## 6.5 Deriving methods

Several methods have been derive from linear cryptanalysis, among them we can underline a few. Using some assumptions on the plaintext distribution allows to build only ciphertext attacks [Mat94b]. Furthermore, some assumptions over the plaintext distribution using chosen plaintext/ciphertext pairs allows to reduce the correlation of targeted approximation [KM01]. Combining linear cryptanalysis with differential cryptanalysis is a quite effective cryptanalysis against many ciphers [LH94, BDK02] and using non linear approximations [KR96, SK98] is also a possible direction.



## Chapter 7

# Multiple-Linear and Multidimensional Linear Cryptanalysis

Basic linear cryptanalysis uses one linear approximation. The question is, can we improve the attacks by using several approximations. In this section, we consider the different developments in this direction.

We will use the same notations as in Section 6. Let us assume that we have  $m$  approximations

$$\alpha_i^T \begin{pmatrix} \mathbf{P} \\ \mathbf{C} \end{pmatrix} = \gamma_i^T \mathbf{K},$$

for  $1 \leq i \leq m$  each with a correlation  $\epsilon_i$ . We can combine the result by considering the result as words in  $\mathbb{F}_2^m$ . Then we have

$$\tilde{\alpha}^T \begin{pmatrix} \mathbf{P} \\ \mathbf{C} \end{pmatrix} = \tilde{\gamma}^T \mathbf{K},$$

where  $\tilde{\alpha}$  is a  $m \times (\log_2 |\mathcal{P}| + \log_2 |\mathcal{C}|)$  matrix and  $\tilde{\gamma}$  a  $m \times \log_2 |\mathcal{K}|$  matrix.

Let  $Y_i = \alpha_i^T \begin{pmatrix} \mathbf{P} \\ \mathbf{C} \end{pmatrix} \oplus \gamma_i^T \mathbf{K}$  be the binary random variable corresponding to the linear approximations defined by  $(\alpha_i, \gamma_i)$ . Then, two linear approximations corresponding to  $(\alpha_i, \gamma_i)$  and  $(\alpha_j, \gamma_j)$  are *statistically independent* if for  $\mathbf{P}, \mathbf{C}, \mathbf{K}$  randomly distributed, the variables  $Y_i, Y_j$  are independent.

There are mainly two approaches, multiple linear approximation and multidimensional linear approximation. The difference is that the first one assumes statically independent approximations.

### 7.1 Multiple Linear Cryptanalysis (Biryukov Method)

Already in [Mat94a, JV03], two statistically independent approximations were used for the key ranking in Algorithm 2. In [KR94], Kalisky and Robshaw used  $m$  different approximations  $\alpha_i$  for the plaintext-ciphertext pairs and the same mask  $\gamma = \gamma_i$  for the key. Assuming that the approximations were statistically independent, they were able to reduce the amount of data needed for Algorithm 1 and 2.

Biryukov et al. [BCQ04] proposed a method using  $m$  statically independent approximations, where the mask for the plaintext-ciphertext pairs as well as for the key vary. Since the approach of Kalisky and Robshaw can be seen as a special case of this method, we use the term *Biryukov method* to refer to the use of independent approximations.

For the  $i$ 'th approximation and  $N$  plaintext-ciphertext pairs, let  $\epsilon_i$  be the correlation of  $\alpha_i^T \begin{pmatrix} \mathbf{P} \\ \mathbf{C} \end{pmatrix}$  and  $\hat{\epsilon}_i = \frac{\hat{u}_{\alpha_i}}{N}$  the empirical correlation.

For the extension of Matsui's Algorithm 1, Biryukov et al. partition the keyspace to  $2^m$  subgroups according to the result of  $\mathbf{z} = \tilde{\gamma}^T \mathbf{k}$ . The goal is to find the right subgroup and thus gain  $m$  bits of information of the key. In an on-line phase, the empirical correlations of the  $m$  approximations are computed. Finally, in the off-line phase, the subgroup corresponding to  $\mathbf{z} = (z_1, z_2, \dots, z_m)$  is chosen for which the sum

$$g(\mathbf{k}) = \sum_{i=1}^m ((-1)^{z_i} \epsilon_i - \hat{\epsilon}_i)^2$$

is minimized. The time and memory complexity is  $mN$  in the on-line phase, and  $m2^m$  and  $m$  in the off-line phase. Under the assumption of independent approximations, Biryukov et al. claim that  $N$  is proportional to  $1/\sum_{i=1}^m \epsilon_i^2$ .

For Algorithm 2, we have a sequence of plaintext-ciphertext pairs  $(\mathbf{p}_1, \mathbf{c}_1), (\mathbf{p}_2, \mathbf{c}_2), \dots, (\mathbf{p}_N, \mathbf{c}_N)$  encrypted over several rounds. Let  $\mathbf{k}$  denote the main key and  $\mathbf{k}_r$  the roundkey used for the encryption of the last round, and  $f$  the round function. We define by  $\hat{\epsilon}_i^{\mathbf{k}_r}$  the empirical correlation of the pairs  $(\mathbf{p}_j, f^{-1}(\mathbf{c}_j, \mathbf{k}_r))$ . The attack tries to minimize the following sum:

$$g(\mathbf{k}_r, \mathbf{z}) = \sum_{i=1}^m ((-1)^{z_i} \epsilon_i - \hat{\epsilon}_i^{\mathbf{k}_r})^2 + \sum_{\mathbf{k}'_r \neq \mathbf{k}_r} \sum_{i=1}^m (\hat{\epsilon}_i^{\mathbf{k}'_r})^2.$$

The first part of the sum is minimized for the right key subclass, and the second sum is minimized for the right last round key.

The problem of this approach is, that it assumes statistically independent approximations. Murphy showed in [Mur06] that this is not true in the general case. More specific, Hermelin et al. [HCN08] pointed out that in the case of reduced round Serpent the strong approximations are normally not independent.

## 7.2 Multidimensional Linear Cryptanalysis

In [JM03], Johansson and Maximov apply for the first time a multidimensional analysis on the stream cipher Scream. Baignères et al. develop in [BJV04] a statistical analysis of multidimensional approximations, without the assumption of statistical independence. They consider the case where a sequence  $X_1, X_2, \dots, X_N$  of i.i.d. random variables is drawn either from distribution  $D_0$  or from distribution  $D_1$ . In our case,  $D_0$  might be the uniform distribution over  $2^m$  and  $D_1$  the distribution according to the cipher and the right key. Baignères et al. show that the optimal statistic to distinguish between the two distributions is the Log-likelihood ratio

$$LLR(\{x_i\}_{i=1}^N, D_0, D_1) = \sum_{i=1}^N \log_2 \frac{\Pr_{D_1}[x_i]}{\Pr_{D_0}[x_i]} = \sum_{\eta \in \mathcal{X}} \hat{N}(\{x_i\}_{i=1}^N | \eta) \log_2 \frac{\Pr_{D_1}[\eta]}{\Pr_{D_0}[\eta]}.$$

The data complexity  $N$  is inversely proportional to  $\sum_{\eta \in \mathcal{X}} \frac{(\Pr_{D_0}[\eta] - \Pr_{D_1}[\eta])^2}{\Pr_{D_0}[\eta]}$ . If  $D_0$  is uniformly distributed, this value is equivalent to the capacity  $C(D_1)$  which is defined as follows:

**Definition 7.2.1** For a distribution  $D$  over the set  $\mathcal{X}$  of size  $2^m$ , the capacity of  $D$  is

$$C(D) = 2^m \sum_{\eta \in \mathcal{X}} (\Pr_D[\eta] - 2^{-m})^2.$$

So given the  $m$ -dimensional distribution of  $D_1$  we have an optimal distinguisher. However, Baignères et al. did not consider how to find this distribution.

Englund and Maximov [EM05] tried to find the probability distribution over the whole stream cipher Dragon. However, this method is only feasible if the word size is not larger than 32 bits.

In [HCN08] Hermelin et al. showed that for finding the probability distribution  $D$  of the  $m$ -dimensional random variable  $\mathbf{X}$ , it is sufficient to know the one dimensional correlations for all values  $\beta \in \mathbb{F}_2^m$ .

**Theorem 7.2.1** Let  $\mathbf{X}$  be a random variable over  $\mathbb{F}_2^m$  with distribution  $D$ ,  $\beta \in \mathbb{F}_2^m$ , and let  $\epsilon_\beta$  denote the correlation of  $\beta^T \mathbf{X}$ . Then, the probability distribution  $D$  is given by

$$\Pr_D[\eta] = 2^{-m} \sum_{\beta \in \mathbb{F}_2^m} (-1)^{\beta^T \eta} \epsilon_\beta.$$

This property can be used as well for deriving the capacity of  $D$  by

$$C(D) = \sum_{\beta \in \mathbb{F}_2^m \setminus \{0\}} \epsilon_\beta^2.$$

### Multidimension Extension of Matsui's Algorithm 1

Let  $\mathbf{Z}$  be the  $m$ -bit random variable  $\tilde{\gamma}^T \mathbf{K}$ , and let  $D$  be the probability distribution of

$$\tilde{\alpha}^T \begin{pmatrix} \mathbf{P} \\ \mathbf{C} \end{pmatrix} \oplus \tilde{\gamma}^T \mathbf{K} = \tilde{\alpha}^T \begin{pmatrix} \mathbf{P} \\ \mathbf{C} \end{pmatrix} \oplus \mathbf{Z}.$$

Then, for every  $\mathbf{z} \in \mathbb{F}_2^m$  the variable  $\tilde{\alpha}^T \begin{pmatrix} \mathbf{P} \\ \mathbf{C} \end{pmatrix}$  has distribution  $D_{\mathbf{z}}$  which is a fixed permutation of  $D$ . Let  $\hat{D} = \left( \hat{q}_\eta = \frac{\hat{N}(\eta)}{N} \right)_{\eta \in \mathbb{F}_2^m}$  be the empirical distribution of  $\mathbf{Z}$ . The goal is to find  $\mathbf{z}$  such that  $\hat{D} = D_{\mathbf{z}}$ . We remark that  $C(D_{\mathbf{z}}) = C(D)$  for all  $\mathbf{z}$ . In the on-line phase, the test obtains the empirical distribution  $\hat{D}$ . This has a data, time and memory complexity of  $N$ ,  $mN$  and  $2^m$ , respectively.

