KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Cryptanalysis of Symmetric-Key Primitives

Tomer Ashur

Supervisor:
Prof. dr. ir. Vincent Rijmen

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

December 2017

Cryptanalysis of Symmetric-Key Primitives

Tomer ASHUR

Examination committee:
Prof. dr. Adhemar Bultheel, chair
Prof. dr. ir. Vincent Rijmen, supervisor
Prof. dr. ir. Bart Preneel
Prof. dr. ir. Ingrid Verbauwhede
Prof. dr. ir. Patrick Wambacq

Prof. dr. Lars Ramkilde Knudsen
(Technical University of Denmark)
Prof. dr. Orr Dunkelman
(University of Haifa)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

December 2017

Preface

> *"'But what are we going to do?' Colonel Cathcart exclaimed with distress. 'The others are all waiting outside.' 'Why don't we give him a medal?' Colonel Korn proposed. 'For going around twice? What can we give him a medal for?' 'For going around twice,' Colonel Korn answered with a reflective, self-satisfied smile. 'After all, I suppose it did take a lot of courage to go over that target a second time with no other planes around to divert the antiaircraft fire. And he did hit the bridge. You know, that might be the answer — to act boastfully about something we ought to be ashamed of. That's a trick that never seems to fail."'*
>
> - Joseph Heller, *Catch-22*

So this is how it feels. The author puts down the pen and reviews his work. Tired of thinking, wary of writing, he compiles one last time. Printing. How truly remarkable it is to be able to hold the fruits of your own labor. If you read this, it means that you too hold a copy of my work (you may be viewing this through a computer screen, but this scenario is so much less romantic that we can neglect it in what follows). Writing this Thesis is, hands down, my greatest achievement so far.[1] It is also one of the hardest. Not the writing part (although this too was not easy), but the work leading to it.

The book you are now holding is both the best and worst of me. I am almost tempted not to submit it, for fear of how it will be judged. It has been an amazing journey.[2] The view was great, the target is satisfying, but most importantly, I would not have changed the company. There are so many people

[1] David Broza, a prominent Israeli singer was offered the opportunity to publish a collection of his "best" work. He declined, saying that he is still an active performer. So they settled for the title *The best of David Broza (so far)*.

[2] Yes, this is a thesis-as-a-journey metaphor. How original.

without whom this Thesis could not have been possible. It would be impossible to name them all, so I will only mention a few.

First and foremost, to my Michal. It is impossible to imagine who I was without you. In the words of Orson Scott Card: "I was not alive until I met my beloved, and now the moon is new, the sea is blue, the month is June, our love is true". this Thesis is dedicated to you.

To Vincent, the best promoter a student can wish for. When writing this, I was not sure if I wanted to thank you more for the freedom to pursue my own path or for the guidance. It was a difficult choice and I decided to go with a Belgian solution: thank you for just the right mix of freedom and guidance.

To Orr, the first person in about 15 years of education to have faith in my academic skills. It is impossible to say how much I owe you. I hope this Thesis makes you proud.

To the members of my supervisory committee, Ingrid Verbauwhede and Patrick Wambacq, for spending the time reading my reports, listening to my presentations, and making sure I'm on the right track to reach the point where I can write this acknowledgment. To the head of my group, Bart Preneel, who is tasked with the ungrateful and under-appreciated job of making the wheels turn. To the last member of my examination committee, Lars R. Knudsen, for taking the time to be on this committee. I hope it was worth it.

To Leandro Panizzon the inventor of Ritalin (*Methylphenidate; Rilatine)* without whom this Thesis would not have been possible, Ciba (now Novartis) for its distribution, and Sophie Maes for prescribing it.

To Belgium, the country which received me so well. It is a wonderful country, albeit (or maybe because) it is so fun to make fun of. A special thanks goes to Atul for helping Michal and myself with our integration, and for being such a great friend. This experience would have been completely different without you. To my various co-authors not yet mentioned: Bart Mennink, Yunwen Liu, Achiya Bar-On, Nimrod Talmon, Daniël Bodden, Glenn De Witte, Adrián Ranea, and Tim Beyne. Those parts of the joint work that are good are thanks to them, any mistake is mine alone.[3] The same goes to Dana, Nele, Lennert, and Jan-Pieter for helping with the Dutch parts of this Thesis. Sorry I didn't accept all of your suggestions—I did it in order to provide you with plausible deniability in case of any translation errors. To the COSIC support staff: Péla, Elsy, Wim, and Saartje without whom everyone would probably be sitting on the floor. To the ESAT support staff, and especially the system group. As I said

[3]In the case of Daniël, Glenn, and Adrián, whom I supervised as master students, some of the mistakes were certainly theirs, but I still take the responsibility for not spotting those. Sorry guys.

before, this is the best and most professional ICT team I have ever encountered, including those under my own command.

To people who contributed to my research. I can't actually remember the contribution of each individual, but since I acknowledged all of you in my publications, it must have been substantial: Adi Shamir, Nathan Keller, Daniel J. Bernstein, Michael Klots, Elena Andreeva, Reza Reyhanitabar, Marc Stevens, Ernst Schulte-Geers, Kaisa Nyberg, Aviad Stier, Jonathan J. Klinger, Amihai Bannett, Dubi Kanengisser, Gustavo Mesch, Claudia Diaz, Tamar Zondiner, Yair Goldberg, Alan Szepieniec, Shir Peled, Güneş Acar, Roger Dingledine, Ian Goldberg, and Marc Juarez.

The support of various funding providers is highly appreciated and was extremely useful (I work better when I'm not hungry and cold): The Israeli Ministry of Science and Technology, the Google Europe Scholarship for Students with Disabilities, The Belgian Federal Science Office (BELSPO) and the KU Leuven Research Fund.

To friends (idiosyncratic and other): Adi & Nofar, Shai & Hanna, Eytan, Shkol & Shay, Barend & Cata, Alex & Stefano, Guilel & Dim, Ohad & Nele, Brian & Peter, Ilana & Siebrecht, and the rest of the OY community.

To all members of COSIC, past and present, who made this journey so remarkable: Abdel Aly, Adrián Ranea, Angshuman Karmakar, Anthony Van Herrewege, Antoon Bosselaers, Archana Bindu Sunil, Arthur Beckers, Ashrujit Ghoshal, Aysajan Abidin, Begül Bilgin, Benedikt Gierlichs, Bing Sun, Bohan Yang, Carl Bootland, Chaoyun Li, Charlotte Bonte, Christina-Angeliki Toli, Danilo Šijačić, Danny De Cock, Dave Singelée, Deniz Toz, Dušan Božilov, Edu Marín, Eleftheria Makri, Enrique Argones Rúa, Ero Balsa, Fatemeh Shirazi, Filipe Beato, Fré Vercauteren, Furkan Turan, Gabriele Calianno, Gunes Acar, Hiro Yoshida, Hyunmin Kim, Ilia Iliashenko, Iraklis Symeonidis, Jens Hermans, Jeroen Delvaux, Jose Maria Bermudo Mera, Josep Balasch, Kan Yasuda, Kent Chuang, Kerem Varici, Kimmo Järvinen, Lauren De Meyer, Liliya Kraleva, Meghna Sengupta, Miloš Grujić, Mustafa Mustafa, Nele Mentens, Nicky Mouha, Oscar Repáraz, Pieter Maene, Qingju Wang, Rafa Galvez, Ren Zhang, Roel Peeters, Ruan de Clercq, Sanghan Lee, Sara Cleemput, Seda Gürses, Simon Dhooghe, Simon Friedberger, Sujoy Sinha Roy, Svetla Petkova-Nikova, Tariq Elahi, Thomas De Cnudde, Victor Arribas Abril, Vladi Rozic, Wenying Zhang, Wouter Castryck, and Yoni De Mulder.

To fallen comrades.

To my siblings - you can't imagine how much I miss you.

And to anyone else I might have forgotten. If you deserve to be mentioned here

and weren't, it is simply because you are too deep in my heart for my brain to remember.

Abstract

"All of the true things I am about to tell you are shameless lies."

- Kurt Vonnegut, *Cat's Cradle*

Cryptography is that field of science dealing with secure communication in the presence of an adversary. From an esoteric craft, on par with tasseography and practiced by few, cryptography grew to be a scientific field whose practitioners are portrayed as heroes in popular media (and rightly so!). What required large codebooks 400 years ago, complicated machinery less than a 100 years ago, or a PC sized box 10 years ago, can now be done by a chip embedded in one's shoe.

Symmetric-key cryptography can be viewed as being crudely composed of three parts: design, analysis, and implementation. The role of designers is to build new systems using a mix of mathematical techniques, engineering methods, and their own good instincts. Implementers are tasked with (securely) translating those abstract designs into something tangible. Analysis deals with the security evaluation of abstract cryptosystems (cryptanalysis) and of their implementation (side-channel analysis). This thesis is about cryptanalysis.

An attempt was made by the Author[4] to give this Thesis a natural flow from theory to application. Chapter 1 is an introduction to this Thesis. The mathematical background required for understanding the rest of the Thesis is provided, as well as a description of the cryptosystems used later in the Thesis to demonstrate new ideas.

This Author's original contribution starts in Chapter 2 where he revisits the theory of cryptanalysis. The first contribution deals with 1-round linear hulls. Their existence is shown and the way they complicate the bias estimation for linear cryptanalysis is discussed. A method for overcoming this complication

[4]For the rest of this Thesis, when capitalized, the word Author can be substituted for a first person pronoun. Likewise, when the word thesis is capitalized it refers to this Thesis.

and a correction to the way linear potential is calculated are presented. The second contribution is a re-evaluation of Matsui's Algorithm 1. It is shown that using the bias from a single trail as an estimate for its hull's bias cripples the success probability which will sometimes even decrease as a result of increasing the data complexity. As a third contribution the wrong-key-randomization hypothesis is reconsidered under the distinct known-plaintext model. As a fourth contribution, the non-monotonicity of a linear attack's success probability is explained and quantified. Finally, as a fifth contribution, a discussion about misconceptions and open problems in linear cryptanalysis is presented.

Chapter 3 is about new methods in cryptanalysis. The first contribution of this chapter is the discovery on RX-cryptanalysis. It is shown that the injection of round constants is not a proper defense against rotational cryptanalysis, and a method for dealing with these constants is developed. Then, a second contribution in this chapter shows that despite the widespread belief, bounding the absolute bias of linear approximations to $2^{-n/2}$ is insufficient to resist linear cryptanalysis. Two new types of distinguishing attacks are developed, one using multiple keys and one using multiple approximations. In both cases we see that linear approximations with absolute bias smaller than $2^{-n/2}$ can still be used to distinguish a block cipher from a random permutation.

The viewpoint of the Thesis then changes from theory to application in Chapter 4 where two automated tools for cryptanalysis are proposed. The first tool is used to search for linear trails in ARX constructions by limiting the search space from all masks to masks containing even-length sequences of active bits. The second tool introduces a Python-like programming language for the description of ARX primitives. After a primitive is described using this language, the tool converts it into a set of logical constraints and uses a SAT/SMT solver to find an optimal RX-characteristic.

Chapter 5 is about performing cryptanalysis. A trivial forgery attack against p-omd is presented. The attack uses only 3 messages and succeeds with probability 1. Then, a reflection attack, an impossible reflection attack, and a fixed-point attack are presented against Gost2, as well as 3 related-key differential attacks against the full cipher. Finally, the resistance of Speck against RX-cryptanalysis is evaluated and previously best distinguishers are extended by 1–4 rounds.

Finally, 4 contributions are presented in Chapter 6. First, a security analysis is performed for the authentication service of Galileo, the European Union's Global Navigation Satellite System (GNSS), and security parameters are recommended. Then, Spoed and Spoednic are introduced. These are 2 authenticated encryption schemes inspired by p-omd. Spoed is a simplified and improved version of p-omd, and Spoednic is an extension restoring p-

OMD's lost nonce-misuse resistance. The last contribution is a novel method for obtaining RUP security. A generic construction is suggested and shown to be secure. Then, we instantiate the construction using components from the GCM mode of operation and show that GCM can be made RUP-secure by a few minor tweaks.

Chapter 7 concludes this Thesis.

Beknopte samenvatting

"De vos weet vele dingen, maar de egel weet één groot ding"

- Erasmus, *Adagia*

Cryptografie is het wetenschappelijke gebied waarin beveiligde communicatie in de aanwezigheid van een tegenstander onderzocht wordt. Van een esoterisch vak, op gelijke voet gesteld met tasseografie en slechts door enkelen beoefend, groeide cryptografie uit tot een wetenschappelijk vakgebied waarvan de beoefenaren als helden worden voorgesteld in de media (en terecht!). Wat 400 jaar geleden enkel kon gedaan worden met grote codeboeken, complexe machinerie minder dan 100 jaar geleden of een doos ter grootte van een PC 10 jaar geleden, kan nu worden gedaan met een chip die in een schoen ingebed zit.

Symmetrische cryptografie kan ruwweg gezegd in drie delen verdeeld worden: ontwerp, analyse en implementatie. De rol van ontwerpers is om nieuwe systemen te bouwen met een mix van wiskundige technieken, ingenieurs methodes, en hun gezond verstand. Implementeerders vertalen die abstracte ontwerpen naar iets tastbaars (op een veilige manier). Analyse omvat de veiligheidsevaluatie van abstracte cryptosystemen (cryptanalyse) en hun implementatie (zijkanaal-informatie analyse). Deze Thesis gaat over cryptanalyse.

De Auteur heeft een poging ondernomen om deze Thesis van theorie tot toepassing van een natuurlijke stroming te voorzien. Hoofdstuk 1 is een introductie tot deze Thesis. De wiskundige achtergrond die nodig is om de rest van de Thesis te begrijpen, en een beschrijving van de cryptosystemen die later in de Thesis gebruikt worden om nieuwe ideeën aan te tonen, worden daarin voorzien.

De bijdrage van de Auteur begint in hoofdstuk 2 waar hij de theorie van cryptanalyse herziet. De eerste bijdrage behandelt 1-ronde lineaire omhulsels. Hun bestaan wordt aangetoond, en de manier waarop ze de onbalans schatting voor lineaire cryptanalyse compliceren, wordt besproken. Een methode om deze

complicatie te overwinnen en een correctie op de manier waarop lineair potentieel berekend wordt, worden voorgesteld. De tweede bijdrage is een reëvaluatie van *Matsui's Algorithm 1*. Er wordt aangetoond dat het gebruik van de onbalans van een enkelvoudig pad als een schatting voor de onbalans van het hele omhulsel de kans op succes belemmert, en dat die soms zelfs nog zal afnemen door de data complexiteit te verhogen. Als derde bijdrage wordt de foute-sleutel-randomizatie hypothese herbekeken onder het verschillende gekend-bericht model. De vierde bijdrage houdt in dat de niet-monotone slaagkans van een lineaire aanval wordt uitgelegd en gemeten. Tenslotte, als vijfde bijdrage, wordt een discussie over misvattingen en huidige problemen in de lineaire cryptanalyse voorgesteld.

Hoofdstuk 3 gaat over nieuwe methodes in cryptanalyse. De eerste bijdrage van dit hoofdstuk is de ontdekking van RX-cryptanalyse. Er wordt aangetoond dat de injectie van rondconstanten geen degelijke verdediging tegen rotationele cryptanalyse biedt, en een methode om met deze constanten om te gaan wordt ontwikkeld. Een tweede bijdrage in dit hoofdstuk toont aan dat, in tegenstelling tot wat doorgaans wordt aangenomen, de absolute waarde van de onbalans tot $2^{-(n-1)/2}$ beperken onvoldoende is om bestand te zijn tegen lineaire cryptanalyse. Twee nieuwe types van onderscheidende aanvallen worden ontwikkeld: een die meerdere sleutels gebruikt en een die meerdere benaderingen gebruikt. In beide gevallen zien we dat een lineaire benadering met een absolute onbalans kleiner dan $2^{-(n-1)/2}$ nog steeds gebruikt kan worden om een blokvercijfering van een willekeurige permutatie te onderscheiden.

Het standpunt van de Thesis verandert van theorie naar applicatie in hoofdstuk 4, waar twee geautomatiseerde tools voor cryptanalyse worden voorgesteld. De eerste tool wordt gebruikt om lineaire paden in ARX-constructies te zoeken door de zoekruimte van alle maskers te beperken naar maskers die een sequentie van gelijke lengte van actieve bits bevatten. De tweede tool gebruikt een Python-achtige programmeertaal voor de beschrijving van ARX-primitieven. Nadat een primitief met deze taal werd beschreven, converteert de tool het naar een set van logische beperkingen en gebruikt het een SAT/SMT oplosser om een optimaal RX-pad te vinden.