In [HCN08], Hermelin et al. presented a multidimensional linear attack on reduced round Serpent based on a log-likelihood statistic. The algorithm outputs the value  $\mathbf{z}'$  for which the  $D_{\mathbf{z}'}$  is closest to the empirical distribution in Kullback-Leibler distance, i.e. for which the sum

$$h_1(\mathbf{z}) = \sum_{\eta \in \mathbb{F}_2^m} \hat{q}_\eta \log \frac{\hat{q}_\eta}{\Pr_{D_{\mathbf{z}}}[\eta]}$$

is minimized. They compared the empirical results, with the experimental results of Collard et al. [CSQ08] using the Biryukov method. It was shown that the multidimensional method was more efficient for the same amount of data.

In a next step, Hermelin et al [HCN09b] compared the log-likelihood method with a method minimizing the  $\chi^2$  statistic

$$h_2(\mathbf{z}) = \sum_{\eta \in \mathbb{F}_2^m} \frac{(\hat{q}_\eta - \text{Pr}_{D_{\mathbf{z}}}[\eta])^2}{\hat{q}_\eta},$$

and method based on the optimal distinguishing technique in [BJV04]. The last method outputs  $\mathbf{z}$  for which

$$h_3(\mathbf{z}) = \sum_{\eta \in \mathbb{F}_2^m} \hat{N}_\eta \log \frac{\text{Pr}_{D_{\mathbf{z}}}[\eta]}{2^{-m}}$$

is maximized. The authors could show that for all three algorithms the time and memory complexity of the off-line phase is  $2^{2m}$  and  $2^m$ , respectively. Theoretical analysis showed that in the on-line phase, the first two methods, log-likelihood and  $\chi^2$ , have a data complexity of  $N = 2^{m/2}/C(D)$ , whereas the last method only has a complexity of  $m/C(D)$ . However, empirical results showed that all three tests performed equally well with a data complexity of  $m/C(D)$ .

In [HN10], Hermelin and Nyberg introduced an additional test statistic based on the convolution of probability distributions. The algorithm outputs  $\mathbf{z}$  which maximizes the convolution of  $\hat{D}$  and  $D$  at point  $\mathbf{z}$

$$h_4(\mathbf{z}) = (q * D)_{\mathbf{z}}.$$

This value can be computed efficiently by Fast Fourier Transform (FFT) in  $m2^m$ . The authors compare the test to a statistic that they call full Biryukov method. In contrary to Biryukov's method, it's statistic  $g'(\mathbf{z}) = \sum_{\beta \in \mathbb{F}_2^m} ((-1)^{\mathbf{z} \cdot \beta} \epsilon_\beta - \hat{\epsilon}_\beta)^2$  uses all linear combinations of the approximations and does not need the statistical independence assumption. The ranking function of the convolution method can be expressed by means of one dimensional correlations:

$$h_4(\mathbf{z}) = 2^{-m} \sum_{\beta \in \mathbb{F}_2^m} (-1)^{\beta^T \mathbf{z}} \epsilon_\beta \hat{\epsilon}_\beta.$$

Thus, we can see that a ranking done by  $h_4$  is equivalent to a ranking done  $g'$ . The data, time and memory complexity of the on-line phase is the same for the full Biryukov method and the convolution method, namely  $N$ ,  $mN$  and  $2^m$ , where  $N$  is proportional to  $m/C(D)$ . In the off-line phase, the full Biryukov methods has a time and memory complexity of  $2^{2m}$  and  $2^m$ . By using FFT to compute the convolution in the off-line phase, this method has a time complexity of only  $m2^m$ , which is significantly better than for the other tests. The computation of the off-line phase in both methods can be reduced by considering only the strongest correlations.

The original Biryukov method has the same time complexity as the convolution method, however it has a higher data complexity.

## Multidimension Extension of Matsui's Algorithm 2

In Matsui's Algorithm 2, we have a sequence of plaintext-ciphertext pairs  $(\mathbf{p}_1, \mathbf{c}_1), (\mathbf{p}_2, \mathbf{c}_2), \dots, (\mathbf{p}_N, \mathbf{c}_N)$ . which are encrypted over several rounds. Let  $\mathbf{k}'_r$  be the actual

subkey for the last round of size  $\ell$  bits. Let  $f$  be the roundfunction. For any possible value  $\mathbf{k}_r \in \mathbb{F}_2^\ell$  we define

$$\mathbf{z}_j^{\mathbf{k}_r} = \tilde{\alpha}^T \left( \begin{array}{c} \mathbf{p}_j \\ f^{-1}(\mathbf{c}_j, \mathbf{k}_r) \end{array} \right).$$

Then we compute in an on-line phase for every  $\mathbf{z} \in \mathbb{F}_2^m$  the empirical probability distribution  $\left( \hat{q}_\eta^{\mathbf{k}_r} = \frac{\hat{N}(\{\mathbf{z}_j^{\mathbf{k}_r}\}_{j=1}^N|\eta)}{N} \right)_{\eta \in \mathbb{F}_2^m}$ . This can be done in a time and memory complexity of  $N + 2^{m+\ell}$

and  $2^{m+\ell}$ , by first counting the frequencies of the  $m + \ell$  bits in  $(\mathbf{p}_j, \mathbf{c}_j)$  which are necessary to compute  $\mathbf{z}_j^{\mathbf{k}_r}$  and doing the decryption only in the end. This part of the attack is the same for all methods. The differences lie in the off-line phase. In the following, let us assume that  $\mathbf{k}'$  and  $\mathbf{k}'_r$  are the correct key and roundkey and  $\mathbf{z}' = \tilde{\gamma}^T \mathbf{k}'$ .

In [HCN09a], Hermelin et al. consider two methods using  $\chi^2$  and LLR statistics. The  $\chi^2$  test makes no assumption on  $D_{\mathbf{z}}$ . It ranks the keys by the value

$$S_{\mathbf{k}_r} = 2^m N \sum_{\eta \in \mathbb{F}_2^m} (\hat{q}_\eta^{\mathbf{k}_r} - 2^{-m})^2$$

It uses the assumption that for the right key  $\mathbf{k}'_r$  the empirical distribution will have the largest distance from the uniform distribution. Let  $D$  be the distribution of  $\mathbf{z}_j^{\mathbf{k}'_r}$ , then we achieve an  $a$ -bit advantage with a data complexity of  $N = \sqrt{a2^m}/C(D)$  and a time and memory complexity of  $2^{m+\ell}$  and  $2^m + 2^\ell$ . By storing not only  $S_{\mathbf{k}_r}$  but the pair  $S_{\mathbf{k}_r}, \hat{q}_\eta^{\mathbf{k}_r}$  we increase the memory complexity to  $2^{m+\ell}$ , but we can apply subsequently Algorithm 1 to obtain  $\mathbf{z}'$ . An attack using the  $\chi^2$  method on the block cipher Present can be found in [Cho10].

The method using the LLR statistic of the optimal distinguishing problem searches for  $\mathbf{k}'_r$  and  $\mathbf{z}'$  at the same time. The ranking of  $\mathbf{k}_r$  is done by the value

$$L_{\mathbf{k}_r} = \max_{\mathbf{z} \in \mathbb{F}_2^m} \sum_{\eta \in \mathbb{F}_2^m} \hat{N}(\{\mathbf{z}_j^{\mathbf{k}_r}\}|\eta) \log \frac{\text{Pr}_{D_{\mathbf{z}}}[\eta]}{2^{-m}}.$$

For a given  $a$ -bit advantage, this method has a data complexity of  $N = (a + m)/C(D)$  and a time and memory complexity of  $2^{m+\ell}$  and  $2^{2m+\ell}$ .

## Applications to Stream Ciphers

There are mainly two sorts of multidimensional linear attacks on stream ciphers. The first one tries to distinguish the keystream from the output of a random source. Let  $\mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_N$  be the keystream of the cipher, then we consider the  $m$ -dimensional approximation

$$\mathbf{z}_t = \tilde{\alpha}^T \begin{pmatrix} \mathbf{Y}_t \\ \mathbf{Y}_{t+j_1} \\ \dots \\ \mathbf{Y}_{t+j_\ell} \end{pmatrix}$$

applied on  $\ell + 1$  keystream words for  $1 \leq t \leq N - j_\ell$ . The idea was first used by Maximov and Johansson in [JM03, MJ07] to the stream cipher Scream. They approximate an 8 to 8 bits non-linear S-box by a linear function. For the distinguisher, they assume an unknown distribution with a known statistical distance from uniform distribution. The statistical distance was

determined empirically. A distinguisher for a known probability distribution was first applied to the Dragon stream cipher by Englund and Maximov in [EM05]. They computed directly the probability distribution of  $\mathbf{Z}$ , which is only possible for small word sizes and small values of  $m$ . Once they had the distribution, they used the optimal distinguisher from [BJV04]. In [HN08], Hermelin and Nyberg studied the case of a keystream generator applying a non-linear filter function on an LFSR. They showed that vector bent functions [NH07] give the best resistance to multidimensional linear attacks. However, for this family of function the multidimensional analysis gains the most in comparison to the one-dimensional analysis. Hermelin and Nyberg use Walsh-Hadamard transform to efficiently compute the capacity of the approximation distribution. In [HN09], Hakala and Nyberg apply a multidimensional distinguisher on the Shannon stream cipher. This work was improved to a practical distinguisher by Ahmadian et al. in [AMS<sup>+</sup>ar].

The second approach can be used on ciphers with a linear state update function. By using a multidimensional linear approximation on the initial state and the keystream we can find  $m$  bits of the initial state. This method was applied in a key recovery attack on the stream cipher Grain by Berbain et al. in [BGM06] and on the stream cipher SOSEMANUK by Cho and Hermelin in [CHar].

## Chapter 8

# Cryptanalysis of ARX Structures

Increasingly, cryptographic primitives use operations such as addition modulo  $2^n$ , rotation and exclusive ORs (ARX), as well as bitwise Boolean functions. In NIST's SHA-3 hash function competition, this applies to 6 out of the 14 second-round candidates. In this report, we give an overview of recent developments in the field of ARX-based cryptography. We provide a summary of results in rotational cryptanalysis, a technique that was very recently used to attack reduced-round Threefish, the block cipher that is at the core of the Skein hash function. We also introduce a toolkit for ARX-based constructions. This toolkit calculates the probability that given input differences lead to given output differences. It can also count the number of possible output differences. The calculations can be efficiently performed using matrix multiplications.

Differential cryptanalysis [BS91] and linear cryptanalysis [MY92] are two of the most common methods in the cryptanalysis of block ciphers, hash functions and MACs.

For block ciphers, differential cryptanalysis [BS91] analyzes how plaintext input differences lead to output differences in the ciphertext. The dual of differential cryptanalysis is linear cryptanalysis [MY92]. In linear cryptanalysis, a linear approximation is made between the bits of the plaintext, key and ciphertext. Both differential cryptanalysis and linear cryptanalysis can be used to construct distinguishers or even key-recovery attacks for the block cipher.

To analyze the non-linear components of a cryptographic primitive, differential cryptanalysis typically involves the construction of a difference table. In this table, the number of occurrences for every combination of input and output differences is shown. For linear cryptanalysis, all linear approximations are enumerated in a linear approximation table. This is the standard approach for designs based on S-boxes.

Not all cryptographic primitives are based on S-boxes, however. It is also possible to achieve non-linearity by combining operations such as addition modulo  $2^n$ , exclusive OR (XOR), Boolean functions, bit rotations and bit shifts. For Boolean functions, it is assumed that the same Boolean function is used for each bit position  $i$  of the  $n$ -bit input words. All of these operations are very well suited for implementation in software, but constructing difference tables and linear approximation tables for them becomes impractical for  $n \geq 32$ .

Examples of ARX-based designs are the XTEA block cipher [NW97], the Salsa20 stream cipher family [Ber08], as well as the hash functions MD5, SHA-1. ARX and bitwise Boolean functions are also used in 6 out of the 14 second-round candidates of NIST's SHA-3 hash function competition [Nat07]: BLAKE [AHMP08], Blue Midnight Wish [GKK<sup>+</sup>09], Cube-Hash [Ber09], Shabal [BCCM<sup>+</sup>08], SIMD [LBF09] and Skein [FLS<sup>+</sup>09].

Cryptographic algorithms that involve addition, XOR and rotation, were referred to recently as AXR [Wei09], a name that was later changed to ARX. In this report, we describe the rotational cryptanalysis of ARX constructions, and introduce a toolkit for the efficient differential cryptanalysis of ARX.

## 8.1 Rotational Cryptanalysis

In [KN10], the concept of rotational cryptanalysis is explained for ARX constructions. Let us consider the pair  $(x, x \lll r)$ , consisting of both  $x$  and  $x$  rotated to the left by  $r$  positions. We refer to  $(x, x \lll r)$  as a rotational pair. Rotational cryptanalysis is based on the observation that if the inputs to the XOR, rotation or bitwise Boolean function operations are rotational pairs, the outputs are rotational pairs as well. With some probability, the same observation also holds for the addition operation. If rotational pairs can be obtained more easily for a given cryptographic primitive than for a random permutation, this observation can be used to build a distinguisher or even a key recovery attack.

The concept of rotational cryptanalysis is not new, but has recently gained a lot of interest because of the increasing number of ARX-based designs. In pioneering work by Biham [Bih94], a rotational pairs of keys were considered for the block ciphers LOKI89, LOKI91 and Lucifer. This approach was extended in [KSW97] to related-key attacks on several block ciphers. This is not pure rotational cryptanalysis, however, because the attacker searches for plaintexts of the form  $(p, F(p))$ , where  $F$  is the round transformation.

For Salsa20 [Ber08], Bernstein explicitly prevented attacks based on rotational pairs, by using constants without rotational symmetry at the input of the permutation. However, he did not give complexity estimates of such an attack.

In [MT09], a related-key attack was constructed using rotational pairs for a variant of the block cipher ESSENCE. This attack uses the observation that rotational pairs pass bitwise Boolean functions with probability one. For the linear function  $L$ , the following observation was made:

Let us use the polynomial representation of  $\mathbb{F}_{2^{32}}$ . A multiplication of any  $a \in \mathbb{F}_{2^{32}}$  by  $x$  then corresponds to a binary left shift by one position, and an XOR with the feedback polynomial if and only if the most significant bit of  $a$  is one. The following relation thus holds:

$$\text{MSB}(a) = 0 \Leftrightarrow a \cdot x = a \ll 1 . \quad (8.1)$$

The linear function  $L$  of ESSENCE is implemented as an LFSR. Using the polynomial representation of  $\mathbb{F}_{2^{32}}$ , we can write  $L(v) = v \cdot x^{32}$ .

We have that  $L(v \cdot x) = (v \cdot x) \cdot x^{32} = (v \cdot x^{32}) \cdot x = L(v) \cdot x$ . For a random  $v \in \mathbb{F}_{2^{32}}$ , with probability  $2^{-2}$  we have that  $\text{MSB}(v) = 0$  and  $\text{MSB}(L(v)) = 0$ . In that case:

$$L(v \ll 1) = L(v) \ll 1 . \quad (8.2)$$

If  $\text{MSB}(a) = 0$ , then  $a \ll 1$  and  $a \lll 1$  are equivalent. Because of the particular choice of the Boolean function in ESSENCE, this attack does not work on ESSENCE itself, but on a variant of the ESSENCE block cipher.

The idea of rotational inputs was used as well to find fixed points and key collisions for the permutation  $\mathcal{P}$  used in the hash function Shabal [KMT09]. Rotational pairs were traced

through bitwise logical operations and rotations, however rotations were chosen in such a way that there was no loss in probability for the additions operations in  $\mathcal{U}(x) = x + x \lll 1$  and  $\mathcal{V}(x) = x + x \lll 2$ .

In [KN10], the concept of rotational cryptanalysis is formally explained, and applied to reduced rounds of the Threefish block cipher, the core of the Skein hash function. As in [MT09], a related-key chosen-plaintext attack is constructed, where both keys and plaintexts are rotated. Cryptanalysis results are presented on 39, 42 and 43 full rounds of Threefish-256, -512 and -1024 respectively, where the attack complexity is estimated to be slightly less than generic.

A mention of the completeness of ARX is given in [Ber08]. A formal proof can be found in [KN10]. Because ARX is shown to be functionally complete, it is possible to use addition, rotation and XOR to implement any circuit (including all block ciphers, hash functions and MACs), but not necessarily in the most efficient way. Although this result indicates the soundness of ARX constructions, a new toolkit is required for the cryptanalysis and design of these primitives. Such a toolkit is presented in the next section.

## 8.2 Toolkit for Cryptanalysis of ARX

We now describe a software toolkit, designed to automate the differential cryptanalysis of cryptographic constructions based on the operations of addition, rotation and XOR (ARX). The source code for this toolkit will be made publicly available on-line. It contains several programs, each of which calculates the probability that XOR or additive differences propagate through a certain type operation. Supported type of operations include XOR, modular addition and multiplications by a constant.

A subset of the cases where the toolkit can be used, were studied in literature. In [LWD04], the differential probability  $\text{xdp}^+$  of addition modulo  $2^n$ , where differences are expressed using XOR, is calculated using matrix multiplications. In the same paper, a similar result was obtained for the differential probability  $\text{adp}^\oplus$  of xor, when differences are expressed using addition modulo  $2^n$ . These computations have a time complexity that is linear in the word size  $n$ .

The algorithms implemented in the ARX toolkit are also based on matrix multiplications. The matrices are generated in a way that they are correct by construction, and automatic techniques are used to minimize their size. The main advantage of the toolkit is that it is general, and can thus easily be extended to a larger number of cases. It can be used to calculate  $\text{xdp}^+$  and  $\text{adp}^\oplus$ , as well as  $\text{xdp}^+(\alpha, \beta, \dots \rightarrow \gamma)$ , which is the calculation of  $\text{xdp}^+$  for more than two inputs, and the differential probability  $\text{xdp}^{\times C}$  of multiplication by a constant  $C$  where differences are expressed by xor.