Hoofdstuk 5 gaat over het uitvoeren van cryptanalyse. Een triviale vervalsingsaanval tegen P-OMD wordt voorgesteld. De aanval gebruikt slechts 3 berichten en slaagt met waarschijnlijkheid van 1. Nadien worden een reflectieaanval, een onmogelijke reflectieaanval, en een dekpuntaanval getoond tegen GOST2, net als 3 verwant-sleutel differentiële aanvallen tegen de hele vercijfering. Tenslotte wordt de weerstand van SPECK tegen RX-cryptanalyse geëvalueerd en de vroegere beste onderscheiders worden uitgebreid met 1–4 ronden.

In hoofdstuk 6 worden 4 bijdragen voorgesteld. Eerst wordt een veiligheidsanalyse van de authenticatiedienst van GALILEO uitgevoerd, het globale satellieten navigatiesysteem van de Europese Unie, en veiligheidsparameters worden aangeraden. Daarna worden SPOED en SPOEDNIC geïntroduceerd. Dit zijn twee geauthentiseerde encryptie-algoritmes die geïnspireerd werden door P-OMD. SPOED is een vereenvoudigde en verbeterde versie van P-OMD, en SPOEDNIC is een extensie die P-OMD's verloren nonce-misbruik weerstand herstelt. De laatste bijdrage is een nieuwe methode om UNK-veiligheid te verkrijgen. Een generische constructie wordt voorgesteld en er wordt aangetoond dat die veilig is. Daarna bevestigen we de constructie met componenten van de GCM *modus operandi* en tonen we dat GCM UNK-veilig kan worden gemaakt mits er enkele kleine aanpassingen plaatsvinden.

De Thesis wordt besloten met hoofdstuk 7.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

> *"Converte gladium tuum in locum suum. Omnes enim, qui acceperint gladium, gladio peribunt."*
>
> - Matthew, *26:52*

The standard exposition for a thesis in cryptography is to write how human beings required cryptography since the dawn of time. While this is possibly true, it misses the point altogether. The examples given to uses of cryptography are always from politics (including war, which is a mere continuation of policy by other means [158]). Until the 1970's, cryptography was used exclusively, or at least chiefly, by nations. Similarly, cryptanalysis was used solely by governments for warfare and diplomacy.

But cryptography is in fact much more than that. The academic interest in cryptography which began in the 1970's had been ever since a locomotive pulling technological advancement. Although computer communication preceded cryptography as a civilian tool, the latter did not lag behind. At first, civilian use of cryptography was a mirror image of military applications—hiding the content of messages and verifying their integrity. However with time, cryptography grew to offer more than just that. In today's world, advances in cryptography allow for things that were previously impossible. Cryptocurrencies, car sharing, and easy exercise of democracy through electronic voting are all new technologies enabled by cryptography.

This is the context in which cryptanalysis should be considered. Indeed, cryptography can be split into three parts: design, cryptanalysis, and

implementation. While it seems that design and cryptanalysis are naturally interleaved, this must not be so. De Leeuw notes in [62] that although Dutch cryptanalysts were able to break Spanish codes during the 17th century, no improvement was observed in their own use of codes. This suggests that code breakers (as they were called back then) and code designers worked independently and did not share knowledge.

There might not be a parallel to the dissonance between how a cryptanalyst is viewed by society and what they really do.[1] Alan Turing is often said to be the most influential individual on World War 2. Note that in hindsight, we know the names of the people breaking the Enigma but not the names of its designers. "It has been estimated", writes Kahn in [93], "that cryptanalysis saved a year of war in the Pacific". In practice, the cryptanalyst's job is tedious and often monotone. It involves applying both math and intuition, long frustrated stares at blank sheets, or worse—sheets full of non-elegant equations. Cryptanalysis is the art of "shaving bits", *i.e.*, reducing the work effort required to break a cryptosystem from incomprehensible to unimaginable and then arguing that the latter has practical significance.

But in fact, cryptanalysis can be viewed as garbage collection, and hence has its own nobility. As a sanitary worker, the cryptanalyst's job is to sieve and sort candidates. Some of these candidates are complete garbage and should be disposed. Others are worn out, and although nostalgia makes it hard, there must come a time when they are replaced. Often, candidates can be recycled, not usable by themselves but offer insights as to what a good cryptosystem would look like. Once in a while, a gem is found. It is the cryptanalyst's job to sort these candidates, prune the bad and nurture the potential ones.

The focus of this Thesis is cryptanalysis and in the following chapters the reader will be exposed to several facets of it. The rest of this chapter lays the groundwork for everything coming next. In the next section (Section 1.1), relevant parts from the theory of cryptanalysis are presented. The section following (Section 1.2) presents cryptosystems used in the sequel. While some find them interesting by their own merits, they are considered here only as a lens allowing to look deeper into cryptanalysis.

The next two chapters deal with the theory of cryptanalysis. In Chapter 2 we revisit the existing theory of cryptanalysis to identify caveats and errors. The former are then addressed, and the latter are corrected. Although the importance of doing so is self-obvious, in light of this Author's communication with some of his colleagues it is never a bad idea to reiterate it: theory is the gauge allowing to distinguish secure cryptosystems from insecure ones. Then,

[1] *E.g.*, Kahn's observation in [93]: "Codebreaking is the most important form of secret intelligence in the world today."

in Chapter 3, the theory of cryptanalysis is extended and new cryptanalytic techniques are developed.

Chapter 4 deals with automated tools. The work presented in this chapter describes automated ways for finding good properties that can be used in statistical cryptanalysis.

In Chapter 5 we cryptanalyze existing cryptosystems. This is the pruning part in the metaphor above. Tools and techniques developed in Chapters 3–4 (as well as tools and techniques developed by others) are applied to existing cryptosystems.

Finally, Chapter 6 applies the lessons learned during the security evaluation of cryptosystems to build better cryptosystems.

Chapter 7 concludes this Thesis.

1.1 Cryptanalysis

This section provides the background required for reading this Thesis. The first part provides mathematical and statistical background. Then, the next four parts deal with existing cryptanalytic techniques: differential cryptanalysis, linear cryptanalysis, rotational cryptanalysis, and self-similarity attacks. Finally, in Section 1.1.6 key recovery algorithms are discussed.

1.1.1 Mathematical and Statistical Background

In this section we briefly describe mathematical and statistical notions used in the rest of this Thesis.

Boolean functions

We denote the field with two elements by $\mathrm{GF}(2)$ and the vector space of dimension n over this field by $\mathrm{GF}(2)^n$. We use $\oplus$ to denote addition in $\mathrm{GF}(2)$ and $\boxplus$ to denote modular addition in some finite group. The field in which the addition is made is always clear from the context.

A Boolean function $y = f(x)$ is a function $f : \mathrm{GF}(2)^n \to \mathrm{GF}(2)$ mapping a vector of size n with binary components into a single bit. A Boolean vector function $y = F(x)$ is a function $F : \mathrm{GF}(2)^n \to \mathrm{GF}(2)^m$ that maps a binary vector of size n into a binary vector of size m. A permutation is an invertible

Boolean vector function. A Boolean vector function $y = F(x)$ with output size m can be viewed as the parallel execution of m Boolean functions such that $y_i = F_i(x)$ where $0 \leq i \leq m-1$ denotes the bit position.

A keyed Boolean vector function $y = F(k, x) = F_k(x)$ is a family of Boolean vector functions, indexed by a key k. An iterative block cipher with r rounds is a composition of r permutations $F_{k_{r-1}} \circ F_{k_{r-2}} \circ \ldots \circ F_{k_0}(x)$. For the sake of brevity, and in compliance with existing literature we will assume that the round keys k_i are independent and the key of a block cipher, denoted by k, is defined as the string consisting of the concatenation of all r round keys k_i.

Probability distributions

A probability distribution is a well defined set of events and the likelihood of their occurrence. In this Thesis, four well known probability distributions are used: the binomial distribution, which we mark with $\mathcal{B}$, the hypergeometric distribution which we mark with $\mathcal{HG}$, the normal distribution which we mark with $\mathcal{N}$ and the χ^2 distribution.

A binomial variable $\boldsymbol{X_0} \sim \mathcal{B}(N, p)$ is a random variable counting the number of successes in a set of N independent experiments, each with two possible results (which are "success" and "failure", without loss of generality), and each experiment is independent and has a success probability of p.

Similarly, a hypergeometric variable $\boldsymbol{X_1} \sim \mathcal{HG}(N, M, R)$ means that $\mathbf{X_1}$ follows a hypergeometric distribution, *i.e.*, $\mathbf{X_1}$ is a random variable counting the number of occurrences of an item of *Type I* in N draws from a population of size M known to include R such items, where the samples are not returned to the "box" after being sampled and the probability to draw an item of a certain type therefore depends on the outcomes of previous draws.

The binomial and hypergeometric distributions are discrete distributions, *i.e.*, they deal with counting events. Unlike those, the *normal distribution* is a family of continuous distributions. We say that $\boldsymbol{X_2} \sim \mathcal{N}(\mu, \sigma^2)$, follows a normal distribution with mean μ and variance σ^2.

The normal distribution appears in many natural phenomena. This is the result of the *central limit theorem* showing that the sum of random variables often converges to a normal distribution. The *standard normal distribution* is a normal distribution with mean $\mu = 0$ and variance $\sigma^2 = 1$. A simple transformation allows to convert any normal variable into a standard normal variable.

Lemma 1. *If* $\boldsymbol{X_2} \sim \mathcal{N}\left(\mu, \sigma^2\right)$ *then*

$$\frac{\boldsymbol{X_2} - \mu}{\sigma} \sim \mathcal{N}\left(0, 1\right).$$

Both the binomial and hypergeometric distributions can be approximated by a normal distribution. The normal approximation for the binomial distribution is easy to show and can be found in any probability theory textbook:

Lemma 2 (The normal approximation of the binomial distribution)**.** *For a sufficiently large* N*, if* $\boldsymbol{X_0} \sim \mathcal{B}\left(N, p\right)$ *then*

$$\boldsymbol{X_0} \approx \mathcal{N}\left(Np, Np\left(1-p\right)\right).$$

The normal approximation of the hypergeometric distribution can take several forms. For our purposes, the one from Feller should suffice:

Lemma 3 ([70])**.** *Let* $\boldsymbol{X_1} \sim \mathcal{HG}\left(N, M, pM\right)$*. If* $N, M \to \infty$ *in such manner that* $N/M \to t \in (0,1)$*, then* $\boldsymbol{X_1}$ *has asymptotic distribution*

$$\boldsymbol{X_1} \approx \mathcal{N}\left(pN, N\left(1-t\right)p\left(1-p\right)\right).$$

Proof. Two proofs for Lemma 3 can be found in [130]. □

Despite the fact that the normal distribution is only an approximation for the binomial and hypergeometric distributions, we shall nevertheless use it in an exact way and write $X \sim \mathcal{N}\left(\cdot, \cdot\right)$.

The last family of distributions used in this Thesis is the χ^2 family. While the χ^2 family is an object of research of its own accord, we are interested in it insofar as it describes the behavior of a squared standard normal variable. *I.e.*, if $\boldsymbol{X_2} \sim \mathcal{N}\left(0, 1\right)$ then $\left(\boldsymbol{X_2}\right)^2 \sim \chi^2_1$. Here, the subscript of χ^2 denotes the degrees of freedom the distribution has. The sum of a set of standard normal variables $\mathbf{X_2^{(1)}}, \ldots, \mathbf{X_2^{(m)}}$ where each sample is drawn independently is said to follow a χ^2_m distribution, *i.e.*,

$$\sum_{i=1}^{m} \left(\boldsymbol{X_2^{(i)}}\right)^2 \sim \chi^2_m.$$

All of these distributions will be used in the sequel.

Compound probability distributions

A compound probability distribution is a probability distribution with one of its parameters being in itself a random variable. When working with compound distributions it is important to distinguish between two different random variables: the sample value of the random variable, and the sample value of the random variable given the value of its parameter. We shall refer to the former by $\hat{\boldsymbol{\epsilon}}_w$ and the latter will be written as $\hat{\boldsymbol{\epsilon}}_w \mid \boldsymbol{\epsilon}_w$.

The probability density of the compound variable $\hat{\boldsymbol{\epsilon}}_w$ is given by the probability density of $\hat{\boldsymbol{\epsilon}}_w | \boldsymbol{\epsilon}_w = \varepsilon$, weighted by the probability that $\boldsymbol{\epsilon}_w = \varepsilon$ for any possible ε that $\boldsymbol{\epsilon}_w$ can take. Formally, if $f_{\hat{\boldsymbol{\epsilon}}_w|\boldsymbol{\epsilon}_w}$ is the probability density function of $\hat{\boldsymbol{\epsilon}}_w \mid \boldsymbol{\epsilon}_w$, and $f_{\boldsymbol{\epsilon}_w}$ likewise for $\boldsymbol{\epsilon}_w$, then we may write

$$f_{\hat{\boldsymbol{\epsilon}}_w}(\epsilon_w) = \int_{\varepsilon} f_{\hat{\boldsymbol{\epsilon}}_w|\boldsymbol{\epsilon}_w}(\epsilon_w, \varepsilon)\, f_{\boldsymbol{\epsilon}_w}(\varepsilon)\, d\varepsilon,$$

for the density of $\hat{\boldsymbol{\epsilon}}_w$. This is depicted in Figure 1.1.

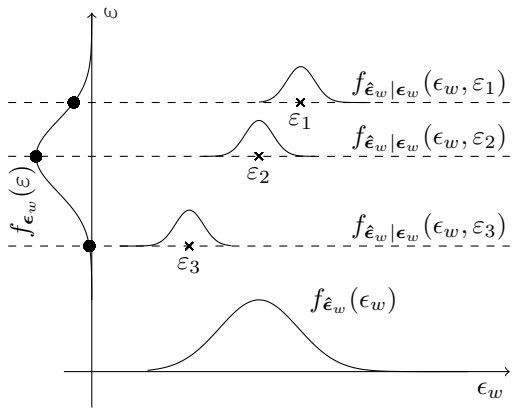


Figure 1.1: **A compound model.** The curve along the vertical axis represents the density function of $\boldsymbol{\epsilon}_w$. The probability density function of $\hat{\boldsymbol{\epsilon}}_w$ is shown at the bottom. $\hat{\boldsymbol{\epsilon}}_w$ has a compound distribution obtained by weighed integration over the smaller curves which represent the conditional density of $\hat{\boldsymbol{\epsilon}}_w$ for different values of $\boldsymbol{\epsilon}_w$.

1.1.2 Differential Cryptanalysis

One of the most notable techniques in cryptanalysis is differential cryptanalysis. Developed by Biham and Shamir and published in 1990 [30], differential cryptanalysis examines the evolution of differences during the encryption of two inputs. An input difference is the difference between two inputs entering a cryptosystem, usually with respect to the exclusive-or (XOR) operation. The output difference is the difference between the outputs of two such inputs. We say that an input difference Δ causes an output difference ∇ under the function f with probability p if a portion p of the possible pairs of inputs having a difference Δ results in outputs having a difference ∇ after applying f. When this happens, we write $\Delta \xrightarrow{f} \nabla$ with probability p.

A differential characteristic that describes a single round of a cryptosystem is called a 1-round characteristic. Biham and Shamir showed that two or more such characteristics can be concatenated to form longer multi-round characteristics if the output difference of one characteristic is the input difference of the other.

Example 1. *Let P and P' be two plaintexts and let $\Delta = P \oplus P'$ be defined as their XOR difference. The most common application of differential cryptanalysis is in the chosen-plaintext model (*i.e.*, where the model gives the adversary some control over the inputs) and with a fixed input difference in mind. For example, an adversary that is interested in an input difference $\Delta = \texttt{00400000}_\texttt{x}$ will generate a random plaintext P and set $P' = P \oplus \texttt{00400000}_\texttt{x}$.*

Let C and C' denote the encryption of P and P' through a single round of the block cipher Simon*32 (a description of* Simon *can be found in Section 1.2 and is immaterial for this example). As per [2], if $\Delta = \texttt{00400000}_\texttt{x}$ is the input difference entering 1-round* Simon*, then with probability 2^{-2} the output difference is $\nabla = C \oplus C' = \texttt{01000040}_\texttt{x}$.*

We can extend this 1-round differential characteristic by another round by considering $\nabla = \texttt{01000040}_\texttt{x}$, the output difference of the first round as an input difference to the next round. Again, [2] shows that an input difference $\nabla = \texttt{01000040}_\texttt{x}$ leads after a single round of Simon *to an output difference of $\Diamond = \texttt{04400100}_\texttt{x}$ with probability 2^{-2}. Multiplying these two probabilities we get that the probability of the 2-round differential characteristic $(\texttt{00400000}_\texttt{x}, \texttt{01000040}_\texttt{x}, \texttt{04400100}_\texttt{x})$ is 2^{-4}.*

A differential is a set of differential characteristics having the same input and output difference, disregarding intermediate differences. The probability of a differential is the sum of probabilities of all differential characteristics in the underlying set. Other extensions of differential cryptanalysis include, amongst others, the boomerang attack [159], impossible differential cryptanalysis [29],

truncated differential cryptanalysis [98], and higher-order differentials [98]. In this Thesis, differential cryptanalysis is used as one of the building blocks for a new cryptanalytic technique, RX-cryptanalysis, which is developed in Section 3.1.