The tool can also efficiently count the number of output differences for each of these operations. For example, this problem occurs in the cryptanalysis of Threefish-512 [AccM<sup>+</sup>09], where an exponential-in- $n$  time algorithm is proposed. Using the toolkit, however, this can be solved in linear time in  $n$ .

Additionally, the toolkit provides a general algorithm to efficiently list the output differences with the highest probability, assuming input differences and the operation are given.



## Chapter 9

# Analysis of AES-based Hash Functions

The submission of many AES-based hash function to the NIST SHA-3 competition [Nat07] has initiated new developments in the analysis of AES [Nat01] as a building block in hash functions. Since all inputs to hash functions are known, different types of attacks than on AES as a block cipher have emerged. Additionally, the message in a hash function can be chosen freely, which usually gives an attacker more freedom than in a block cipher, where the key is unknown and/or fixed. Two different directions of analyzing AES based hash functions have been published recently. Both attacks need specific (truncated) differential paths to work with. The recent attack on the AES block cipher applied to AES in hashing mode [BKN09] uses local collisions to construct high-probability differential paths, whereas the rebound attack [MRST09] uses the available degrees of freedom to find pairs for the (multiple) expansive parts of a path more efficiently.

In [KBN09] the triangulation algorithm, a new tool for the collision search in hash functions has been proposed and applied to Rijndael in hashing mode. Further, in [BKN09] an attack to construct  $q$ -pseudo collisions for AES-256 in Davies-Meyer hashing mode has been published. In this work, differences of the key-schedule are used to construct high-probability differential paths using local collisions in the state update. Due to the degrees of freedom of the 256-bit key-schedule, many local collisions and thus, a distinguishing attack on the full compression function is possible.

Further, the additional degrees of freedom available in a hash function have been extensively used by the rebound attack, especially in the analysis of many AES based hash functions. The rebound attack [MRST09] has been published by Mendel et al. in the analysis of the AES-based hash functions Whirlpool [BR00] and Grøstl [GKM<sup>+</sup>08]. It can be applied to both block cipher based and permutation based constructions. The idea of the rebound attack is to divide an attack into two phases, an inbound and outbound phase. The inbound phase is an efficient meet-in-the-middle phase, which exploits the available degrees of freedom in the middle of a (truncated) differential path to guarantee that the expensive part of the path holds. In the (mainly) probabilistic outbound phase the solutions of the inbound phase are computed backwards and forwards to obtain an attack on the hash or compression function.

The rebound idea has also been applied to the AES-based SHA-3 candidates Twister [MRS09], Cheetah [Wu09] and LANE [WFW09]. Further, two different lines of improvements have

been made to the rebound attack in the last year. First, the efficiency of the inbound phase has been improved using different techniques in [MPRS09], and the inbound phase has also been extended by one round using SuperBox techniques in the improved analysis of Whirlpool [LMR<sup>+</sup>09, Appendix] and later applied to Grøstl in [MRST10] and [GP10]. Second, multiple inbound phases can be used if more degrees of freedom from a key-schedule or due to sparse truncated differential paths are available. This more powerful improvement allows to double the number of attacked rounds and has been used in the analysis of the compression functions Whirlpool [LMR<sup>+</sup>09] and LANE [MNPN<sup>+</sup>09].

The improved analysis of AES based hash functions can be interpreted as a weakness of AES based designs. However, one can also argue that the simple structure of AES simply accelerates understanding, and thereby the development of attacks. In that case, more results can be expected later on other types of hash functions.

## Chapter 10

# Meet-in-the-middle preimage attacks on hash functions

Even though they may be traced back to Lai and Massey [LM92], meet-in-the-middle attacks on the preimage resistance of hash functions recently received renewed attention. This is both in the context of MD2 by Knudsen et al. [KM05,KMMT10,Mul04b], attacks on round-1 SHA-3 candidates by Khovratovic et al. [KNW09], as well as a series of results on members of the MD4 family of hash functions by Aoki, Sasaki and others [AGM<sup>+</sup>09, AS09a, AS09b, SA08, SA09, GLRW10], and Tiger [IS09, GLRW10, Men09]. In the following, we describe them in a way to fit into the meet-in-the-middle (MITM) framework of Aoki and Sasaki. Other interesting approaches to preimage attacks appeared e.g. in [Bih08, DR08, Leu08, MPR08a, MPR<sup>+</sup>08b, YWZW05, Rec10].

### 10.1 The basic approach

The general idea of the preimage attack can be explained as follows. Let us consider a block-cipher based hash function in Davies-Meyer mode. Further let us define a splitting point at a particular round during the computation, and a matching point at another round.

1. Split the compression function into two chunks, where the values in one chunk do not depend on some message word  $W_p$  and the values in the other chunk do not depend on another message word  $W_q$  ( $p \neq q$ ). We follow the convention and call such words neutral with respect to the first and second chunk, respectively.
2. Fix all other values except for  $W_p, W_q$  to random values and assign random values to the chaining registers at the splitting point.
3. Start the computation both backward and forward from the splitting point to form two lists  $L_p, L_q$  indexed by all possible values of  $W_p$  and  $W_q$ , containing the computed values of the chaining registers at the matching point.
4. Compare two lists to find partial matches (match for one or a few registers instead of the full chaining) at the matching point.
5. Repeat the above three steps with different initial configurations (values for splitting point and other message words) until a full match is found.

With the work effort of  $2^l$  compression evaluations (let the space for both  $W_p$  and  $W_q$  be  $2^l$ ), we obtain two lists, each one containing  $2^l$  values of the register to match. When we consider all of the  $2^{2l}$  possible pairs, we expect to get around  $2^l$  matches (assume we match  $l$  bits at the matching point). This means that after  $2^l$  computations we get  $2^l$  matches on one register, effectively reducing the search space by  $2^l$ . Leaving all the other registers to chance allows us to find a complete match and thus a pseudo-preimage in  $2^{n-l}$  computations if the chaining is of  $n$  bits. We repeat the pseudo-preimage finding  $2^{l/2}$  times, which costs  $2^{n-l/2}$ , and then find a message which links to one of the  $2^{l/2}$  pseudo-preimages, this costs  $2^{n-l/2}$ . So the overall complexity for finding one preimage is  $2^{n-l/2+1}$ , with memory requirement of order  $2^l$ .

## 10.2 Refinements and improvements

A number of refinements and improvements have been proposed recently that allowed attacks to cover more and more rounds, and sometimes also led to improved attack complexities. We first discuss a selection of them with a focus on compression function attack techniques.

- **Initial Structure.** An Initial Structure (introduced in [SA09]) can swap the order of some message words near the splitting point, so that the length of the two chunks can be extended. Assume that originally both chunks  $p$  and  $q$  contain both neutral words  $W_p$  and  $W_q$ . After the initial structure, essentially the  $W_p$  and  $W_q$  near the splitting point are swapped, so that chunk  $p$  is independent from  $W_q$  and chunk  $q$  is independent from  $W_p$ . A probabilistic generalization is described and used in [GLRW10].
- **Partial Matching and Partial Fixing.** Partial matching can extend the attack for a few additional rounds. There are  $W_p$  and  $W_q$  near the matching point, which appear in other chunks and destroy the independence. However we can still compute few bits at the matching point, independently for both chunks, assuming no knowledge of  $W_p$  and  $W_q$  near the matching point. Partial fixing will fix part of the  $W_p$  and  $W_q$  so that we can still make use of those fixed bits, and extend the attack for a few more rounds. Sometimes,  $W_p$  and  $W_q$  near the matching point behave in such a way that we can express the matching point as  $f(W_q) + \sigma(W_p)$  from chunk  $q$ , and  $g(W_p) + \mu(W_q)$  from chunk  $p$ , for some functions  $f, \sigma, g, \mu$  depending on the underlying hash function. So we can compute  $f(W_q) - \mu(W_q)$  from chunk  $q$  and  $g(W_p) - \sigma(W_p)$  from chunk  $p$  independently, and then find matches. This is called indirect partial matching, see [AGM<sup>+</sup>09].

Note that a successful match gives only a pseudo-preimage, as the initial value is determined during the attack. However, various techniques are known to convert pseudo-preimages to a preimage.

- **Generic.** A generic technique described in [MvOV96, Fact 9.99]. One can compute many pseudo-preimages, and then find a message linking the IV to one of the input chaining values of the pseudo-preimages.
- **Dedicated.** A number of tree [DR08, Leu08, MR07] and graph [DR08] based approaches are known that allow to exploit properties of compression function attacks for a more efficient conversion from pseudo-preimages to preimages than the generic approach. Until recently the powerful pseudo-preimages techniques like partial matching/fixing

and initial structure were believed to be incompatible with those conversion methods. In [GLRW10], an approach to combine both is described, leading to more efficient conversions while at the same time allowing to take advantage of compression function attack techniques as described above.

### 10.3 Discussion of results

The method is well suited to cover many rounds in preimage attacks. For the SHA-2 family and the Tiger hash function, this type of attack is able to cover more rounds than any other attack (up to 43/46 rounds for SHA-2, and full-round Tiger [AGM<sup>+</sup>09, GLRW10]). On the other hand, it was not demonstrated yet that this approach can yield practical attacks as time complexity is often close to that of brute force search. The result on the MD4 compression function in time  $2^{72}$  [GLRW10] may be seen as an exception to this. Also memory-requirements are often non-negligible. As an example, the result on full MD5 [SA09] with 128-bit output needs time  $2^{123.4}$  and about  $2^{45}$  memory, while the result on full Tiger [GLRW10] with 192-bit output needs time  $2^{188.2}$  and about  $2^8$  memory.