1.1.3 Linear Cryptanalysis

Linear cryptanalysis is a known-plaintext attack trying to identify linear relations between plaintext bits and ciphertext bits. The attack was presented by Matsui in [110] and was the first attack against DES with feasible complexities. The idea behind linear cryptanalysis is to find a probabilistic relation between the parity of some subset of ciphertext bits and that of plaintext and key bits. Identifying this relation is the first phase in a linear attack. In the second phase, a statistical procedure is used for distinguishing or key recovery.

Extensions of linear cryptanalysis include, amongst others, multiple linear cryptanalysis [31, 92], multidimensional linear cryptanalysis [80], multivariate linear cryptanalysis [43], and zero-correlation attacks [40, 41]. In this Thesis, we present in Section 4.1 an automated tool assisting in the first phase of the attack. The second phase is extensively analyzed in Chapter 2 where existing models are refined and errors exposed. In Section 3.2 a new extension of (multiple) linear cryptanalysis is presented that can be used against cryptosystems which were previously believed to be secure with respect to linear attacks.

In this section, we recall some definitions and terminology of linear cryptanalysis from [28, 48, 58, 95, 110, 123].

Masks and approximations

Let α, β be two vectors of size n. Then

$$\alpha^t \cdot x = \bigoplus_{i=0}^{n-1} \alpha_i \ \& \ x_i \,,$$

where the subscript i denotes the i-th bit of a vector. We will call α the mask of x. In practical examples, the masks will often contain many zero bits. In order to emphasize which bits are nonzero, we will sometimes use the following set notation:

$$\alpha = \{i_1, i_2, \ldots, i_u\} \Leftrightarrow \begin{cases} \alpha_i = 1, \forall i \in \{i_1, i_2, \ldots, i_u\} \\ \alpha_i = 0, \forall i \notin \{i_1, i_2, \ldots, i_u\}. \end{cases}$$

A linear approximation for a keyed Boolean permutation is a tuple (α, β, γ) such that α, β, and γ are masks for the input, the output and the key, respectively. Let p be the fraction of inputs x for which the equation $\alpha^t \cdot x \oplus \beta^t \cdot F_k(x) \oplus \gamma^t \cdot k = 0$ holds. The correlation of the linear approximation (α, β, γ) is defined as $\mathrm{cor}(\alpha, \beta, \gamma) = 2 \cdot \left(p - \frac{1}{2}\right) = 2p - 1$. In general, both p and $\mathrm{cor}(\alpha, \beta, \gamma)$ will depend on k. Similar to differential cryptanalysis, linear approximations can be concatenated to form longer approximations if the output mask of one is the input mask of the other. A sequence of r concatenated approximations $(\alpha_i, \beta_i, \gamma_i)$ is called a *linear trail*. When $\gamma = 0$, we abbreviate the notation $(\alpha, \beta, 0)$ and $\mathrm{cor}(\alpha, \beta, 0)$ to (α, β) and $\mathrm{cor}(\alpha, \beta)$, respectively, and the approximation is called a *linear hull*.

The existing literature interchangeably uses correlations and biases. The bias $-\frac{1}{2} \leq \epsilon \leq \frac{1}{2}$ and correlation $-1 \leq c \leq 1$ of a linear approximation are both measures for the error in a linear approximation, or in simple words, how its probability differs from $\frac{1}{2}$. The probability p, the bias ϵ, and the correlation c can be obtained from one another through $\epsilon = c/2 = p - \frac{1}{2}$. Correlations are more frequently used when the mathematical aspects of linear cryptanalysis are investigated, and biases are more often used when linear cryptanalysis is applied to a cryptosystem. We will continue this trend and use both terms, bearing in mind that the bias can easily be obtained from the correlation and *vice versa*.

Linear hulls and trails

Following the terminology of [59], a (linear) trail Ω covering r rounds of an iterative block cipher is a concatenation of linear approximations each covering a single round such that the output mask of round i equals to the input mask of round $i + 1$. Hence, we can identify the trail with a vector of $r + 1$ masks $\omega_i, 0 \leq i \leq r$, $\Omega = (\omega_0, \omega_1, \ldots, \omega_r)$. Round i has input mask ω_i and output mask ω_{i+1}. The correlation contribution of a trail Ω is the product of the correlations of the individual rounds: $\mathrm{cor}(\Omega) = \prod_{i=0}^{r-1} \mathrm{cor}_{\text{round } i}(\omega_i, \omega_{i+1})$.

A key-alternating cipher is a common model for block ciphers where the round consists of a fixed part g followed by an addition with the round key. We can thus write for a key-alternating cipher:

$$\mathrm{cor}_{\text{round } i}(\omega_i, \omega_{i+1}) = (-1)^{\omega_i^t k_i} \mathrm{cor}_g(\omega_i, \omega_{i+1}). \tag{1.1}$$

Note, however, that this notation implicitly assumes that to each bit of the round input a different bit of the round key is added. We will show that this is not always the case in Section 2.1.2. We obtain for now:

$$\operatorname{cor}(\Omega) = \prod_i (-1)^{\omega_i^t k_i} \operatorname{cor}_g(\omega_i, \omega_{i+1}) = |\operatorname{cor}(\Omega)| \cdot (-1)^{d_\Omega + \sum_i \omega_i^t k_i}, \qquad (1.2)$$

with $d_\Omega = 1$ if $\prod_i \operatorname{cor}_g(\omega_i, \omega_{i+1})$ is negative; otherwise $d_\Omega = 0$.

A linear hull covering r rounds of a block cipher is a pair (α, β). The hull is composed of a set of linear trails all having the same input mask and output mask but that can differ in intermediate masks. The correlation of a linear hull is

$$\operatorname{cor}(\alpha, \beta) = \sum_{\substack{\Omega \\ \omega_0 = \alpha, \omega_r = \beta}} \operatorname{cor}(\Omega). \qquad (1.3)$$

Some works use the notion of *linear characteristic* to describe linear trails. The advantage of using this notion is that it emphasizes the similarities between differential cryptanalysis and linear cryptanalysis. For this reason exactly, the term *linear trail* is preferred in this Thesis. While the similarities between the two techniques are interesting, they often obfuscate important differences. For example, the probability of a differential can be lower bounded by a single differential characteristic. As we see in Section 2.2 no such equivalent exists for a linear hull. The term *linear path* is also sometimes used in the literature, but less frequently.

1.1.4 Rotational Cryptanalysis

Similar to differential cryptanalysis, rotational cryptanalysis [96] is a chosen-plaintext attack taking advantage of the propagation of a certain property. In differential cryptanalysis this property was the XOR difference of two plaintexts while in rotational cryptanalysis the property is the rotational difference.

Given a pair of plaintexts (x, x') we say that they form a *rotational pair* or that they *satisfy the rotational property with respect to* γ if $x' = x \lll \gamma$ where $\lll$ is a cyclic shift. Rotational cryptanalysis uses the known probability that an input rotational pair leads to an output rotational pair through what is known as "the ARX operations" (modular Addition, Rotation, and XOR). In other words, rotational cryptanalysis examines the evolution of the rotational property during the encryption of a rotational input pair. For the rotation and XOR operations, the probability that an input rotational pair leads to an output rotational pair is 1. For modular addition, the following proposition

provides a general way to compute the propagation probability of a rotational pair:

Proposition 1 ([61]). *For $x, y \in \mathbb{F}_{2^n}$, and $0 < \gamma < n$,*

$$\Pr[(x \boxplus y) \lll \gamma = (x \lll \gamma) \boxplus (y \lll \gamma)] = \left(1 + 2^{\gamma-n} + 2^{-\gamma} + 2^{-n}\right)/4.$$

If n is large and $\gamma = 1$ the probability is maximized and we get

$$\Pr[(x \boxplus y) \lll \gamma = (x \lll \gamma) \boxplus (y \lll \gamma)] = 2^{-1.415}.$$

Whenever the two inputs to the modular addition are independent and uniformly distributed, the probabilities of consecutive modular additions can be directly multiplied. However, as was shown in [97], if a modular chain exists, an adjustment is required to the formula, and the resulting probability is in fact smaller.[2] A similar effect was mentioned for linear cryptanalysis in [125], and for differential cryptanalysis in [160].

In this Thesis, rotational cryptanalysis is used as the stage on which RX-cryptanalysis is developed in Section 3.1. RX-cryptanalysis is then further used in Sections 4.2 and 5.3.

1.1.5 Self-similarity Cryptanalysis

We now turn our attention to self-similarity attacks. These attacks exploit similarities within the cipher structure. In this Thesis we use two self-similarity attacks: a reflection attack and a fixed-point attack. We leave the exact description of these attacks to Section 5.2 and give here just their essence. It is important to note that other types of self-similarity attacks exist (*e.g.*, slide attacks), as well as different variants of the two presented below.

The reflection attack uses the well known fact that in a Feistel structure, decryption can be obtained from the encryption function by changing the order of words. Hence, for a palindromic sequence of round keys, if the two halves of the message in the middle of the palindrome are equal, the two parts of the palindrome cancel each other.

The fixed-point attack is similar, but instead of using a palindrome it exploits a repeating sequence of subkeys. Suppose that the round keys for rounds 0–$(\ell-1)$ are the same as the keys for rounds ℓ–$(2\ell-1)$, and suppose that for some input S we have that $S \to S$ in rounds 0–$(\ell-1)$. Then, we also have $S \to S$ for

[2] A modular chain is a sequence of modular addition operations where the output of one is given directly as an input to the next. It is outside the scope of this Thesis.

rounds ℓ–$(2\ell - 1)$ and of course, due to the transitive property, also $S \to S$ for rounds 0–$(2\ell - 1)$.

1.1.6 Key Recovery

The end-goal of cryptanalysis is to recover the secret key used in a specific instance of the attacked algorithm. We now present several generic methods used for key recovery in statistical attacks (such as differential-, linear-, and rotational-cryptanalysis). The key recovery phase in self-similarity attacks is a more *ad hoc* procedure and we defer its description to Section 5.2 where it is explained as part of the various attacks.

After a non-ideal statistical property was identified (*e.g.*, a differential with high probability, a linear hull with large bias, *etc.*), the attack proceeds to the *data collection phase*. In the known-plaintext model, this phase consists of collecting pairs of plaintexts and their respective ciphertexts. For chosen-plaintext attacks, the adversary queries selected messages and collects their respective ciphertexts. Once enough data was collected, the adversary tries to detect the non-ideal property. If the property can be identified in the dataset, the cryptosystem is said to be *distinguishable* from a random oracle. In statistical attacks, this distinguishing is the basis for key recovery.

The amount of data required for the attack to be successful differs. For differentials and rotational characteristics with probability p the amount of required data is of the order of $\frac{1}{p}$. For a linear hull with correlation c the data requirement is usually $2^{\ell} \cdot c^{-2}$ for some small ℓ. Chapter 2 deals extensively with the data complexity of linear attacks.

Matsui's Algorithm 2 (r-round attack)

A differential r-round attack and a linear attack using Matsui's Algorithm 2 are two similar key recovery procedures for block ciphers. The idea behind both is, without loss of generality, to extend a q-round property by r additional rounds. This is done by collecting data encrypted for $q + r$ rounds of the block cipher. Once enough data was collected, the adversary guesses the part of the key required to remove the r rounds, leaving the inputs and outputs to the other q rounds exposed. The intuition behind this procedure is that when the right key is used to remove the r rounds, the q-round property can be detected. On the other hand, using a wrong key is akin to adding r rounds of encryption instead of removing them, and hence, using a wrong key would make the ciphertext

look more random (*i.e.*, as if it was encrypted using $q + r + r$ rounds), and the adversary will be unable to identify the q-round property.

In linear cryptanalysis this is often called *the wrong-key-randomization hypothesis* or simply *the wrong-key hypothesis*. The wrong-key hypothesis was first introduced by Harpes, Kramer and Massey in [79]. For quite some time, it was understood to mean that an attempt to decrypt the last r rounds would result in bias 0 for all wrong keys until Bogdanov and Tischhauser showed in [42] that even the bias of a random permutation is not necessarily 0. They corrected the wrong-key hypothesis to account for the bias distribution in random permutations using a compound model and derived the success probability and data complexity requirements of such an attack. Their work is extended in Section 2.3 for the case of distinct known-plaintexts, and the observation they made about the non-monotonicity of the success probability is reconsidered in Section 2.4.

Matsui's Algorithm 1

Matsui's Algorithm 1 is another key recovery algorithm that is unique to linear cryptanalysis. When Algorithm 1 is used, the sign of the bias is used to determine the value of the XOR of some key bits. The idea behind this algorithm is to find an affine relation between certain input bits, output bits, and key bits (*i.e.*, a relation of the form $\alpha^t \cdot x \oplus \beta^t \cdot F_k(x) \oplus \gamma^t \cdot k = 0$). The input and output bits allow the adversary to measure the bias of the trail, and the key bits determine its sign. By inverting the process, *i.e.*, writing the equation as $\alpha^t \cdot x \oplus \beta^t \cdot F_k(x) = \gamma^t \cdot k$, an adversary can determine the parity (of a linear combination) of the key bits selected by γ.

Algorithm 1 is considered to be inferior to Algorithm 2. The main reason is that it can only be used to recover a single key bit. In Section 2.2 we show how to extend Algorithm 1 to recover more key bits. However, as we see in Sections 2.1–2.2, Algorithm 1 suffers from further limitations and it is very tempting to use it incorrectly. This is why Algorithm 1 should be abandoned as a general method and only be used in certain well-understood scenarios.

Key ranking vs. hypothesis testing

Two approaches for key recovery are discussed in the literature for Matsui's Algorithm 2: key ranking and hypothesis testing. In the key ranking approach which was first described by Matsui in [110] candidate keys are ordered according to their likelihood and tried in this order. It is implicitly expected that the

right key will have a high absolute bias and will hence be at the beginning of the list (*i.e.*, amongst the most likely keys).[3] When hypothesis testing is used, the likelihood of each key is evaluated independently using a statistical test. Key ranking is used for the analyses of Sections 2.3–2.4; hypothesis testing is used for the new technique we develop in Section 3.2.

Measuring the adversary's advantage when using key ranking is easy:

Definition 1. *When key ranking is used, the adversary is said to gain an advantage of a bits if the right key is ranked amongst the highest* 2^{m-a} *keys. Otherwise, the attack is said to fail.*

However, for hypothesis testing, measuring the advantage is not as simple and in fact, different definitions exist in the literature. Instead of arbitrarily choosing one we will now explain how hypothesis testing is used in linear cryptanalysis in more detail.

Using hypothesis testing for linear cryptanalysis

Suppose π is a random permutation of n bits and ψ is a linear approximation. We denote by $\mathbf{T}$ a random variable for the number of times ψ is satisfied over all 2^n possible inputs to π. The behavior of $\mathbf{T}$ is given by the following Lemma:

Lemma 4. *For* π, ψ, *and* $\mathbf{T}$ *as before*

$$\mathbf{T} \sim \mathcal{N}\left(2^{n-1}, 2^{n-2}\right) \tag{1.4}$$

Proof. We define $T|\epsilon = 2^{n-1} + 2^n \cdot \epsilon$ to be the number of times the linear approximation was satisfied over all possible inputs once π was chosen uniformly from the set of all $2^n!$ possible random permutations. Fixing π also fixes the bias ϵ of ψ when it is applied to π. Daemen and Rijmen show in [60] that $\boldsymbol{\epsilon} \sim \mathcal{N}\left(0, 2^{n-2}\right)$. Substituting π's bias ϵ for $\boldsymbol{\epsilon}$ we obtain the compound random variable $\mathbf{T}$. □

The random variable $\mathbf{T}$ can be converted into a standard normal variable by setting $\frac{\mathbf{T}-2^{n-1}}{2^{n/2-1}}$. Furthermore, knowing that the square of a standard normal variable follows the χ_1^2 distribution, we obtain the following corollary:

Corollary 1. *Let* π, ψ, *and* $\mathbf{T}$ *as before, then*

$$\left(\frac{\mathbf{T} - 2^{n-1}}{2^{n/2-1}}\right)^2 \sim \chi_1^2 \tag{1.5}$$

[3] As explained in Section 2.4, this implicit assumption is the reason for the non-monotonicity of the success probability.