# Bibliography

- [AccM<sup>+</sup>09] Jean-Philippe Aumasson, Çağdas Çalik, Willi Meier, Onur Özen, Raphael C.-W. Phan, and Kerem Varıcı. Improved Cryptanalysis of Skein. In Mitsuru Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 542–559. Springer, 2009.
- [ADH<sup>+</sup>] Jean-Philippe Aumasson, Itai Dinur, Luca Henzen, Willi Meier, and Adi Shamir. Efficient FPGA implementation of high-dimensional cube testers on the stream cipher Grain-128. SHARCS 2009, Online at <http://eprint.iacr.org/2009/218>.
- [ADMS09] Jean-Philippe Aumasson, Itai Dinur, Willi Meier, and Adi Shamir. Cube testers and key recovery attacks on reduced-round MD6 and Trivium. In Orr Dunkelman, editor, *FSE*, volume 5665 of *Lecture Notes in Computer Science*, pages 1–22. Springer, 2009.
- [AGM<sup>+</sup>09] Kazumaro Aoki, Jian Guo, Krysitan Matusiewicz, Yu Sasaki, and Lei Wang. Preimages for Step-Reduced SHA-2. In Matsui [Mat09], pages 578–597.
- [AHMP08] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and Raphael C.-W. Phan. SHA-3 proposal BLAKE. Submission to the NIST SHA-3 Competition (Round 2), 2008.
- [AKK<sup>+</sup>03] Noga Alon, Tali Kaufman, Michael Krivelevich, Simon Litsyn, and Dana Ron. Testing low-degree polynomials over GF(2). In Sanjeev Arora, Klaus Jansen, José D. P. Rolim, and Amit Sahai, editors, *RANDOM-APPROX*, volume 2764 of *Lecture Notes in Computer Science*, pages 188–199. Springer, 2003.
- [AM] Jean-Philippe Aumasson and Willi Meier. Zero-sum distinguishers for reduced Keccak- $f$  and for the core functions of Luffa and Hamsi. Online at <http://www.131002.net/papers.html>.
- [AM07] Jean-Philippe Aumasson and Willi Meier. Analysis of multivariate hash functions. In Kil-Hyun Nam and Gwangsoo Rhee, editors, *ICISC*, volume 4817 of *LNCS*, pages 309–323. Springer, 2007.
- [AMS<sup>+</sup>ar] Z. Ahmadian, J. Mohajeri, M. Salmasizadeh, R. Hakala, and K. Nyberg. A practical distinguisher for the Shannon cipher. *Journal of Systems and Software*, to appear.
- [AS09a] Kazumaro Aoki and Yu Sasaki. Meet-in-the-middle preimage attacks against reduced SHA-0 and SHA-1. In Shai Halevi, editor, *Advances in Cryptology*

- *CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 70–89, Berlin, Heidelberg, New York, 2009. Springer-Verlag.
- [AS09b] Kazumaro Aoki and Yu Sasaki. Preimage attacks on one-block MD4, 63-step MD5 and more. In Roberto Avanzi, Liam Keliher, and Francesco Sica, editors, *Selected Areas in Cryptography - SAC 2008*, volume 5381 of *Lecture Notes in Computer Science*, pages 103–119, Sackville, Canada, 2009. Springer-Verlag.
- [Bab95] Steve Babbage. A space/time tradeoff in exhaustive search attacks on stream ciphers. *European Convention on Security and Detection*, 408, 1995.
- [BCCM<sup>+</sup>08] Emmanuel Bresson, Anne Canteaut, Benoît Chevallier-Mames, Christophe Clavier, Thomas Fuhr, Aline Gouget, Thomas Icart, Jean-François Misarsky, Maria Naya-Plasencia, Pascal Paillier, Thomas Pornin, Jean-René Reinhard, Céline Thuillet, and Marion Videau. Shabal, a Submission to NIST’s Cryptographic Hash Algorithm Competition. Submission to the NIST SHA-3 Competition (Round 2), 2008.
- [BCD03] Alex Biryukov, Christophe De Cannière, and Gustaf Dellkrantz. Cryptanalysis of SAFER++. In *Crypto 2003*, volume 2729 of *LNCS*, pages 195–211. Springer-Verlag, 2003.
- [BCQ04] A. Biryukov, C. De Cannière, and M. Quisquater. On multiple linear approximations. In M.K. Franklin, editor, *CRYPTO’04*, volume 3152 of *Lecture Notes in Computer Science*, pages 1–22. Springer, 2004.
- [BDK02] E. Biham, O. Dunkelman, and N. Keller. Enhancing differential-linear cryptanalysis. In Y. Zheng, editor, *ASIACRYPT’02*, volume 2501 of *Lecture Notes in Computer Science*, pages 254–266. Springer, 2002.
- [BDK<sup>+</sup>10] Alex Biryukov, Orr Dunkelman, Nathan Keller, Dmitry Khovratovich, and Adi Shamir. Key recovery attacks of practical complexity on AES-256 variants with up to 10 rounds. In *EUROCRYPT’10*, volume 6110 of *LNCS*, pages 299–319. Springer-Verlag, 2010.
- [Ber05] Dan J. Bernstein. Understanding brute force. In *Workshop on Symmetric Key Encryption (SKEW 2005)*, 2005.
- [Ber08] Daniel J. Bernstein. The Salsa20 Family of Stream Ciphers. In Matthew J. B. Robshaw and Olivier Billet, editors, *The eSTREAM Finalists*, volume 4986 of *Lecture Notes in Computer Science*, pages 84–97. Springer, 2008.
- [Ber09] Daniel J. Bernstein. CubeHash specification (2.B.1). Submission to the NIST SHA-3 Competition (Round 2), 2009.
- [BGM06] C. Berbain, H. Gilbert, and A. Maximov. Cryptanalysis of grain. In M.J.B. Robshaw, editor, *FSE’06*, volume 4047 of *Lecture Notes in Computer Science*, pages 15–29. Springer, 2006.
- [Bih94] Eli Biham. New Types of Cryptanalytic Attacks Using Related Keys. *J. Cryptology*, 7(4):229–246, 1994.

- [Bih02] Eli Biham. How to decrypt or even substitute des-encrypted messages in 228 steps. *Information Processing Letters*, 84(3):117–124, 2002.
- [Bih08] Eli Biham. New Techniques for Cryptanalysis of Hash Functions and Improved Attacks on Snefru. In Nyberg [Nyb08], pages 444–461.
- [BJV04] T. Baignères, P. Junod, and S. Vaudenay. How far can we go beyond linear cryptanalysis? In P.L. Lee, editor, *ASIACRYPT'04*, volume 3329 of *Lecture Notes in Computer Science*, pages 432–450. Springer, 2004.
- [BK03] Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: Rka-prps, rka-prfs, and applications. In *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 491–506. Springer-Verlag, 2003.
- [BK09] Alex Biryukov and Dmitry Khovratovich. Related-key cryptanalysis of the full AES-192 and AES-256. In *ASIACRYPT'09*, volume 5912 of *LNCS*, pages 1–18. Springer-Verlag, 2009.
- [BKN09] Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolic. Distinguisher and Related-Key Attack on the Full AES-256. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *LNCS*, pages 231–249. Springer, 2009.
- [BLR90] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. In *STOC*, pages 73–83. ACM, 1990.
- [BMS06] Alex Biryukov, Sourav Mukhopadhyay, and Palash Sarkar. Improved time-memory trade-offs with multiple data. In *Selected Areas in Cryptography (SAC 2005)*, volume 3897 of *LNCS*, pages 110–127. Springer-Verlag, 2006.
- [BR00] Paulo S. L. M. Barreto and Vincent Rijmen. The WHIRLPOOL Hashing Function. Submitted to NESSIE, September 2000, revised May 2003, 2000. Available online: <http://paginas.terra.com.br/informatica/paulobarreto/WhirlpoolPage.html>.
- [BRP07] Olivier Billet, Matthew J. B. Robshaw, and Thomas Peyrin. On building hash functions from multivariate quadratic equations. In Josef Pieprzyk, Hossein Ghodosi, and Ed Dawson, editors, *ACISP*, volume 4586 of *LNCS*, pages 82–95. Springer, 2007.
- [BS91] Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. *J. Cryptology*, 4(1):3–72, 1991.
- [BS00] Alex Biryukov and Adi Shamir. Cryptanalytic time/memory/data tradeoffs for stream ciphers. In *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 1–13. Springer-Verlag, 2000.
- [Cho10] J.Y. Cho. Linear cryptanalysis of reduced-round PRESENT. In J. Pieprzyk, editor, *CT-RSA '10*, volume 5985 of *Lecture Notes in Computer Science*, pages 302–317. Springer, 2010.
- [CHar] J.Y. Cho and M. Hermelin. Improved linear cryptanalysis of SOSEMANUK. In *ICISC'09*, *Lecture Notes in Computer Science*. Springer, to appear.

- [CP07] J.Y. Cho and J. Pieprzyk. An improved distinguisher for dragon. eSTREAM ECRYPT Stream Cipher Project, report 2007/002, 2007.
- [CP08] Christophe De Cannière and Bart Preneel. Trivium. In *New Stream Cipher Designs*, volume 4986 of *LNCS*, pages 84–97. Springer, 2008.
- [CSQ07] B. Collard, F.-X. Standaert, and J.-J. Quisquater. Improving the time complexity of Matsui’s linear cryptanalysis. In K.-H. Nam and G. Rhee, editors, *ICISC’07*, volume 4817 of *Lecture Notes in Computer Science*, pages 77–88. Springer, 2007.
- [CSQ08] B. Collard, F.-X. Standaert, and J.-J. Quisquater. Experiments on the multiple linear cryptanalysis of reduced round serpent. In K. Nyberg, editor, *FSE’08*, volume 5086 of *Lecture Notes in Computer Science*, pages 382–397. Springer, 2008.
- [DGV94] J. Daemen, R. Govaerts, and J. Vandewalle. Resynchronization weaknesses in synchronous stream ciphers. In *Eurocrypt 1993*, volume 765 of *LNCS*, pages 159–167. Springer-Verlag, 1994.
- [DK08] O. Dunkelman and N. Keller. A new attack on the lex stream cipher. In *Asiacrypt 2008*, volume 5350, pages 539–556. Springer-Verlag, 2008.
- [DKS] Orr Dunkelman, Nathan Keller, and Adi Shamir. A practical-time attack on the A5/3 cryptosystem used in third generation GSM telephony. Technical report, <http://eprint.iacr.org/2010/013.pdf>, YEAR = 2010,.
- [DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002.
- [DR08] Christophe De Cannière and Christian Rechberger. Preimages for Reduced SHA-0 and SHA-1. In Wagner [Wag08], pages 179–202.
- [DS09] Itai Dinur and Adi Shamir. Cube attacks on tweakable black box polynomials. In Antoine Joux, editor, *EUROCRYPT*, volume 5479 of *Lecture Notes in Computer Science*, pages 278–299. Springer, 2009.
- [Dun09] Orr Dunkelman, editor. *Fast Software Encryption, 16th International Workshop, FSE 2009, Leuven, Belgium, February 22-25, 2009, Revised Selected Papers*, volume 5665 of *LNCS*. Springer, 2009.
- [EHJ] A note on distinguishing attacks. IEEE 2007.
- [EJT07] Håkan Englund, Thomas Johansson, and Meltem Sönmez Turan. A framework for chosen IV statistical analysis of stream ciphers. In K. Srinathan, C. Pandu Rangan, and Moti Yung, editors, *INDOCRYPT*, volume 4859 of *LNCS*, pages 268–281. Springer, 2007.
- [EM05] H. Englund and A. Maximov. Attack the Dragon. In S. Maitra, C.E. Veni Madhavan, and R. Venkatesan, editors, *INDOCRYPT’05*, volume 3797 of *Lecture Notes in Computer Science*, pages 130–142. Springer, 2005.

- [Fil02] Eric Filiol. A new statistical testing for symmetric ciphers and hash functions. In Robert H. Deng, Sihan Qing, Feng Bao, and Jianying Zhou, editors, *ICICS*, volume 2513 of *LNCS*, pages 342–353. Springer, 2002.
- [FIP99] PUB FIPS. 46-3: Data Encryption Standard (DES). *National Institute for Standards and Technology, Gaithersburg, MD, USA*, 1999.
- [FKM08] Simon Fischer, Shahram Khazaei, and Willi Meier. Chosen IV statistical analysis for key recovery attacks on stream ciphers. In Serge Vaudenay, editor, *AFRICACRYPT*, volume 5023 of *LNCS*, pages 236–245. Springer, 2008.
- [FLS<sup>+</sup>09] Niels Ferguson, Stefan Lucks, Bruce Schneier, Doug Whiting, Mihir Bellare, Tadayoshi Kohno, Jon Callas, and Jesse Walker. The Skein Hash Function Family. Submission to the NIST SHA-3 Competition (Round 2), 2009.
- [FM00] S. Fluhrer and D. McGrew. Statistical analysis of the alleged rc4 keystream. In *Fast Software Encryption 2000*, volume 1978. Springer-Verlag, 2000.
- [GKK<sup>+</sup>09] Danilo Gligoroski, Vlastimil Klima, Svein Johan Knapskog, Mohamed El-Hadedy, Jørn Amundsen, and Stig Frode Mjølsnes. Cryptographic Hash Function BLUE MIDNIGHT WISH. Submission to the NIST SHA-3 Competition (Round 2), 2009.
- [GKM<sup>+</sup>08] Praveen Gauravaram, Lars R. Knudsen, Krystian Matusiewicz, Florian Mendel, Christian Rechberger, Martin Schl affer, and S oren S. Thomsen. Grstl – a SHA-3 candidate. Submission to NIST, 2008. Available online: <http://groestl.info>.
- [GLRW10] Jian Guo, San Ling, Christian Rechberger, and Huaxiong Wang. Advanced Meet-in-the-Middle Preimage Attacks: First Results on Full Tiger, and Improved Results on MD4 and SHA-2. Cryptology ePrint Archive, Report 2010/016, 2010. <http://eprint.iacr.org/>.
- [Gol97] Jovan Dj. Golić. Cryptanalysis of alleged a5 stream cipher. In *EUROCRYPT*, volume 1233 of *LNCS*. Springer-Verlag, 1997.
- [GP10] Henri Gilbert and Thomas Peyrin. Super-Sbox Cryptanalysis: Improved Attacks for AES-like permutations. FSE, 2010. to appear.
- [HCN08] M. Hermelin, J.Y. Cho, and K. Nyberg. Multidimensional linear cryptanalysis of reduced round Serpent. In Y. Mu, W. Susilo, and J. Seberry, editors, *ACISP’08*, volume 5107 of *Lecture Notes in Computer Science*, pages 203–215. Springer, 2008.
- [HCN09a] M. Hermelin, J.Y. Cho, and K. Nyberg. Multidimensional extension of Matsui’s Algorithm 2. In O. Dunkelman, editor, *FSE’09*, volume 5665 of *Lecture Notes in Computer Science*, pages 209–227. Springer, 2009.
- [HCN09b] M. Hermelin, J.Y. Cho, and K. Nyberg. Statistical tests for key recovery using multidimensional extension of Matsui’s Algorithm 1. EUROCRYPT’09 POSTERSESSION, 2009.

- [Hel80] Martin Hellman. A cryptanalytic time-memory trade-off. *IEEE Transactions on Information Theory*, 26:401–406, 1980.
- [HJ06] M. Hell and T. Johansson. Cryptanalysis of achterbahn-version 2. eSTREAM ECRYPT Stream Cipher Project, report 2006/042, 2006.
- [HN08] M. Hermelin and K. Nyberg. Multidimensional linear distinguishing attacks and boolean functions. In *BFCA '08*, 2008. Proceedings available on-line <http://www.liafa.jussieu.fr/bfca>.
- [HN09] R. Hakala and K. Nyberg. A multidimensional linear distinguishing attack on the Shannon cipher. *International Journal of Applied Cryptography*, 1(3):161–168, 2009.
- [HN10] M. Hermelin and K. Nyberg. Dependent linear approximations - the algorithm of Biryukov and others revisited. In J. Pieprzyk, editor, *CT-RSA '10*, volume 5985 of *Lecture Notes in Computer Science*, pages 318–333. Springer, 2010.
- [HS05] Jin Hong and Palash Sarkar. New applications of time memory data tradeoffs. In *ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 353–372. Springer-Verlag, 2005.
- [IOKM06] T. Isobe, T. Ohigashi, H. Kuwakado, and M. Morii. How to break py and pyp by a chosen-iv attack. eSTREAM, ECRYPT Stream Cipher Project, report 2006/060, 2006.
- [IS09] Takanori Isobe and Kyoji Shibutani. Preimage attacks on reduced Tiger and SHA-2. In Dunkelman [Dun09], pages 139–155.
- [JM03] T. Johansson and A. Maximov. A linear distinguishing attack on Scream. In *IEEE International Symposium on Information Theory, ISIT 2003*, page 164, 2003.
- [JR07] A. Joux and J.-R. Reinhard. Overtaking vest. eSTREAM, ECRYPT Stream Cipher Project, report 2007/021, 2007.
- [JRSN09] Michael J. Jacobson Jr., Vincent Rijmen, and Reihaneh Safavi-Naini, editors. *Selected Areas in Cryptography, 16th Annual International Workshop, SAC 2009, Calgary, Alberta, Canada, August 13-14, 2009, Revised Selected Papers*, volume 5867 of *LNCS*. Springer, 2009.
- [Jun01] P. Junod. On the complexity of Matsui's attack. In S. Vaudenay and A.M. Youssef, editors, *SAC'01*, volume 2259 of *Lecture Notes in Computer Science*, pages 199–211. Springer, 2001.
- [JV03] P. Junod and S. Vaudenay. Optimal key ranking procedures in a statistical cryptanalysis. In T. Johansson, editor, *FSE'03*, volume 2887 of *Lecture Notes in Computer Science*, pages 235–246. Springer, 2003.
- [KBN09] Dmitry Khovratovich, Alex Biryukov, and Ivica Nikolic. Speeding up Collision Search for Byte-Oriented Hash Functions. In Marc Fischlin, editor, *CT-RSA*, volume 5473 of *LNCS*, pages 164–181. Springer, 2009.