Let E be a family of block ciphers with block size n, and E_k a member of this family characterized by a key k. Further, let ψ be a linear approximation that is biased when applied to a member of E. We define a counter $\hat{T}$ to count the number of times ψ is satisfied after observing 2^n pairs of plaintexts and ciphertexts. As ψ is biased when applied to E_k, so is $\hat{T}$ and the value $\Pr[\mathbf{T} = \hat{T}]$ is small. On the other hand, when applied to a random permutation, $\hat{T}$ follows Lemma 4 and hence $\Pr[\mathbf{T} = \hat{T}]$ is large. The goal of the adversary upon receiving $\hat{T}$ is to decide if it was obtained by applying ψ to a member of E or to a random permutation.

In a classical linear attack, the adversary would normally use $\hat{T}$ to calculate a sample bias $\hat{\epsilon}$ and compare it to some threshold. For example, they may decide to use $\pm 2^{-n/2}$, meaning that the distinguishing algorithm returns "random permutation" if $-2^{-n/2} \leq \hat{\epsilon} \leq 2^{-n/2}$; "a member of E" otherwise.

This procedure can be viewed as a form of hypothesis testing. The null hypothesis H_0 is that $\hat{T}$ was generated through a random permutation. The alternative hypothesis, namely H_1, is that it was generated using a block cipher. The adversary constructs a confidence interval for H_0, and rejects the null hypothesis if the test statistic falls outside of it. According to [60], $\boldsymbol{\epsilon}$, the bias of a random permutation, has a normal distribution with mean 0 and standard deviation $2^{-n/2-1}$, which means that in the above example, $2^{-n/2}$ is 2 standard deviations away from the mean, yielding a confidence interval of size $\alpha = \Phi(2) - \Phi(-2) = 0.95449$. Using this interval means that the distinguisher reports $1-\alpha \approx 4.5\%$ of the values coming from a random permutation incorrectly, and thus, it gives an advantage when the probability to correctly identify the block cipher, namely β, is high enough such that $\beta - (1 - \alpha) > 0$.

Note that this procedure does not require much knowledge about the bias. It is sufficient that the underlying key-dependent bias be large enough, and that the number of samples (*i.e.*, the data complexity) be sufficiently large to provide a good estimator to the underlying bias.

1.2 Cryptosystems Mentioned in this Thesis

Throughout this Thesis we will use specific cryptosystems either as examples, test-beds, or simply to attack them and expose weaknesses. In this section we describe those.

1.2.1 Simon

SIMON is a family of lightweight block ciphers designed by the US National Security Agency (NSA) and published in 2013 [20]. The SIMON$2n/mn$ family of lightweight block ciphers has 10 members differing in their block and key sizes. All members of the family have a Feistel structure with round function R employing a non-linear function f. In each round i, R receives two n-bit input words X_i and Y_i, and outputs two n-bit words X_{i+1} and Y_{i+1}. The round function uses three operations: addition in $\mathrm{GF}(2)^n$ (exclusive-or; XOR), bitwise AND, and a left circular shift by j positions, which we denote by $\oplus, \&$, and $\lll j$, respectively. The internal non-linear function f is defined as:

$$f(X_i) = [(X_i \lll 1) \,\&\, (X_i \lll 8)] \oplus (X_i \lll 2).$$

The output of the round function R on an input block (X_i, Y_i) is:

$$R_i(X_i, Y_i) = (Y_i \oplus f(X_i) \oplus k_i, X_i),$$

where i is the round number and k_i is the round key. The entire cipher is a composition of round functions $R_{r-1} \circ R_{r-2} \circ \ldots \circ R_0(X_0, Y_0)$. The structure of the round function of SIMON is depicted in Figure 1.2.

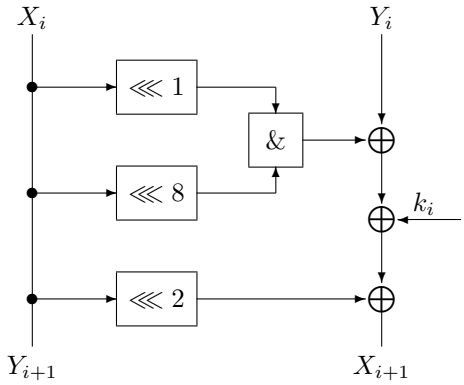


Figure 1.2: One round of SIMON (without the final swap operation)

1.2.2 Speck

SPECK is another family of lightweight block ciphers designed by the NSA [20]. The family includes 10 members, where each member is denoted by

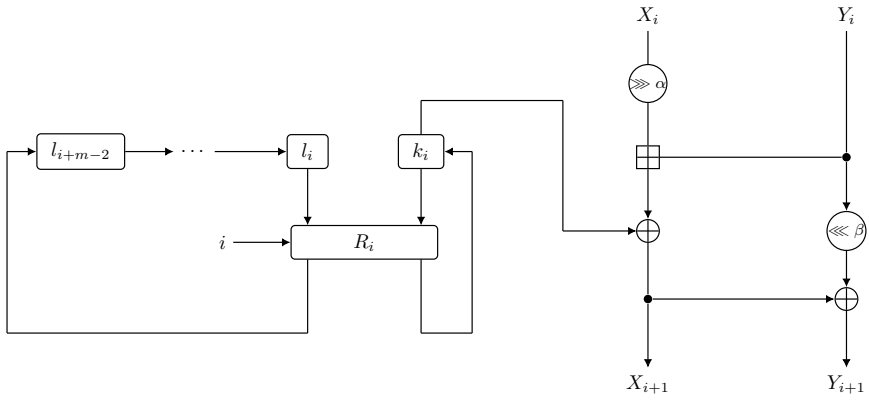


Figure 1.3: One round of SPECK

SPECK$2n/mn$, where the block size is $2n$ for $n \in \{16, 24, 32, 48, 64\}$, and the key size is mn for $m \in \{2, 3, 4\}$, depending on the desired security.

The round function of SPECK receives two words X_i and Y_i, and a round key k_i, all of size n, and outputs two words of size n, X_{i+1} and Y_{i+1}, such that

$$(X_{i+1}, Y_{i+1}) = R_{k_i}(X_i, Y_i) = (f_{k_i}(X_i, Y_i), f_{k_i}(X_i, Y_i) \oplus (Y_i \lll \beta)),$$

where $f_{k_i}(\cdot, \cdot)$ is

$$f_{k_i}(X_i, Y_i) = ((X_i \ggg \alpha) \boxplus Y_i) \oplus k_i.$$

The SPECK key schedule algorithm uses the same round function to generate the round keys. Let $K = (l_{m-2}, ..., l_0, k_0)$ be a master key for SPECK$2n$, where $l_i, k_0 \in \mathbb{F}_{2^n}$. The sequence of round keys k_i is generated as

$$k_{i+1} = f_{ct}(l_i, k_i) \oplus (k_i \lll \beta)$$

for

$$l_{i+m-1} = f_{ct}(l_i, k_i),$$

with $ct = i$ the round number starting from 0.

The rotation offsets (α, β) are $(7, 2)$ for SPECK32, and $(8, 3)$ for the larger versions. A single round of SPECK with $m = 4$ is depicted in Figure 1.3. For more details, we refer the interested reader to the original design in [20].

1.2.3 GOST and GOST2

Gost [140] is a block cipher designed during the 1970's by the Soviet Union as an alternative to the American DES [155]. Similarly to DES, it has a 64-bit Feistel structure, employing 8 S-boxes and is intended for civilian use. Unlike DES, it has a significantly larger key (256 bits instead of just 56), more rounds (32 compared with DES' 16), and it uses different sets of S-boxes. What is unique about Gost is that the S-boxes are not an integral part of the standard, and in fact, they were kept secret, which allowed the government to give different sets of S-boxes to different users.

The Gost block cipher has a 64-bit Feistel structure using a 256-bit key. The 64-bit block is treated as two words of 32-bit each which are referred to as the "left word" and the "right word". The state in round i is denoted by $S_i = (L_i, R_i)$ where L_i and R_i are the left and right words entering round i, respectively. In each round, a 64-bit to 32-bit non-linear function f is applied to the right word and the round's subkey K_i. The output of f is XORed to the left input word, and the words are swapped. This is repeated 32 times, for rounds numbered 0–31, and the output of the last round is used as the ciphertext. We get that

$$R_{i+1} = f\left(R_i, K_i\right) \oplus L_i$$

$$L_{i+1} = R_i\,,$$

where R_0 is the right half of the plaintext, L_0 is the left half of it, and K_i is the i-th round subkey.

Inside the non-linear function, the input is mixed with the round's 32-bit subkey K_i using modular addition. Then, it is split into 8 chunks of 4 bits entering the eight S-boxes. Finally, the output of the S-boxes is left rotated by 11 bits. A schematic view of 1-round Gost is depicted in Figure 1.4.

After the USSR had been dissolved, Gost was accepted as a Russian standard in [140] and was proposed to be included in ISO/IEC 18033-3 [82]. At the time Gost seemed like a natural candidate to be included in the standard. From a security point of view, although there have been several attacks such as [100] in the related-key model, the only attack on the full Gost in the single-key model was published in [94], and was limited to a weak-key class.

However, as a result of the renewed interest due to the standardization process, Isobe presented in [83] an improvement to [94] that eliminates the weak-key assumption resulting in an attack with time complexity of 2^{224}. A year later, as the attack was improved by Dinur, Dunkelman, and Shamir in [68], and as new attacks were presented by Courtois in [56], the idea to standardize Gost was rejected.

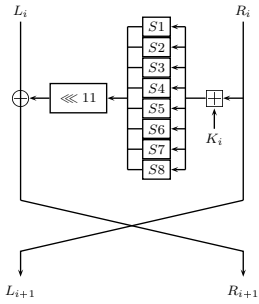


Figure 1.4: One round of GOST/GOST2

In 2015, authors from the Russian Technical Committee for Standardization (TC 26) published a modified version of the GOST cipher that was claimed to resist previous attacks [69]. The modified version differs from the original GOST in only two aspects: (i) it has a different key schedule, designed to avoid previous attacks and, (ii) it makes an explicit choice for the S-boxes.

In both versions, the key schedule takes the 256-bit key K and splits it into 8 subkeys of 32 bits denoted K^0 to K^7. In the original GOST, the first 24 rounds used the subkeys in their cyclic order (*i.e.*, $K_0 = K_8 = K_{16} = K^0$, $K_1 = K_9 = K_{17} = K^1$, *etc.*). In the final 8 rounds (*i.e.*, rounds 24–31), the subkeys were used in a reverse order such that K^7 was used in round 24, K^6 was used in round 25, *etc.* In the modified version, the key schedule was changed, but the keys are still used in an ascending cyclic order in rounds 0–7, 8–15, and 16–23, and in a descending cyclic order in rounds 24–31. The order of the subkeys for both versions is presented in Table 1.1. From here on, we refer to the modified version of GOST presented in [69] as GOST2. Also, whenever we want to stress that K^j is used in round i (*i.e.*, $K^j = K_i$), we write it as K_i^j.

Table 1.1: The order of subkeys in GOST and GOST2

Round	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Subkey (GOST)	K^0	K^1	K^2	K^3	K^4	K^5	K^6	K^7	K^0	K^1	K^2	K^3	K^4	K^5	K^6	K^7
Subkey (GOST2)	K^0	K^1	K^2	K^3	K^4	K^5	K^6	K^7	K^3	K^4	K^5	K^6	K^7	K^0	K^1	K^2
Round	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Subkey (GOST)	K^0	K^1	K^2	K^3	K^4	K^5	K^6	K^7	K^7	K^6	K^5	K^4	K^3	K^2	K^1	K^0
Subkey (GOST2)	K^5	K^6	K^7	K^0	K^1	K^2	K^3	K^4	K^6	K^5	K^4	K^3	K^2	K^1	K^0	K^7

Another change made to the design of GOST is the proposal of concrete S-boxes. The designers suggested to use the same permutation Π_1 for the first 4 S-boxes, and another permutation Π_2 for the other 4 S-boxes. Both permutations are presented in Table 1.2. We refer the interested reader to [69] for the rationale behind the choice of S-boxes.

Table 1.2: The proposed S-boxes for GOST2

Input	0_x	1_x	2_x	3_x	4_x	5_x	6_x	7_x	8_x	9_x	A_x	B_x	C_x	D_x	E_x	F_x
Π_1	6_x	A_x	F_x	4_x	3_x	8_x	5_x	0_x	D_x	E_x	7_x	1_x	2_x	B_x	C_x	9_x
Π_2	E_x	0_x	8_x	1_x	7_x	A_x	5_x	6_x	D_x	2_x	4_x	9_x	3_x	F_x	C_x	B_x

1.2.4 P-OMD

OFFSET MERKLE-DAMGÅRD (OMD) is a submission to the CAESAR competition [46] by Cogliani *et al.* [53]. OMD is an *Authenticated Encryption with Associated Data (AEAD)* mode of operation taking as input a key k, a nonce N, a message M, and associated data A, and uses a keyed compression function[4] F to produce a ciphertext C, which is the encryption of M, and an authentication tag T which allows to authenticate (*i.e.*, verify the correctness of) both M and A. OMD is proven to be secure as long as the adversary is not allowed to use the same nonce more than once, and as long as some restrictions about the number of messages processed using the same key are met.

P-OMD is an extension by Reyhanitabar, Vaudenay, and Vizár [134], which improves over OMD in that the associated data is processed almost for free. The designers prove that P-OMD inherits all security features of OMD. As a bonus, they claim that P-OMD maintains its security even if the adversary is allowed to use the same nonce more than once.

Formally, let $k, m, n, \tau \in \mathbb{N}$ such that $m \leq n$ and let $F : \{0,1\}^k \times \{0,1\}^{n+m} \to \{0,1\}^n$ be a keyed compression function. P-OMD is a mapping that takes as input a key $K \in \{0,1\}^k$, a nonce $N \in \{0,1\}^{\leq n-1}$, an arbitrarily sized associated data $A \in \{0,1\}^*$, and an arbitrarily sized message $M \in \{0,1\}^*$, and returns a ciphertext $C \in \{0,1\}^{|M|}$ and an authentication tag $T \in \{0,1\}^\tau$.

For the sake of brevity we describe P-OMD only for the specific case where $|A| = 2n$ and $|M| = m$ (or in other words, the associated data consists of two integral blocks and the message of one integral block). It is depicted in

[4] A keyed compression function is a keyed Boolean vector function mapping inputs of size $n + m$ to outputs of size n.

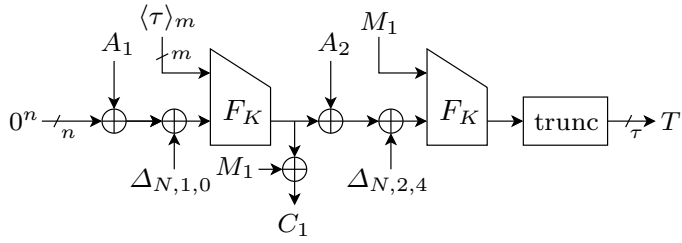


Figure 1.5: P-OMD for the specific case of $|A| = 2n$ and $|M| = m$.

Figure 1.5 (and corresponds to Case A of [135]). Here,

$$\Delta_{N,1,0} = F_K\left(N\|10^{n-1-|N|}, 0^m\right) \oplus 16F_K\left(0^n, 0^m\right) ,$$

$$\Delta_{N,2,4} = F_K\left(N\|10^{n-1-|N|}, 0^m\right) \oplus (32 \oplus 16 \oplus 4)\, F_K\left(0^n, 0^m\right) ,$$

1.2.5 GCM

In this section we describe the GCM mode of operation [111, 112] with 128-bit nonces. We let $E : \{0,1\}^{128} \times \{0,1\}^{128} \to \{0,1\}^{128}$ denote a 128-bit block cipher. The function GHASH is defined in Algorithm 1.1. Algorithm 1.3 provides pseudocode for GCM encryption, which also uses the keystream generator CTR mode from Algorithm 1.2. See Figure 1.6 for an illustration.