- [Kel03] L. Keliher. *Linear cryptanalysis of substitution-permutation networks*. PhD thesis, School of Computing, Queen's University, Kingston, Ontario, Canada, 2003.
- [KM01] L.R. Knudsen and J.E. Mathiassen. A chosen-plaintext linear attack on DES. In B. Schneier, editor, *FSE'00*, volume 1978 of *Lecture Notes in Computer Science*, pages 262–272. Springer, 2001.
- [KM05] Lars R. Knudsen and John Erik Mathiassen. Preimage and Collision Attacks on MD2. In Henri Gilbert and Helena Handschuh, editors, *FSE*, volume 3557 of *LNCS*, pages 255–267. Springer, 2005.
- [KM08] Shahram Khazaei and Willi Meier. New directions in cryptanalysis of self-synchronizing stream ciphers. In Dipanwita Roy Chowdhury, Vincent Rijmen, and Abhijit Das, editors, *INDOCRYPT*, volume 5365 of *LNCS*, pages 15–26. Springer, 2008.
- [KMMT10] Lars R. Knudsen, John Erik Mathiassen, Frédéric Muller, and Søren S. Thomsen. Cryptanalysis of MD2. *J. Cryptology*, 23(1):72–90, 2010.
- [KMT09] Lars R. Knudsen, Krystian Matusiewicz, and Søren S. Thomsen. Observations on the Shabal keyed permutation. Official Comment to the NIST SHA-3 Competition, 2009.
- [KN10] Dmitry Khovratovich and Ivica Nikolic. Rotational Cryptanalysis of ARX. *Fast Software Encryption, 11th International Workshop, FSE 2010, Seoul, Korea*, 2010.
- [Knu94] Lars R. Knudsen. Truncated and higher order differentials. In Bart Preneel, editor, *FSE*, volume 1008 of *LNCS*, pages 196–211. Springer, 1994.
- [KNW09] Dmitry Khovratovich, Ivica Nikolic, and Ralf-Philipp Weinmann. Meet-in-the-Middle Attacks on SHA-3 Candidates. In Dunkelman [Dun09], pages 228–245.
- [KR94] B.S. Jr Kaliski. and M.J.B. Robshaw. Linear cryptanalysis using multiple approximations. In Y. Desmedt, editor, *CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, pages 26–39. Springer, 1994.
- [KR96] L.R. Knudsen and M.J.B. Robshaw. Non-linear approximations in linear cryptanalysis. In U.M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 224–236. Springer, 1996.
- [KR04] Tali Kaufman and Dana Ron. Testing polynomials over general fields. In *FOCS*, pages 413–422. IEEE Computer Society, 2004.
- [KS05] Alexander Klimov and Adi Shamir. New applications of T-functions in block ciphers and hash functions. In Henri Gilbert and Helena Handschuh, editors, *FSE*, volume 3557 of *Lecture Notes in Computer Science*, pages 18–31. Springer, 2005.

- [KS08] Tali Kaufman and Madhu Sudan. Algebraic property testing: the role of invariance. In Richard E. Ladner and Cynthia Dwork, editors, *STOC*, pages 403–412. ACM, 2008.
- [KSW97] John Kelsey, Bruce Schneier, and David Wagner. Related-key cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA. In Yongfei Han, Tatsuaki Okamoto, and Sihan Qing, editors, *ICICS*, volume 1334 of *Lecture Notes in Computer Science*, pages 233–246. Springer, 1997.
- [KYS] Bonwook Koo, Yongjin Yeom, and Junghwan Song. Related-key boomerang attack on block cipher SQUARE. Technical report, <http://eprint.iacr.org/2010/073.pdf>, YEAR = 2010,.
- [LBF09] Gaëtan Leurent, Charles Bouillaguet, and Pierre-Alain Fouque. SIMD Is a Message Digest. Submission to the NIST SHA-3 Competition (Round 2), 2009.
- [Leu08] Gaëtan Leurent. MD4 is not one-way. In Nyberg [Nyb08], pages 412–428.
- [LH94] S.K. Langford and M.E. Hellman. Differential-linear cryptanalysis. In Y. Desmedt, editor, *CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, pages 17–25. Springer, 1994.
- [LM92] Xuejia Lai and James L. Massey. Hash Function Based on Block Ciphers. In *EUROCRYPT*, pages 55–70, 1992.
- [LMR<sup>+</sup>09] Mario Lamberger, Florian Mendel, Christian Rechberger, Vincent Rijmen, and Martin Schl affer. Rebound Distinguishers: Results on the Full Whirlpool Compression Function. In Matsui [Mat09], pages 126–143.
- [Luc01] Stefan Lucks. The saturation attack - a bait for Twofish. In Mitsuru Matsui, editor, *FSE*, volume 2355 of *LNCS*, pages 1–15. Springer, 2001.
- [LWD04] Helger Lipmaa, Johan Wall en, and Philippe Dumas. On the Additive Differential Probability of Exclusive-Or. In Bimal K. Roy and Willi Meier, editors, *FSE*, volume 3017 of *Lecture Notes in Computer Science*, pages 317–331. Springer, 2004.
- [Mat94a] M. Matsui. The first experimental cryptanalysis of the Data Encryption Standard. In Y. Desmedt, editor, *CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, pages 1–11. Springer, 1994.
- [Mat94b] M. Matsui. Linear cryptoanalysis method for DES cipher. In T. Helleseth, editor, *EUROCRYPT'93*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer, 1994.
- [Mat96] Mitsuru Matsui. New structure of block ciphers with provable security against differential and linear cryptanalysis. In *FSE 1996*, volume 1039 of *LNCS*, pages 205–218. Springer-Verlag, 1996.
- [Mat09] Mitsuru Matsui, editor. *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, volume 5912 of *LNCS*. Springer, 2009.

- [Max05] A. Maximov. Two linear distinguishing attacks on vmpc and rc4a and weakness of the rc4 family of stream ciphers. In *Fast Software Encryption 2005*, volume 3557 of *LNCS*, pages 342–358. Springer-Verlag, 2005.
- [MB07] Alexander Maximov and Alex Biryukov. Two trivial attacks on Trivium. In Carlisle M. Adams, Ali Miri, and Michael J. Wiener, editors, *Selected Areas in Cryptography*, volume 4876 of *LNCS*, pages 36–55. Springer, 2007.
- [MCP07] Cameron McDonald, Chris Charnes, and Josef Pieprzyk. Attacking Bivium with MiniSat. eSTREAM, ECRYPT Stream Cipher Project, Report 2007/040, 2007.
- [Men09] Florian Mendel. Two Passes of Tiger Are Not One-Way. In Bart Preneel, editor, *AFRICACRYPT*, volume 5580 of *Lecture Notes in Computer Science*, pages 29–40. Springer, 2009.
- [MJ07] A. Maximov and T. Johansson. A linear distinguishing attack on Scream. *IEEE Transactions on Information Theory*, 53(9):3127–3144, 2007.
- [MNPN<sup>+</sup>09] Krystian Matusiewicz, María Naya-Plasencia, Ivica Nikolic, Yu Sasaki, and Martin Schläffer. Rebound Attack on the Full Lane Compression Function. In Matsui [Mat09], pages 106–125.
- [MPR08a] Florian Mendel, Norbert Pramstaller, and Christian Rechberger. A (Second) Preimage Attack on the GOST Hash Function. In Nyberg [Nyb08], pages 224–234.
- [MPR<sup>+</sup>08b] Florian Mendel, Norbert Pramstaller, Christian Rechberger, Marcin Kontak, and Janusz Szmidt. Cryptanalysis of the GOST Hash Function. In Wagner [Wag08], pages 162–178.
- [MPRS09] Florian Mendel, Thomas Peyrin, Christian Rechberger, and Martin Schläffer. Improved Cryptanalysis of the Reduced Grøstl Compression Function, ECHO Permutation and AES Block Cipher. In Jr. et al. [JRSN09], pages 16–35.
- [MR07] Florian Mendel and Vincent Rijmen. Weaknesses in the HAS-V compression function. In Kil-Hyun Nam and Gwangsoo Rhee, editors, *Information Security and Cryptology ICISC 2007*, volume 4817 of *Lecture Notes in Computer Science*, pages 335–345, Berlin, Heidelberg, New York, 2007. Springer-Verlag.
- [MRS09] Florian Mendel, Christian Rechberger, and Martin Schläffer. Cryptanalysis of Twister. In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *ACNS*, volume 5536 of *LNCS*, pages 342–353, 2009.
- [MRST09] Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grøstl. In Dunkelman [Dun09], pages 260–276.
- [MRST10] Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. Rebound Attacks on the Reduced Grøstl Hash Function. In Josef Pieprzyk, editor, *CT-RSA*, volume 5985 of *LNCS*, pages 350–365. Springer, 2010.

- [MT09] Nicky Mouha and Meltem Sönmez Turan. A related-key attack on the ESSENCE block cipher. *ECRYPT II First Hash Function Retreat, 2010, Graz, Austria (Rump session talk)*, 2009.
- [Mul04a] F. Muller. Differential attacks and stream ciphers. State of the Art in Stream Ciphers. eSTREAM, ECRYPT Network of Excellence in Cryptology, 2004.
- [Mul04b] Frédéric Muller. The MD2 Hash Function Is Not One-Way. In Pil Joong Lee, editor, *ASIACRYPT*, volume 3329 of *LNCSS*, pages 214–229. Springer, 2004.
- [Mur06] S. Murphy. The independence of linear approximations in symmetric cryptanalysis. *IEEE Transactions on Information Theory*, 52(12):5510–5518, 2006.
- [MvOV96] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.
- [MY92] Mitsuru Matsui and Atsuhiro Yamagishi. A New Method for Known Plaintext Attack of FEAL Cipher. In *EUROCRYPT*, pages 81–91, 1992.
- [Nat01] National Institute of Standards and Technology (NIST). FIPS PUB 197: Advanced Encryption Standard. Federal Information Processing Standards Publication 197, U.S. Department of Commerce, November 2001. Available online: <http://www.itl.nist.gov/fipspubs>.
- [Nat07] National Institute of Standards and Technology (NIST). Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family. *Federal Register*, 27(212):62212–62220, November 2007. Available: [http://csrc.nist.gov/groups/ST/hash/documents/FR\\_Notice\\_Nov07.pdf](http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf).
- [NH07] K. Nyberg and M. Hermelin. Multidimensional Walsh transform and a characterization of bent functions. In P.V. Kumar T. Hellesteth and O. Ytrehus, editors, *Proceedings of the 2007 IEEE Information Theory Workshop on Information Theory for Wireless Networks*, IEEE, pages 83–86, 2007.
- [NK95] Kaisa Nyberg and Lars R. Knudsen. Provable security against a differential attack. *Journal of Cryptology*, 8(1):27–38, 1995.
- [NP07] M. Naya-Plasencia. Cryptanalysis of achterbahn-128/80. eSTREAM, ECRYPT Network of Excellence in Cryptology, report 2007-019, 2007.
- [NW97] Roger M. Needham and David J. Wheeler. Tea extensions. Computer Laboratory, Cambridge University, England, 1997. <http://www.movable-type.co.uk/scripts/xtea.pdf>.
- [Nyb95] K. Nyberg. Linear approximation of block ciphers. In A. De Santis, editor, *EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 439–444. Springer, 1995.
- [Nyb08] Kaisa Nyberg, editor. *Fast Software Encryption — 15th International Workshop, FSE 2008*, volume 5086 of *Lecture Notes in Computer Science*, Berlin, Heidelberg, New York, 2008. Springer-Verlag.

- [O’N07] Sean O’Neil. Algebraic structure defectoscopy. Cryptology ePrint Archive, Report 2007/378, 2007.
- [Pas08] Enes Pasalic. Transforming chosen iv attack into a key differential attack: how to break TRIVIUM and similar designs. Cryptology ePrint Archive, Report 2008/443, 2008.
- [PP04] S. Paul and B. Preneel. A new weakness in the rc4 keystream generator and an approach to improve the security of the cipher. In *Fast Software Encryption 2004*, volume 3017 of *LNCS*, pages 245–259. Springer-Verlag, 2004.
- [PPS06] S. Paul, B. Preneel, and G. Sekar. Distinguishing attacks on the stream cipher py. In *Fast Software Encryption 2006*, volume 4047 of *LNCS*, pages 405–421. Springer-Verlag, 2006.
- [RAB<sup>+</sup>] Ronald L. Rivest, Benjamin Agre, Daniel V. Bailey, Christopher Crutchfield, Yevgeniy Dodis, Kermin Elliott Fleming, Asif Khan, Jayant Krishnamurthy, Yuncheng Lin, Leo Reyzin, Emily Shen, Jim Sukha, Drew Sutherland, Eran Tromer, and Yiqun Lisa Yin. The MD6 hash function – a proposal to NIST for SHA-3. <http://groups.csail.mit.edu/cis/md6/>.
- [Rad06] Havard Raddum. Cryptanalytic results on Trivium. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/001, 2006.
- [Rec10] Christian Rechberger. Second-preimage analysis of reduced SHA-1. In *ACISP 2010*, Lecture Notes in Computer Science. Springer-Verlag, 2010. To appear.
- [RH02] G. Rose and P. Hawkes. On the applicability of distinguishing attacks against stream ciphers. IACR ePrint archive, <http://eprint.iacr.org/2002/142>, 2002.
- [Rij10] Vincent Rijmen. Stream ciphers and the estream project. *ISeCure*, 2(1):3–11, January 2010.
- [Riv95] R.L. Rivest. The RC5 encryption algorithm. *Dr Dobb’s Journal-Software Tools for the Professional Programmer*, 20(1):146–149, 1995.
- [RS96] Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1996.
- [SA08] Yu Sasaki and Kazumaro Aoki. Preimage attacks on 3, 4, and 5-pass HAVAL. In Josef Pawel Pieprzyk, editor, *Advances in Cryptology - ASIACRYPT 2008*, volume 5350 of *Lecture Notes in Computer Science*, pages 253–271, Berlin, Heidelberg, New York, 2008. Springer-Verlag.
- [SA09] Yu Sasaki and Kazumaro Aoki. Finding preimages in full MD5 faster than exhaustive search. In Antoine Joux, editor, *Advances in Cryptology — EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 134–152, Berlin, Heidelberg, New York, 2009. Springer-Verlag.
- [Saa06] Markku-Juhani Olavi Saarinen. Chosen-IV statistical attacks on eStream ciphers. In Manu Malek, Eduardo Fernández-Medina, and Javier Hernando, editors, *SECRYPT*, pages 260–266. INSTICC Press, 2006.

- [Sam07] Alex Samorodnitsky. Low-degree tests at large distances. In David S. Johnson and Uriel Feige, editors, *STOC*, pages 506–515. ACM, 2007.
- [Sel98] A.A. Selçuk. New results in linear cryptanalysis of RC5. In S. Vaudenay, editor, *FSE '98*, volume 1372 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 1998.
- [Sel08] A.A. Selçuk. On probability of success in linear and differential cryptanalysis. *Journal of Cryptology*, 21(1):131–147, 2008.
- [Sha] Adi Shamir. How to solve it: New techniques in algebraic cryptanalysis. Invited talk at CRYPTO 2008.
- [SK98] T. Shimoyama and T. Kaneko. Quadratic relation of S-box and its application to the linear attack of full round DES. In H. Krawczyk, editor, *CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 200–211. Springer, 1998.
- [SM88] A. Shimizu and S. Miyaguchi. Fast data encipherment algorithm FEAL. In D. Chaum and W.L. Price, editors, *EUROCRYPT'87*, volume 304 of *Lecture Notes in Computer Science*, pages 267–278. Springer, 1988.
- [Tao06] Terence Tao. The dichotomy between structure and randomness, arithmetic progressions, and the primes. In *International Congress of Mathematicians*, pages 581–608. European Mathematical Society, 2006.
- [TCG92] A. Tardy-Corffdir and H. Gilbert. A known plaintext attack of FEAL-4 and FEAL-6. In J. Feigenbaum, editor, *CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 172–181. Springer, 1992.
- [TK07] Meltem Sönmez Turan and Orhun Kara. Linear approximations for 2-round Trivium. eSTREAM, ECRYPT Stream Cipher Project, Report 2007/008, 2007.
- [TSK<sup>+</sup>05] Y. Tsunoo, T. Saito, H. Kubo, M. Shigeri, T. Suzuki, and T. Kawabata. The most efficient distinguishing attack on vmpe and rc4a. eSTREAM, ECRYPT Stream Cipher Project, report 2005/037, 2005.
- [Vau03] Serge Vaudenay. Decorrelation: A theory for block cipher security. *Journal of Cryptology*, 16(4):249–286, 2003.
- [Vie07] Michael Vielhaber. Breaking ONE.FIVIUM by AIDA an algebraic IV differential attack. Cryptology ePrint Archive, Report 2007/413, 2007.
- [Wag08] David Wagner, editor. *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *LNCS*. Springer, 2008.
- [Wei09] Ralf-Philipp Weinmann. AXR - Crypto Made from Modular Additions, XORs and Word Rotations. *Dagstuhl Seminar 09031*, January 2009. Available: <http://www.dagstuhl.de/Materials/AbstractListing/index.en.phtml?09031>.

- [WFW09] Shuang Wu, Dengguo Feng, and Wenling Wu. Cryptanalysis of the LANE Hash Function. In Jr. et al. [JRSN09], pages 126–140.
- [WP06a] H. Wu and B. Preneel. Attacking the iv setup of py and pypy. eSTREAM, ECRYPT Stream Cipher Project, report 2006/050, 2006.
- [WP06b] H. Wu and B. Preneel. Key recovery attack on py and pypy with chose ivs. eSTREAM, ECRYPT Stream Cipher Project, report 2006/052, 2006.
- [WP07] H. Wu and B. Preneel. Differential-linear attacks against the stream cipher phelix. eSTREAM, ECRYPT Stream Cipher Project, report 2007/056, 2007.
- [Wu09] Shuang Wu. Semi-free Start Collision for 12-round Cheetah-256. NIST hash function mailing list: `hash-forum@nist.gov`, 2009. Available online: <http://ehash.iaik.tugraz.at/uploads/0/08/Cheetah256-12r.txt>.
- [YWZW05] Hongbo Yu, Gaoli Wang, Guoyan Zhang, and Xiaoyun Wang. The Second-Preimage Attack on MD4. In Yvo Desmedt, Huaxiong Wang, Yi Mu, and Yongqing Li, editors, *CANS*, volume 3810 of *LNCS*, pages 1–12. Springer, 2005.
- [ZB05] E. Zenner and M. Bosgaard. How secure is secure? on message and iv lengths for synchronous stream ciphers. eSTREAM, ECRYPT Stream Cipher Project, report 2005/039, 2005.
- [Zol04] B. Zoltak. Vmpc one-way function and stream cipher. In *Fast Software Encryption 2004*, volume 3017, pages 210–225. Springer-Verlag, 2004.