Algorithm 1.1 $\mathsf{GHASH}_L\,(A, C)$

Input: $L \in \{0,1\}^n$, $A \in \{0,1\}^{\leq n\left(2^{n/2}-1\right)}$, $C \in \{0,1\}^{\leq n\left(2^{n/2}-1\right)}$
Output: $Y \in \{0,1\}^n$

1: $X \leftarrow A0^{*n} \parallel C0^{*n} \parallel \mathsf{str}_{n/2}\,(|A|) \parallel \mathsf{str}_{n/2}\,(|C|)$
2: $X[1]X[2]\cdots X[x] \xleftarrow{n} X$
3: $Y \leftarrow 0^n$
4: **for** $j = 1$ to x **do**
5: $\quad Y \leftarrow L \cdot (Y \oplus X[j])$
6: **end for**
7: Return Y

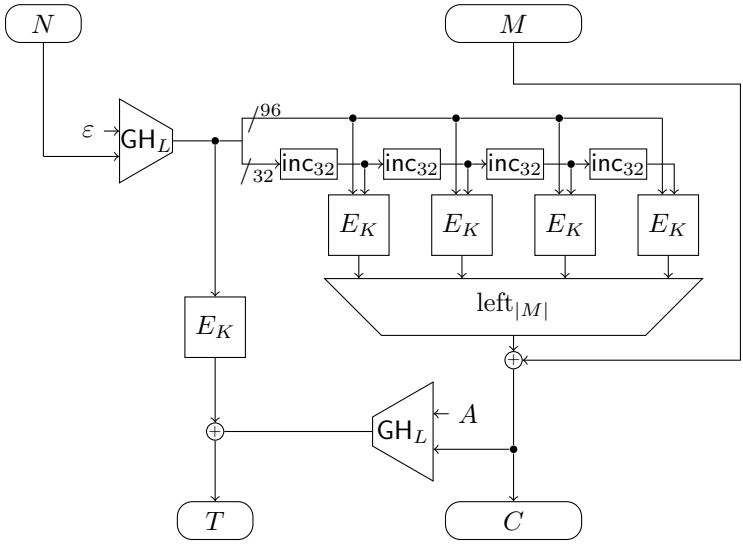


Figure 1.6: The GCM mode of operation with 128-bit nonces. GH is GHASH and $/_m$ indicates the number of bits on a wire. The value L is $E_K\,(0)$ and ϵ is the empty string.

Algorithm 1.2 $\mathsf{CTR}[F](X, m)$

Input: $F : \{0,1\}^x \to \{0,1\}^n, X \in \{0,1\}^x, m \in \mathbb{N}$
Output: $S \in \{0,1\}^{mn}$

1: $I \leftarrow X$
2: **for** $j = 1$ to m **do**
3: $\quad S[j] \leftarrow F(I)$
4: $\quad I \leftarrow \mathsf{inc}_x(I)$
5: **end for**
6: $S \leftarrow S[1]S[2] \cdots S[m]$
7: Return S

Algorithm 1.3 $\mathsf{GCM}_K(N, A, M)$

Input: $K \in \{0,1\}^{128}$, $N \in \{0,1\}^{128}$, $A \in \{0,1\}^{\leq 128 \cdot 2^{32}}$, $M \in \{0,1\}^{\leq 128 \cdot 2^{32}}$
Output: $(C, T) \in \{0,1\}^{\leq 128 \cdot 2^{32}} \times \{0,1\}^{128}$

1: $L \leftarrow E_K(\mathsf{str}_{128}(0))$
2: $I \leftarrow \mathsf{GHASH}_L(\varepsilon, N)$
3: $m \leftarrow |M|_{128}$
4: $F \leftarrow E_K(\mathrm{left}_{96}(I) \parallel \cdot)$
5: $C \leftarrow M \oplus \mathrm{left}_{|M|}(\mathsf{CTR}[F](\mathsf{inc}_{32}(\mathrm{right}_{32}(I)), m))$
6: $T \leftarrow E_K(I) \oplus \mathsf{GHASH}_L(A, C)$
7: Return (C, T)

Chapter 2

Revisiting the Theory of Cryptanalysis

> *"At any given moment there is an orthodoxy, a body of ideas which it is assumed that all right-thinking people will accept without question. It is not exactly forbidden to say this, that or the other, but it is 'not done' to say it, just as in mid-Victorian times it was 'not done' to mention trousers in the presence of a lady."*
>
> - George Orwell, *Animal Farm*

Cryptanalysis is the scientific field of investigating the security of cryptosystems. This investigation has two facets: theory and application. The current chapter and the next one both deal with the theory part. In this chapter, we identify and correct errors in the existing theory of cryptanalysis. In the next chapter we go beyond current theory by extending existing attacks and proposing new ones.

Sections 2.1–2.2 both deal with trail biases. In Section 2.1 we demonstrate the existence of 1-round linear hulls in linear cryptanalysis. We explain how and why they form, what effect they have on a linear attack, and introduce a correction to the way the Expected Linear Probability (ELP) is computed. In Section 2.2 we revisit Matsui's Algorithm 1 and show the importance of properly estimating the hull bias in a linear attack. We show that when a trail bias is taken as an approximation of a hull bias, the success probability of a linear attack may be greatly affected. We conclude this section by showing that

when the bias is properly accounted for, Matsui's Algorithm 1 can be extended to recover more than a single key bit. Both sections were published as a single paper in [16] together with Vincent Rijmen. Only those parts from [16] where this Author was a main contributor were included in this Thesis.

In Sections 2.3–2.4 we revisit a fundamental assumption in linear cryptanalysis, namely the *wrong-key-randomization hypothesis*. The wrong-key-randomization hypothesis deals with the behavior of the bias of wrong keys in Matsui's Algorithm 2. In Section 2.3 we show that previous work assumed that data sampling in linear cryptanalysis is done with replacement and develop a new statistical model for attacks using distinct known-plaintexts. We derive formulae for the data complexity and the success probability of an attack using this model. Our formula for the success probability shows once again the possibility that increasing the data complexity may lead to a decrease in the success probability. In Section 2.4 we explain this phenomenon and derive the necessary conditions for its occurrence. Both sections were merged into a single paper [7] and submitted to J. of Cryptology. The paper was co-authored with Tim Beyne and Vincent Rijmen. This Author was a main contributor to this work.

Finally, in Section 2.5 a few misconceptions about linear cryptanalysis are discussed.

2.1 On the Existence of 1-Round Linear Hulls

In this section we show the existence of 1-round linear hulls. The phenomenon is presented and its implications on bias estimation is investigated. Finally, a correction for the way ELP should be computed in this case is proposed.

In the rest of the section, the notation (a, b, c, d, e) is used to describe a 1-round linear trail for Simon (*cf.* Figure 2.1). Here a and b denote the left and right input masks; c and d denote the masks at the outputs of the two topmost rotations; e and b denote the left and right output masks (before the swap operation).

We now study the behavior of 1-round linear trails for Simon using the rules of propagation of linear trails introduced in [28, 48]. The rule for trail propagation over the branch operation implies the following constraint on a, b, c, d, e:

$$a \oplus e = (b \ggg 2) \oplus (c \ggg 1) \oplus (d \ggg 8). \tag{2.1}$$

The rule for trail propagation over the XOR operation is already implicit in the way we propagate the b mask through Figure 2.1.

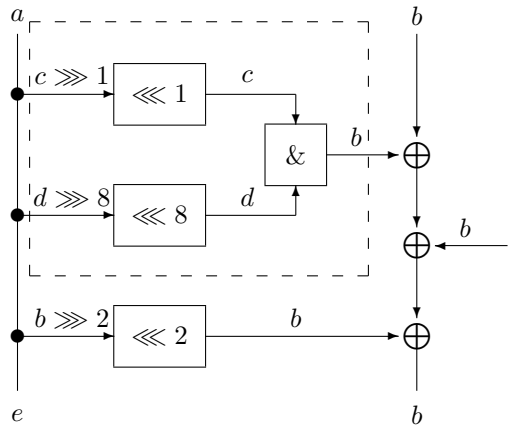


Figure 2.1: A 1-round linear trail for SIMON (without the final swap operation). The dashed box indicates the part of the round that we discuss in this section.

The output bit z of a bitwise AND operation $z = x \cdot y$ is correlated to the 4 linear functions of the two input bits:

$$\mathrm{cor}(z, 0) = \mathrm{cor}(z, x) = \mathrm{cor}(z, y) = 1/2,\ \mathrm{cor}(z, x \oplus y) = -1/2.$$

It follows that the AND operation in SIMON leads to the following constraints on b, c, d: if a bit in c or d is set, then the bit in b at the corresponding position needs to be set. This translates to:

$$\bar{c} \text{ OR } b = 1 \tag{2.2}$$

$$\bar{d} \text{ OR } b = 1 \tag{2.3}$$

The following lemma expresses that some 1-round trails come in groups.

Lemma 5. *Let (a, b, c, d, e) be a 1-round trail over* SIMON. *If there exists an index i such that $b_i = b_{i+7} = 1$, then the trail (a, b, c, d, e) satisfies the Constraints* (2.1)–(2.3) *if and only if the trail $(a, b, c \oplus (1 \lll i), d \oplus (1 \lll (i + 7)), e)$ satisfies the Constraints* (2.1)–(2.3).

Proof. For Constraint (2.1) we have:

$$\begin{aligned}((c \oplus (1 \lll i)) \ggg 1) \oplus ((d \oplus (1 \lll (i+7))) \ggg 8) \\ &= (c \ggg 1) \oplus (1 \lll (i-1)) \oplus (d \ggg 8) \oplus (1 \lll (i+7-8)) \\ &= (c \ggg 1) \oplus (d \ggg 8) \oplus ((1 \lll (i-1)) \oplus (1 \lll (i-1))) \\ &= (c \ggg 1) \oplus (d \ggg 8).\end{aligned}$$

Hence, either both trails satisfy Constraint (2.1) or neither does.

For Constraint (2.2) we see that if bit i of b is set, then the value of c at position i does not matter. Hence, either both c and $c \oplus (1 \lll i)$ satisfy Constraint (2.2), or they both don't satisfy it. Similarly for Constraint (2.3) and $d \oplus (1 \lll (i+7))$. □

Since the trails in Lemma 5 have the same input mask (a, b) and the same output mask (e, b), they form a 1-round linear hull. Each of the Figures 2.2–2.4 shows two trails derived from one another by means of Lemma 5. Notice that in each set both trails select exactly the same bits of the round keys.

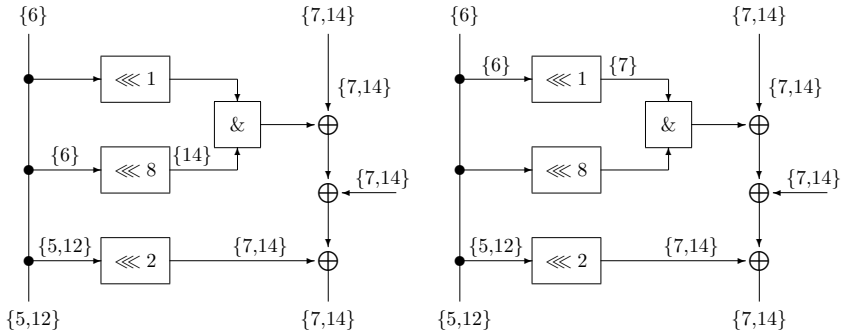


Figure 2.2: Two trails of a 1-round linear hull

2.1.1 Correlations and Correlation Contributions

We now want to compute the correlation contributions of the trails in Figures 2.2–2.4. The usual rule is to assume that all nonlinear functions act independently

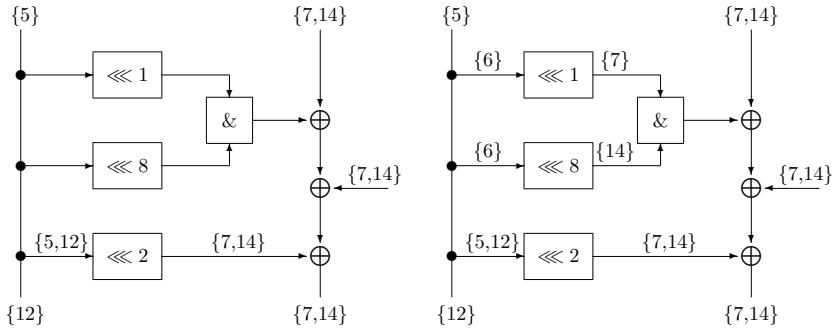


Figure 2.3: Two trails of a second 1-round linear hull

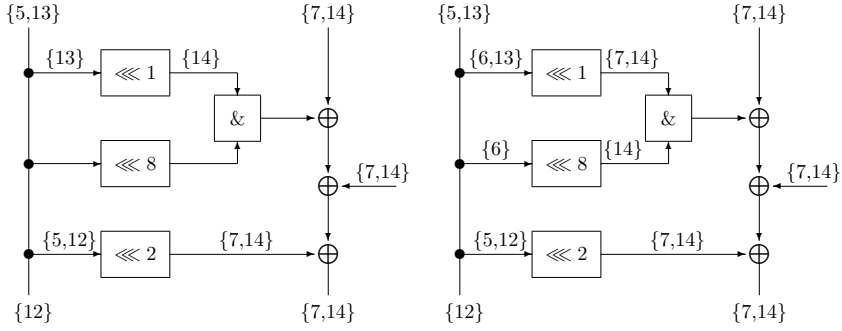


Figure 2.4: Two trails of a third 1-round hull. The trails have nonzero contributions of the same magnitude and opposite sign. The hull has correlation zero despite being composed of two trails with non-zero correlation.

and to multiply their correlations. This results in the following values for the correlation contributions of the six trails:

	c	d	cor
Figure 2.2	$\varnothing$	$\{14\}$	2^{-2}
	$\{7\}$	$\varnothing$	2^{-2}
Figure 2.3	$\varnothing$	$\varnothing$	2^{-2}
	$\{7\}$	$\{14\}$	2^{-2}
Figure 2.4	$\{14\}$	$\varnothing$	2^{-2}
	$\{7,14\}$	$\{14\}$	-2^{-2}

In each case by adding the correlation contributions of the two trails we obtain the correct correlation of the hull. However, starting from the observation that when $b_i = b_{i+7} = 1$, there are pairs of AND gates that share input bits, we can follow a different approach. Let

$$s = y \cdot x, \; t = y \cdot z,$$

then we have

$$s \oplus t = y \cdot (x \oplus z),$$

which implies the following correlations:

$$\mathrm{cor}(s \oplus t, 0) = \mathrm{cor}(s \oplus t, y) = \mathrm{cor}(s \oplus t, x \oplus z) = 1/2$$

$$\mathrm{cor}(s \oplus t, y \oplus x \oplus z) = -1/2$$

$$\mathrm{cor}(s \oplus t, x \oplus y) = \mathrm{cor}(s \oplus t, y \oplus z) = 0$$

$$\mathrm{cor}(s \oplus t, x) = \mathrm{cor}(s \oplus t, z) = 0.$$

These values can be used to derive immediately the exact correlations of the linear hulls of Figures 2.2–2.4. Observe that the linear hull of Figure 2.4 has correlation zero, while both trails have nonzero correlation contributions. Hence, mounting an attack and using the correlation contribution of one of the trails from Figure 2.4 as an estimate for the correlation of this linear hull will likely lead to wrong results. Likewise, using a single trail from Figures 2.2–2.3 would lead to a higher data complexity than what is actually needed. This phenomenon is further explained in Section 2.2.

2.1.2 Expected Correlation and Potential

Several recent works provide bounds for the security of ciphers defined at bit-level against linear cryptanalysis by bounding the potential of linear hulls [148–150,152]. The potentials of the hulls are computed by summing the squares of the expected values of the correlation contributions of the linear trails, which are constructed automatically using *e.g.*, mixed-integer linear programming (MILP) techniques. Several of these works mention the problem that may arise in the computation of the correlation contribution of a linear trail when non-linear functions share inputs.

We now address a second problem with the computation of the potential. Note that this problem does not occur for differential characteristics and differentials. It is one reason why we do not agree that differential characteristics and linear trails can be treated in exactly the same manner, as is sometimes claimed in existing literature (*e.g.*, [99, Sect. 7.3.2]).

Expected correlation

For a key-alternating cipher, the expected value (over all round keys) of the correlation contribution of a linear trail equals

$$\mathrm{E}[\mathrm{cor}(\Omega)] = 0 \ . \tag{2.4}$$

This follows directly by taking the expectation of (1.2). Intuitively, (2.4) may look contradictory to Matsui's Algorithm 1 in [110]. The apparent contradiction can be solved as follows. Although [58] writes:

> *The multiple-round linear expressions described in [110] correspond with what we call linear trails.*

there is in fact a difference. The approximations of [110] are linear expressions in terms of plaintext bits, ciphertext bits and round key bits. In the trails of [58], the round key bits are left out of the expression. It follows that the expected value of the correlation contribution becomes zero. By (1.3) we obtain that the expected value over all round keys of the correlation of a linear hull is

$$\mathrm{E}[\mathrm{cor}(a,b)] = 0.$$

Potential

Since the data complexity of a linear attack is inversely proportional to the square of the hull correlation, it is of importance to know or to bound the value $\mathrm{E}[(\mathrm{cor}(a,b))^2]$. In [123], Nyberg calls this quantity the *potential* of the linear hull, and gives the following formula to compute it:

$$\mathrm{E}[(\mathrm{cor}(a,b))^2] = \sum_{\substack{\Omega \\ \omega_0=a,\omega_r=b}} (\mathrm{cor_p}(\Omega))^2. \tag{2.5}$$

The potential is also called the Expected Linear Probability (ELP). We briefly recall here the proof for (2.5), using our own notation. By definition of expectations we have:

$$\mathrm{E}[(\mathrm{cor}(a,b))^2] = \frac{1}{K}\sum_k \left(\sum_{\substack{\Omega \\ \omega_0=a,\omega_r=b}} \mathrm{cor_p}(\Omega) \right) \left(\sum_{\substack{\Omega' \\ \omega'_0=a,\omega'_r=b}} \mathrm{cor_p}(\Omega') \right).$$

Using (1.2):

$$= \frac{1}{K} \sum_{\Omega} \sum_{\Omega'} \left(|\mathrm{cor}(\Omega)||\mathrm{cor}(\Omega')| \cdot \sum_{k} (-1)^{d_\Omega \oplus d_{\Omega'} \oplus \sum_i (\omega_i \oplus \omega_i')^t k_i} \right)$$

since

$$\sum_{k} (-1)^{\sum_i (\omega_i \oplus \omega_i')^t k_i} = \begin{cases} K & \text{if } \omega_i = \omega_i', \ \forall i, \\ 0 & \text{otherwise,} \end{cases} \tag{2.6}$$

we have:

$$\mathrm{E}[(\mathrm{cor}(a,b))^2] = \sum_{\Omega} (\mathrm{cor_p}(\Omega))^2 \ . \tag{2.7}$$

□

Additions/corrections

We will now show that if a cipher exhibits 1-round hulls, Formula (2.7) is no longer correct. The existence of 1-round hulls implies that we can have more than a single trail corresponding to the same linear mask for the round key. For example, each of the Figures 2.2–Figure 2.4 shows two different trails corresponding to the same linear mask of the round key.

In order to explain the consequences, (1.1) has to be slightly rewritten using a different notation. In fact, we need to distinguish between *trails* and *masks for the round key*. From now on, we use κ_i to denote the mask for the round key of round i, and $\mathcal{K}$ to denote the vector of round key masks. We use W to denote the vector of the data masks required to uniquely define the trail: $W = (w_0, w_1, \ldots, w_r)$. Note that the domain of the w_i may be larger than the domain of the κ_i. For example, in Figure 2.1, the data mask w_i contains a, b, c and d, while the round key mask κ_i needs to contain only b.

We denote by l, respectively L, the functions mapping w_i to the corresponding κ_i, respectively W to the corresponding $\mathcal{K}$. These functions are specific to the cipher. With this notation, (1.1) becomes:

$$\mathrm{cor}_{\text{round } i}(w_i, w_{i+1}) = (-1)^{\kappa_i^t k_i} \mathrm{cor}_g(w_i, w_{i+1}), \text{ with } \kappa_i = l(w_i).$$

When L is one-to-one, Formula (2.7) applies without modifications. However, if L is a non-injective map, then the sum of (2.6) becomes nonzero as soon as $\mathcal{K} = \mathcal{K}'$, which still allows $W \neq W'$. Hence (2.7) becomes:

$$\mathrm{E}[(\mathrm{cor}(a,b))^2] = \sum_{\mathcal{K}} \sum_{\substack{W,W' \\ L(W)=L(W')=\mathcal{K}}} (\mathrm{cor_p}(W) \cdot \mathrm{cor_p}(W')) \ .$$

Converting back, we obtain:

$$\mathrm{E}[(\mathrm{cor}(a,b))^2] = \sum_{\mathcal{K}} \left(\sum_{\substack{W \\ L(W)=\mathcal{K}}} \mathrm{cor}_{\mathrm{p}}(W) \right)^2 . \tag{2.8}$$

Comparing (2.5) to (2.8), we see that the difference between the two values can take positive as well as negative values. In particular when there are several trails with correlation contributions of comparable magnitude, the difference can be significant. Applied to the 1-round hulls of Figures 2.2–2.4, we get the following results:

$(\{a;b\},\{e;b\})$,	$\mathrm{E}[(\mathrm{cor}(a,b))^2]$ with (2.5)	$\mathrm{E}[(\mathrm{cor}(a,b))^2]$ with (2.8)
$(\{6;7,14\},\{5,12;7,14\})$	2^{-3}	2^{-2}
$(\{5;7,14\},\{12;7,14\})$	2^{-3}	2^{-2}
$(\{5,13;7,14\},\{12;7,14\})$	2^{-3}	0

We confirmed the values in the rightmost column experimentally.

2.2 On Matsui's Algorithm 1

In this section, we investigate how the success probability of Matsui's Algorithm 1 is influenced by all the trails in the same linear hull. As described already in [137], this can be used to extend Matsui's Algorithm 1 and to extract multiple key bits. We illustrate this for a reduced round version of Simon32 in Section 2.2.5.

In Sections 2.2.3–2.2.4 we study another consequence of this phenomenon: sometimes, the success rate of Matsui's Algorithm 1 will be worse than the estimate based on the study of a single trail. Somewhat counter-intuitively, the success rate of an attack may even decrease when the number of known-plaintexts is increased. But first, we describe the background for this phenomenon using an example: the 4 trails that constitute a hull over three rounds of Simon (Section 2.2.1) and the key-dependency of their correlations (Section 2.2.2).

2.2.1 Four Trails Through Three Rounds of Simon

Figures 2.5–2.6 show four trails through three rounds of Simon32. All four trails start from plaintext bits $\{(8,10);(0,12)\}$ and end in the ciphertext bit $\{16\}$. Hence they belong to the same 3-round linear hull. Observe that these 4

linear trails are linearly dependent. Denoting the vector of round key masks of Trail i by Ω_i, we have

$$\Omega_1 + \Omega_2 + \Omega_3 + \Omega_4 = 0.$$

All trails involve bits $\{0, 12\}$ from the first round key, bit $\{14\}$ from the second, and bit $\{0\}$ from the third round key. Additionally, each of these trails have the following key bits involved in the second round:

Trail 1:	$\emptyset$
Trail 2:	bit 8
Trail 3:	bit 15
Trail 4:	bits $8, 15$

We denote by Z the sum of the round key bits involved in all trails. The sum of the round key bits involved in Trails 2, 3 and 4, we denote respectively by $Z + z_0$, $Z + z_1$ and $Z + z_0 + z_1$.

2.2.2 Correlation Contributions of the Trails

Straightforward computations similar to the one in Section 2.1.1 show that the trails have the following correlation contributions:

$$\begin{array}{ll}
\text{Trail 1: } \mathrm{cor}^{(1)} = (-1)^{Z} & \cdot 2^{-4} \\
\text{Trail 2: } \mathrm{cor}^{(2)} = (-1)^{Z+z_0} & \cdot 2^{-5} \\
\text{Trail 3: } \mathrm{cor}^{(3)} = (-1)^{Z+z_1+1} & \cdot 2^{-5} \\
\text{Trail 4: } \mathrm{cor}^{(4)} = (-1)^{Z+z_0+z_1} & \cdot 2^{-5}
\end{array}$$

Note that these correlation contributions exist only as intermediate mathematical results. An attacker who can observe only plaintext and ciphertext bits, can measure only the sum of the four correlation contributions, *i.e.* the correlation of the hull (*cf.* [119]).

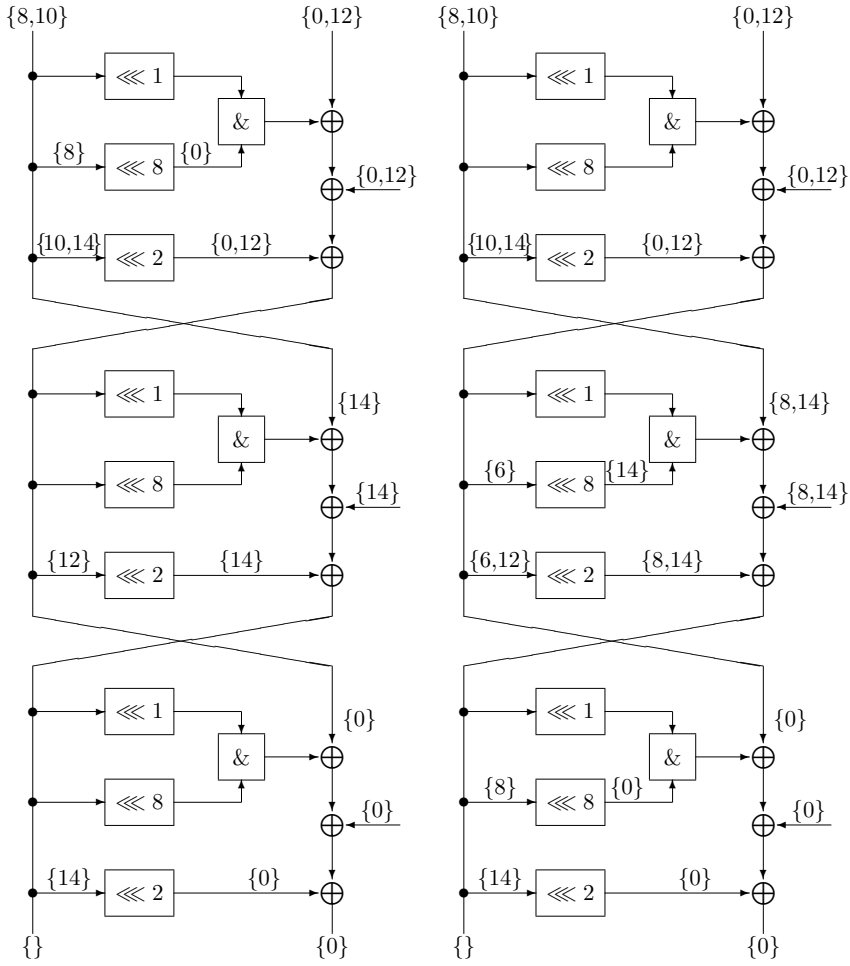


Figure 2.5: Two trails in a 3-round linear hull. Trail 1 is shown on the left, Trail 2 on the right.

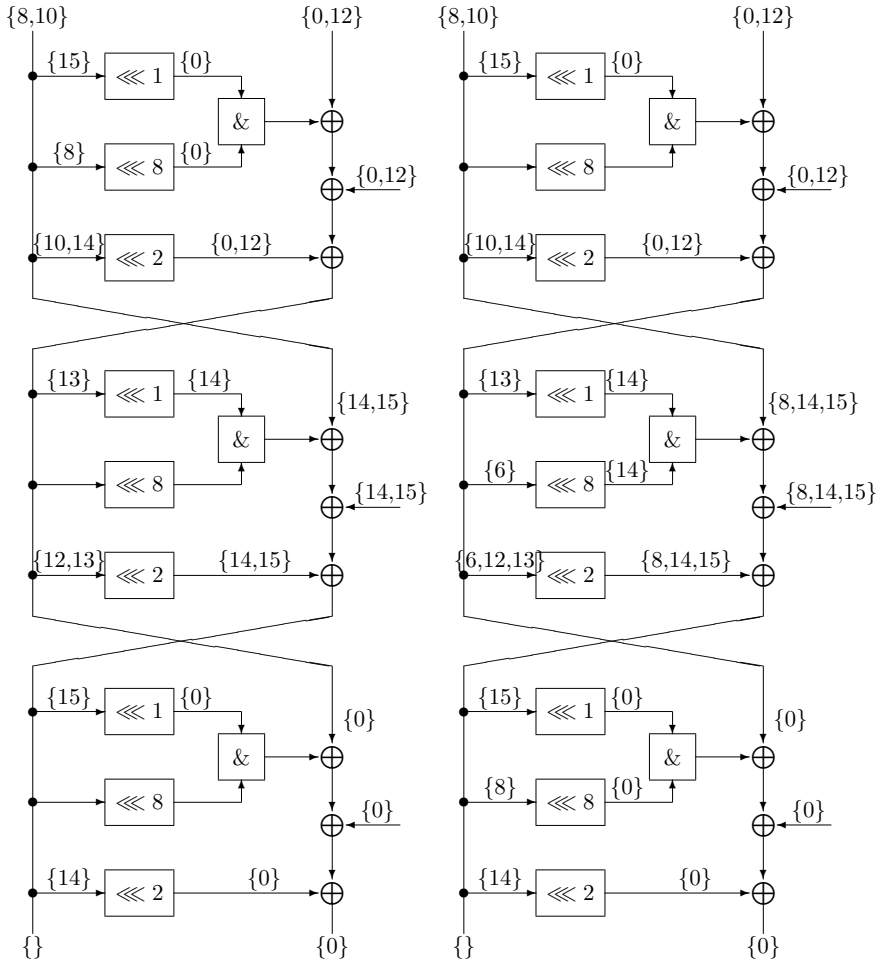


Figure 2.6: Two more trails in the same 3-round linear hull as Figure 2.5. Trail 3 is shown on the left, Trail 4 on the right.

We denote the correlation of the hull by cor_h and obtain:

$$\mathrm{cor}_h = (-1)^{Z} \cdot 2^{-4} + (-1)^{Z+z_0} \cdot 2^{-5} + (-1)^{Z+z_1+1} \cdot 2^{-5} + (-1)^{Z+z_0+z_1} \cdot 2^{-5} \tag{2.9}$$

$$= (-1)^{Z} \cdot 2^{-5} \left(2 + (-1)^{z_0} + (-1)^{z_1+1} + (-1)^{z_0+z_1}\right) \tag{2.10}$$

$$= (-1)^{Z+z_0} \cdot 2^{-5} \left((-1)^{z_0} \cdot 2 + 1 + (-1)^{z_1+z_0+1} + (-1)^{z_1}\right) \tag{2.11}$$

$$= (-1)^{Z+z_1} \cdot 2^{-5} \left((-1)^{z_1} \cdot 2 + (-1)^{z_0+z_1} - 1 + (-1)^{z_0}\right) \tag{2.12}$$

$$= (-1)^{Z+z_0+z_1} \cdot 2^{-5} \left((-1)^{z_0+z_1} \cdot 2 + (-1)^{z_1} + (-1)^{z_0+1} + 1\right). \tag{2.13}$$

From (2.9) we see that the correlation is determined by the values of $Z, Z + z_0, Z + z_1 + 1$, and $Z + z_0 + z_1$. Table 2.1 considers the 8 possible assignments for these variables and their correlations. We see that for a fixed Z, the value $(-1)^{Z} \cdot 3 \cdot 2^{-5}$ is three times more likely to occur than the value $(-1)^{Z+1} \cdot 2^{-5}$. In the following, we will investigate how likely each value is, and show how different values affect the success rate of Matsui's Algorithm 1 when different trails are considered as if they were the only trails.

Table 2.1: The possible values for cor_h obtained from (2.9)

Z	z_0	z_1	cor_h
0	0	0	$3 \cdot 2^{-5}$
0	0	1	$3 \cdot 2^{-5}$
0	1	0	-2^{-5}
0	1	1	$3 \cdot 2^{-5}$
1	0	0	$-3 \cdot 2^{-5}$
1	0	1	$-3 \cdot 2^{-5}$
1	1	0	2^{-5}
1	1	1	$-3 \cdot 2^{-5}$

We adopt the figures of [110, Table 2] to express the relation between correlation of a hull, the number of known-plaintexts and the success rate. Concretely, we derive from the table that if the hull has correlation c, then using c^{-2}, $4c^{-2}$ and $8c^{-2}$ known-plaintexts, the algorithm achieves respective success rates of 84%, 98% and 100%.

2.2.3 Knowing Trail 1 Only

In order to apply Algorithm 1 using Trail 1, the adversary first computes the correlation contribution of Trail 1:

$$c = \mathrm{cor}^{(1)} = (-1)^Z \cdot 2^{-4}\,. \tag{2.14}$$

Using the assumption that the correlation of the hull is approximately equal to the correlation contribution of Trail 1, the adversary concludes that a sample of $N = 4 \cdot c^{-2} = 2^{10}$ known-plaintexts should be sufficient to estimate Z with a success rate of 98%.

Subsequently, the adversary collects a sample of N known-plaintexts and uses them to compute the sample correlation $\hat{c}$. Depending on the value of $\hat{c}$, the adversary "guesses" a value for the sum (XOR) of the round key bits associated with the trail. Using (2.14) the adversary is led to believe that the actual bias can only take the values 2^{-4} and -2^{-4} and so the obvious decision rule is to guess for the XOR of the round key bits the value 1 if $\hat{c} < 0$, and the value 0 if $\hat{c} > 0$. From (2.10), however, we obtain:

$$z_0 = 0, z_1 = 0 \rightarrow \mathrm{cor}_h = (-1)^Z \cdot 3 \cdot 2^{-5}$$

$$z_0 = 0, z_1 = 1 \rightarrow \mathrm{cor}_h = (-1)^Z \cdot 3 \cdot 2^{-5}$$

$$z_0 = 1, z_1 = 0 \rightarrow \mathrm{cor}_h = (-1)^Z \cdot (-1) \cdot 2^{-5}$$

$$z_0 = 1, z_1 = 1 \rightarrow \mathrm{cor}_h = (-1)^Z \cdot 3 \cdot 2^{-5}$$

In the first, second and last cases, the actual correlation is $(-1)^Z \cdot 3 \cdot 2^{-5}$, which is 50% larger than the value that would have been obtained using Trail 1 only. Using 2^{10} known-plaintexts, the success rate of Algorithm 1 would increase from the predicted 98% to 100%.

In the third case, however, not only the magnitude of the correlation has decreased, but also the sign has changed. This means that Algorithm 1's estimate for Z will be *usually wrong*! The success rate drops from the predicted 98% to $100 - 84.89 = 15.1\%$. We conclude that the average success rate of Matsui's Algorithm 1 drops from the predicted 98% to

$$0.75 \cdot 100\% + 0.25 \cdot 15.1\% \approx 79\%.$$

When the data complexity is increased, the estimate of the actual correlation through the sample correlation is improved. This means that the first term in the sum increases, while the second one decreases. The success probability in

the general case is given approximately by:

$$1-\left(0.75\cdot\Phi\left(\frac{-\left(\frac{N}{2}+3\cdot N\cdot 2^{-6}-\frac{N}{2}\right)}{\sqrt{\frac{N}{4}-9\cdot N\cdot 2^{-12}}}\right)+0.25\cdot\Phi\left(\frac{-\left(\frac{N}{2}-N\cdot 2^{-6}-\frac{N}{2}\right)}{\sqrt{\frac{N}{4}+N\cdot 2^{-12}}}\right)\right).$$

Differentiating with respect to N shows that the function is maximized with a success rate of 80% when $N = 2^{9.12}$, and tends to 75% as N tends to 2^{32}. So we get the following observation.

Observation: In an attack based on (the original, non-extended version of) Matsui's Algorithm 1 the optimal number of plaintexts can be smaller than the full codebook. Increasing the number of plaintexts beyond this optimal number may *decrease* the success rate of the attack.

2.2.4 Knowing Only One of the Trails 2–4

Similar to the case of Trail 1, we can use the individual correlations presented in Section 2.2.2. Hence, for Trail 2, the adversary computes

$$\mathrm{cor}_{\mathrm{p}}^{(2)} = (-1)^{Z+z_0}2^{-5}$$

and concludes that 2^{12} known-plaintexts should be sufficient to estimate $Z + z_0$ with a success rate of 98%. Since the predicted correlation differs only in sign, the decision rule for the guessed sum of the round key bits is as before. Repeating the success rate analysis and using the numbers from Table 2.1, we learn that the success rate with 2^{12} known-plaintexts drops from the predicted 98% to

$$0.5\cdot 100\% + 0.25\cdot 98\% + 0.25\cdot 0\% = 74.5\%\,.$$

The success rate is maximized and saturates with 75% when N grows beyond $2^{12.1}$ as the middle term tends to 100% and the others stay steady. Similar computations give for Trail 3 and Trail 4 the same result.

2.2.5 Knowing all Trails

When all trails are taken into account, Matsui's Algorithm 1 can be extended and recover more than a single bit, see also [137]. The approach can be summarized as follows. The adversary knows now that the correlation of the hull can take 4 values, *cf.* (2.9) and Table 2.1. The adversary divides the space of possible $\hat{c}$ outcomes into four regions instead of just two. After collecting N plaintexts, the adversary computes $\hat{c}$ and guesses for the key bits the values that produce the correlation closest to $\hat{c}$. We can compute the success rate as follows:

If $Z = 0$ and $z_0 z_1 \in \{00, 01, 11\}$, then $\text{cor}_h = 3 \cdot 2^{-5}$. The attack will be successful if $\hat{c} > 2^{-4}$. When $N = 2^{12}$, this happens with probability 0.98. The adversary obtains $1 + 3(-1/3 \log_2(1/3)) = 1 + \log_2(3) \approx 2.6$ bits of information.

If $Z = 0$ and $z_0 z_1 = 10$, then $\text{cor}_h = -1 \cdot 2^{-5}$. The attack will be successful if $-2^{-4} < \hat{c} < 0$. When $N = 2^{12}$, this happens with probability 0.95. The adversary obtains 3 bits of information.

If $Z = 1$ and $z_0 z_1 = 10$, then $\text{cor}_h = 2^{-5}$. The attack will be successful if $0 < \hat{c} < 2^{-4}$. When $N = 2^{12}$, this happens with probability 0.95. The adversary obtains 3 bits of information.

If $Z = 1$ and $z_0 z_1 \in \{00, 01, 11\}$, then $\text{cor}_h = -3 \cdot 2^{-5}$. The attack will be successful if $\hat{c} < -2^{-4}$. When $N = 2^{12}$, this happens with probability 0.98. The adversary obtains 2.6 bits of information.

2.3 On the Wrong-Key-Randomization Hypothesis

In this section we point out the importance of the sampling strategy employed for obtaining plaintext/ciphertext pairs in a linear attack. In previous work, sampling with replacement was often assumed. We continue this line of research, by assuming that sampling without replacement is used (*i.e.*, that no duplicate pairs of plaintexts and ciphertext are considered). In some applications, this is the more natural setting. For instance, some modes of operation such as counter mode are designed to avoid duplicate plaintext/ciphertext pairs.

Under this assumption, we derive a formula for the success probability and the data complexity of a linear attack for sampling without replacement. This formula confirms the intuitive notion that, for sampling without replacement, the sample bias converges faster to the underlying bias, which means that the data complexity can be reduced. For the purpose of deriving this formula, we redevelop a model for the distribution of the sample bias in wrong keys.

2.3.1 The Wrong-Key-Randomization Hypothesis

The success rate analysis performed by Selçuk [147] uses order statistics to investigate the probability that the right key is amongst the 2^{m-a} top ranked keys. The main underlying assumption of this analysis is that the real bias for a wrong key is zero, and thus, that the sample bias for a wrong key would have a normal distribution centered around zero. This assumption may be summarized in the following hypothesis:

Hypothesis 1 (Simple wrong-key-randomization hypothesis)**.** *The bias for a wrong key equals zero:*

$$\epsilon_w = 0.$$

If Hypothesis 1 is true, we have the following lemma.

Lemma 6. *Let $\hat{\boldsymbol{\epsilon}}_w$ be the sample bias obtained from a counter associated with a wrong key using N pairs of plaintexts and ciphertexts. Assuming Hypothesis 1 is true and sampling is performed with replacement, we have approximately*

$$\hat{\boldsymbol{\epsilon}}_w \sim \mathcal{N}(0, \frac{1}{4N}).$$

However, Bogdanov and Tischhauser noted in [42] that, in accordance with Daemen and Rijmen [60], the underlying bias of a random linear approximation is not necessarily zero but a random variable. This resulted in the following extension of Hypothesis 1.

Hypothesis 2 (Bogdanov and Tischhauser [42])**.** *The bias $\boldsymbol{\epsilon}_w$ for a wrong key is distributed as for a random permutation,* i.e.

$$\boldsymbol{\epsilon}_w \sim \mathcal{N}(0, 2^{-n-2}).$$

This hypothesis requires the use of a compound model for the sample bias, which takes into account the distribution of the wrong bias. It leads to the following statement about the distribution of the sample bias for wrong keys.

Lemma 7 ([42], Lemma 1)**.** *Let $\hat{\boldsymbol{\epsilon}}_w$ be defined as before, then assuming the validity of Hypothesis 2, we have approximately*

$$\hat{\boldsymbol{\epsilon}}_w \sim \mathcal{N}(0, \frac{1}{4} \cdot \left(\frac{1}{N} + \frac{1}{2^n}\right)).$$

Selçuk gives the success probability of a linear attack as

$$P_S(N) = \Phi\left(2\sqrt{N}|\epsilon_0| - \Phi^{-1}(1 - 2^{-a-1})\right), \tag{2.15}$$

which holds under Hypothesis 1. However, as was noted by Bogdanov and Tischhauser, the bias for wrong keys has a normal distribution centered around zero, in accordance with Hypothesis 2. Using the distribution of Lemma 7, they adapt Selçuk's formula to

$$P_S(N) = \Phi\left(2\sqrt{N}|\epsilon_0| - \sqrt{1 + \frac{N}{2^n}}\Phi^{-1}(1 - 2^{-a-1})\right). \tag{2.16}$$

An experimental verification of the above formula is provided in [39].

2.3.2 Sample Bias

This section deals with the distribution of the sample bias for wrong keys. The details of this distribution are relevant for the construction and the analysis of statistical procedures that attempt to distinguish the right key from wrong keys.

Generally speaking, the sample bias of wrong keys can be fully described given the distribution of the bias for the wrong keys and a *sampling strategy*. The former is completely determined by the choice of Hypotheses 1 or 2 and requires no further discussion. The latter will now be discussed.

Finally, the main result of this section is presented in Theorem 1, which approximates the distribution of the sample bias for a random wrong key in the case of sampling without replacement. The resulting distribution turns out to be the same as the one given in Lemma 6.

Sampling strategies

The way plaintext/ciphertext pairs are obtained in linear cryptanalysis corresponds to sampling from a population of size 2^n. For each sample, a trial is conducted: it is checked whether or not a fixed linear approximation holds. The outcome of each trial (zero or one) is added to a counter $\hat{T}$.

One could conceive of many strategies for sampling, but here we will limit the discussion to two common cases:

1. **Sampling with replacement**. Trials are independent and $N > 2^n$ is possible.
2. **Sampling without replacement**. Trials are not independent and $N \leq 2^n$.

The use of the first sampling strategy leads to a binomial distribution for the counters. The existing analyses that we are aware of [34, 42, 147] implicitly start from this assumption. An exception is the notion of *distinct known-plaintext* attacks in recent work by Blondeau and Nyberg [37], which was developed in parallel to this work.

We now argue that the second strategy is preferable. This will lead to a hypergeometric distribution for the counters. Since for a given key, a specific plaintext always results in the same ciphertext, duplicates provide no new information for the estimation of the real bias. Moreover, increasing the data

complexity beyond what is needed for the attack reduces its efficiency, and may render it worse than exhaustive search when the bias is small and the data complexity grows beyond the time complexity of exhaustive search. Hence, whenever possible, an adversary should prefer sampling without replacement.

Bias for the right key

In Expressions (2.15) and (2.16) given in Section 2.3.1 for the success probability, the absolute bias for the right key is represented by a fixed value $|\epsilon_0|$. In practice, however, the right-key bias depends on the unknown value of the right key. The absolute right-key bias is therefore better modeled as a random variable $|\boldsymbol{\epsilon}_0|$. Hence, it is necessary to find an adequate model for the distribution of the bias for the right key. This is an independent problem, which we briefly discuss in Section 2.5.2, but that is not solved in this work. Instead, we will assume that the probability density function $f_{\boldsymbol{\epsilon}_0}(\varepsilon)$ of the bias for the right key is available. Several proposals for such a *right-key hypothesis* can be found in the literature, see for instance [1, 37, 42, 43].[1]

In the setting described above, the success probability becomes a random variable $\mathbf{P}_S(N)$, or more explicitly $\mathbf{P}_S(N, \boldsymbol{\epsilon}_0)$. Typically, one is interested in the average success probability $\mathbf{E}[\mathbf{P}_S(N, \boldsymbol{\epsilon}_0)]$. That is,

$$\mathbf{E}[\mathbf{P}_S(N, \boldsymbol{\epsilon}_0)] = \int_{-1/2}^{1/2} P_S(N, \varepsilon) f_{\boldsymbol{\epsilon}_0}(\varepsilon) d\varepsilon.$$

Despite this complication, we shall continue to use the notation $P_S(N)$ in the sense that $P_S(N) = \mathbf{E}[\mathbf{P}_S(N, \boldsymbol{\epsilon}_0)]$. Furthermore, we will continue to use the term $|\epsilon_0|$ to denote the actual right-key bias selected from the appropriate distribution as a result of fixing the key.

Finally, we note that Hypothesis 2 makes no mention of the right key. The dependence of the biases for wrong keys on the right key is neglected.

Sampling without replacement

Given that duplicate plaintexts provide no additional information to the cryptanalyst, we would like to analyse the attack without them.

Assume then that N distinct plaintext/ciphertext pairs are sampled at random from the total population of 2^n pairs. The counter for a specific wrong key

[1] This Author finds none of these proposals satisfactory, which is why none of them is used in this Thesis. This is further discussed in Section 2.5.

follows a hypergeometric distribution

$$\mathbf{T}_w \mid \mathbf{R} \sim \mathcal{HG}(N, 2^n, R),$$

where $R = 2^n(\frac{1}{2} + \epsilon_w)$ is the number of plaintext/ciphertext pairs in the population for which the linear approximation holds (*i.e.*, the "real" bias as per [60]). Given this starting point, the proof of the next theorem derives the distribution of the sample bias for a random wrong key.

Theorem 1 (Lemma 6, *stet.*)**.** *Under Hypothesis 2, and for sampling without replacement, the sample bias $\hat{\boldsymbol{\epsilon}}_w$ of a random wrong key can be approximated by*

$$\hat{\boldsymbol{\epsilon}}_w \sim \mathcal{N}(0, \frac{1}{4N}).$$

Proof. By Lemma 3, we have asymptotically

$$\mathbf{T}_w \mid \boldsymbol{\epsilon}_w \sim \mathcal{N}(N\left(\frac{1}{2} + \epsilon_w\right), N\left(1 - \frac{N}{2^n}\right)\left(\frac{1}{4} - \epsilon_w^2\right)).$$

Since ϵ_w^2 is small, we have approximately

$$\mathbf{T}_w \mid \boldsymbol{\epsilon}_w \sim \mathcal{N}(N\left(\frac{1}{2} + \epsilon_w\right), \frac{N}{4}\left(1 - \frac{N}{2^n}\right)).$$

It follows that for the sample bias

$$\hat{\boldsymbol{\epsilon}}_w \mid \boldsymbol{\epsilon}_w \sim \mathcal{N}(\epsilon_w, \frac{1}{4N}\left(1 - \frac{N}{2^n}\right)).$$

A compound normal distribution with normally distributed mean is again normal. That is, if $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \sigma_1^2)$ with $\boldsymbol{\mu} \sim \mathcal{N}(\mu, \sigma_2^2)$, then $\mathbf{X} \sim \mathcal{N}(\mu, \sigma_1^2 + \sigma_2^2)$. In this particular case we obtain

$$\mathbf{E}\left[\hat{\boldsymbol{\epsilon}}_w\right] = 0$$

$$\mathbf{Var}\left[\hat{\boldsymbol{\epsilon}}_w\right] = \frac{1}{4N}\left(1 - \frac{N}{2^n}\right) + \frac{1}{2^{n+2}} = \frac{1}{4N}.$$

□

The preceding theorem shows that $\hat{\boldsymbol{\epsilon}}_w$ for sampling without replacement has approximately the same distribution as given by Lemma 6. In other words, the correction provided in Lemma 7 should *not* be taken into account for sampling without replacement. Note, however, that the result is based on Hypothesis 2 rather than the obsolete Hypothesis 1.

2.3.3 Average Success Probability and Data Complexity

We will now derive formulae for the average success probability of an attack using sampling without replacement and the data complexity required for a successful attack.

To compute the average success probability, we will make the approximation that the non-identically distributed sample biases for wrong keys can be replaced by an equal amount of independent and identically distributed random variables with distribution given by Theorem 1. A similar approximation was also implicitly made in [42] and greatly simplifies the distribution of the order statistics.

The derivation of $P_S(N)$ is similar to that of Selçuk [147], with the important difference that the counter for the right key is distributed as

$$\mathbf{T}_0 \sim \mathcal{HG}(N, 2^n, \left(\frac{1}{2} + \epsilon_0\right) 2^n). \tag{2.17}$$

The corresponding distribution function of $\hat{\boldsymbol{\epsilon}}_0$ will be denoted by $F_{\hat{\boldsymbol{\epsilon}}_0}$ and can be written in terms of the distribution function $F_{\mathbf{T}_0}$ of $\mathbf{T}_0$:

$$F_{\hat{\boldsymbol{\epsilon}}_0}(\varepsilon) = F_{\mathbf{T}_0}\left(\frac{N}{2} + N\varepsilon\right).$$

Following Selçuk, without loss of generality, we only consider the case $\epsilon_0 \geq 0$. The discussion for $\epsilon_0 < 0$ is completely analogous. It will be assumed that the distribution of an order statistic of the sample biases $\hat{\boldsymbol{\epsilon}}_w$ for wrong keys is approximately degenerate relative to that of the right key—Selçuk makes the same approximation in his discussion. Per [147], the expected value of the $(2^m - 2^{m-a})$-th order statistic, namely $\boldsymbol{\zeta}$, is approximately given by $\mathbf{E}[\boldsymbol{\zeta}] = \Phi^{-1}(1 - 2^{-a-1})/(2\sqrt{N})$. Noting that $\Pr[\hat{\boldsymbol{\epsilon}}_\mathbf{0} < 0] \approx 0$, we have for the average success probability

$$\begin{aligned}
P_S(N) &= \Pr[\hat{\boldsymbol{\epsilon}}_0 - \boldsymbol{\zeta} > 0] \\
&\approx \Pr[\hat{\boldsymbol{\epsilon}}_0 > \mathbf{E}[\boldsymbol{\zeta}]] \\
&= 1 - F_{\hat{\boldsymbol{\epsilon}}_0}\left(\frac{\Phi^{-1}(1 - 2^{-a-1})}{2\sqrt{N}}\right) \\
&= 1 - F_{\mathbf{T}_0}\underbrace{\left(\frac{N}{2} + \frac{\sqrt{N}}{2}\Phi^{-1}(1 - 2^{-a-1})\right)}_{k(N)}.
\end{aligned}$$

An accurate approximation of $F_{\mathbf{T}_0}$ can be obtained by using a normal approximation with respect to N. Indeed, by applying Lemma 3 to $\mathbf{T}_0$, one obtains the approximation (assuming $\epsilon_0^2 \approx 0$)

$$\mathbf{T}_0 \sim \mathcal{N}(N\left(\frac{1}{2} + |\epsilon_0|\right), \frac{N}{4}\left(1 - \frac{N}{2^n}\right)),$$

which is accurate if N and 2^n are sufficiently large. It can be verified that the above expression also holds for $\epsilon_0 < 0$. In terms of the standard normal distribution, we have

$$\begin{aligned} F_{\mathbf{T}_0}(k(N)) &\approx \Phi\left(\frac{k(N) - N\left(\frac{1}{2} + |\epsilon_0|\right)}{\sqrt{\frac{N}{4}\left(1 - \frac{N}{2^n}\right)}}\right) \\ &= \Phi\left(\frac{\sqrt{N}\Phi^{-1}(1 - 2^{-a-1})/2 - N|\epsilon_0|}{\sqrt{\frac{N}{4}\left(1 - \frac{N}{2^n}\right)}}\right) \\ &= \Phi\left(\frac{\Phi^{-1}(1 - 2^{-a-1}) - 2\sqrt{N}|\epsilon_0|}{\sqrt{1 - \frac{N}{2^n}}}\right). \end{aligned}$$

By symmetry, we then obtain the simple result of the theorem below.

Theorem 2. *Assume Hypothesis 2 holds. Let $P_S(N)$ denote the average success probability of a linear attack on a block cipher given N distinct known-plaintext/ciphertext pairs. If the bias of the right key is ϵ_0 and the desired advantage is a, then we have*

$$P_S(N) = \Phi\left(\frac{2\sqrt{N}|\epsilon_0| - \Phi^{-1}(1 - 2^{-a-1})}{\sqrt{1 - \frac{N}{2^n}}}\right),$$

for sampling without replacement.

Note that in the above expression for the success probability, and in the preceding discussion, we have assumed that the bias ϵ_0 is fixed. In practice, this is not the case and instead the right-key hypothesis should be taken into account. Recall from Section 2.3.2 that the average success probability can be obtained as

$$\mathbf{E}\left[P_S(N, \boldsymbol{\epsilon}_0)\right] = \int_{-1/2}^{1/2} P_S(N, \varepsilon) f_{\boldsymbol{\epsilon}_0}(\varepsilon) d\varepsilon,$$

where $f_{\boldsymbol{\epsilon}_0}(\varepsilon)$ is the probability density function of $\boldsymbol{\epsilon}_0$.

Theorem 2 directly leads to an expression for the data complexity, which is given below. It also shows that non-monotonicity is possible for sampling without replacement. Note that when $P_S(N)$ is non-monotonous, a value of the success probability P_S will correspond to two data complexities N. For simplicity, we defer the discussion on non-monotonicity to Section 2.4 and only deal with the monotonous case here.

Theorem 3. *Under the same conditions as Theorem 2, the number of plaintext/ciphertext pairs required to obtain an average success probability P_S is*

$$N = \left(\frac{2|\epsilon_0|\alpha \pm \sqrt{(2\epsilon_0\alpha)^2 - (\alpha^2 - \beta^2)(2^{-n}\beta^2 + 4\epsilon_0^2)}}{4\epsilon_0^2 + 2^{-n}\beta^2} \right)^2,$$

where $\alpha = \Phi^{-1}(1 - 2^{-a-1})$ and $\beta = \Phi^{-1}(P_S)$. The plus sign applies whenever $P_S \geq 1/2$, otherwise the minus sign applies.

Proof. The result is obtained by solving the equation

$$\Phi^{-1}(P_S)\sqrt{1 - \frac{N}{2^n}} = 2\sqrt{N}|\epsilon_0| - \Phi^{-1}(1 - 2^{-a-1}).$$

Letting $\alpha = \Phi^{-1}(1 - 2^{-a-1})$ and $\beta = \Phi^{-1}(P_S)$, and squaring yields

$$\beta^2 \left(1 - 2^{-n}N\right) = 4N|\epsilon_0|^2 - 4\sqrt{N}|\epsilon_0|\alpha + \alpha^2.$$

Grouping terms appropriately, we obtain

$$(4|\epsilon_0|^2 + 2^{-n}\beta^2)N - 4|\epsilon_0|\alpha\sqrt{N} + (\alpha^2 - \beta^2) = 0.$$

This equation is quadratic in $\sqrt{N}$ and has the solutions

$$\sqrt{N} = \frac{2|\epsilon_0|\alpha \pm \sqrt{(2\epsilon_0\alpha)^2 - (\alpha^2 - \beta^2)(2^{-n}\beta^2 + 4|\epsilon_0|^2)}}{4|\epsilon_0|^2 + 2^{-n}\beta^2}.$$

□

Note that the approximation for large $|\epsilon_0|$ gives the same data complexity as Selçuk [147]. This is because large biases require fewer plaintext/ciphertext pairs, and for very small N the difference between sampling with and without replacement is negligible.

In general, the data complexity for sampling without replacement is lower. This is a consequence of the fact that no duplicates are used. Bogdanov and

Tischhauser provide an algorithm to compute the data complexity for a given success probability [42].[2]

Figure 2.7 shows the data complexity for a large bias and for a small bias. For $|\epsilon_0| = 2^{-14}$ the difference between the data complexities is relatively small. For instance, at a success probability of 95%, the data complexity is about 14% higher for sampling with replacement. The difference is more significant for small values of the bias. In this case, for a success probability of 95%, the data complexity is 69% higher for sampling with replacement. Note that, due to duplicates, the data complexity for sampling with replacement can exceed the size of the codebook, but not that of the key.

2.4 On the Success Probability of Matsui's Algorithm 2 and its Non-Monotonicity

In the previous section we saw in Theorem 2 that for a fixed key the success probability $P_S(N)$ is given by

$$P_S(N) = \Phi\left(\frac{2\sqrt{N}|\epsilon_0| - \Phi^{-1}(1-2^{-a-1})}{\sqrt{1-\frac{N}{2^n}}}\right), \tag{2.18}$$

for sampling without replacement. Investigating this function we can see that for some values of ϵ_0 and a, the function may be non-monotonous in N (*i.e.*, increasing the data complexity beyond some point may decrease the success probability). Non-monotonicity was already observed in [42] where it was attributed to the noise introduced due to duplicate data pairs. Since we see that non-monotonicity occurs also when sampling is done without replacement, a new explanation is required.

2.4.1 Explanation of Non-monotonicity

When the bias ϵ_0 of the right key is close to zero, there is a non-negligible probability that some of the wrong keys have a higher absolute bias than $|\epsilon_0|$. This is depicted in Figure 2.8. In this case, the correct key should *not* be expected to be ranked higher than (some of the) wrong keys. As N increases,

[2] In the non-monotonous case, their algorithm returns the lowest data complexity corresponding to the given success probability.

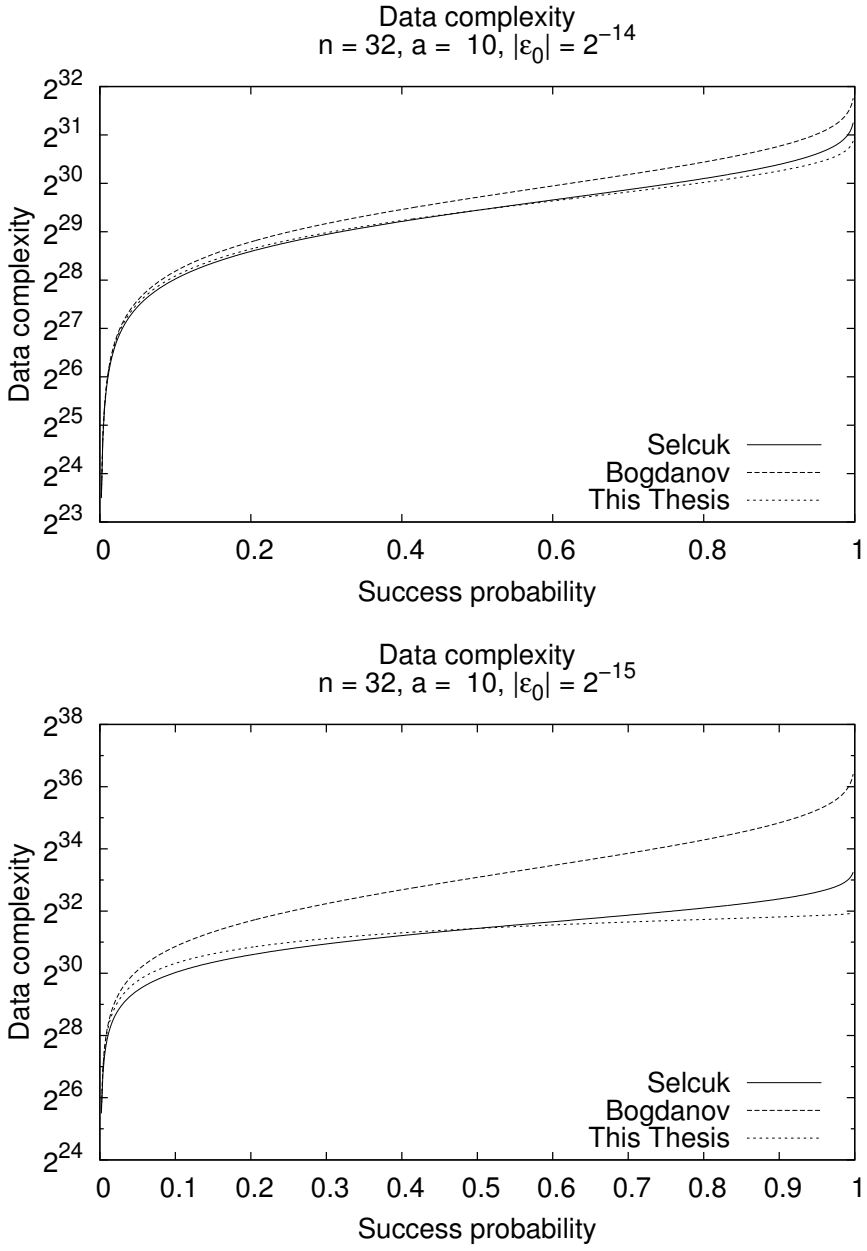


Figure 2.7: The theoretical data complexity for a given success probability. The top figure corresponds to a relatively large bias compared to the bias in the bottom figure.

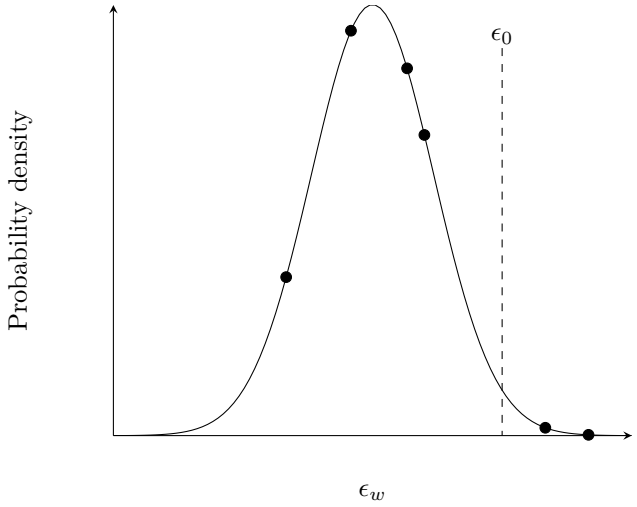


Figure 2.8: The biases for a few wrong keys are indicated by dots, the dashed line represents the bias for the right key. Two of the wrong keys have a larger bias than the right key. If the adversary requires such an advantage that the right key needs to be amongst the top 2 keys, the attack would fail once enough data is obtained to place the keys in their true order.

the accuracy of the ranking increases because the variances of all sample biases decrease (*i.e.*, they converge to the underlying bias). It follows that, if there are wrong keys with absolute bias higher than $|\epsilon_0|$, then for large N those will be ranked higher than the right key. If there are more such keys than the advantage allows, *i.e.* more than 2^{m-a}, then, due to the sample variance, the right key may start high on the list of candidate keys (*i.e.*, amongst the top 2^{m-a} values) as a false-positive but will slowly sink to its "real" position (*i.e.*, below the top 2^{m-a} keys) due to improved accuracy as a result of using more data. In this case, if $N \to \infty$ (or $N = 2^n$ without replacement) then also $\mathbf{P_S}(N) \to 0$, almost surely. In other words: given all possible information, the attack **should** fail, as it indeed does.

From the discussion above we conclude that non-monotonous behavior indicates that the attack can not be conducted using Matsui's Algorithm 2. In fact, a correct identification of the right key amounts to a false positive. This is formalized in the next section by giving a bound on the required bias for given

values of m and a with respect to the block size n.

2.4.2 Conditions for Non-monotonicity

This section derives necessary conditions for non-monotonous behavior of the success probability. These conditions are necessarily probabilistic, since they are determined by the biases of individual wrong keys. Hence, it can be expected that for some values of a, n, m and ϵ_0, $\mathbf{P_S}(N)$ is non-monotonous only for *some* keys. The main result of this section is Theorem 4, which gives a necessary condition for average non-monotonicity.

The following lemma gives a first result on the probability of monotonicity. It is independent of the sampling strategy.

Lemma 8. *The probability that* $\mathbf{P_S}(N)$ *is a monotonous function is given by*

$$\Pr[\mathbf{P_S}(N) \textit{ is monotonous}] = \Phi\left(\frac{\left(2^{-a} - 2^{-m-1} - p\right) 2^{m/2}}{\sqrt{p(1-p)}}\right),$$

where

$$p = 2\left(1 - \Phi\left(|\epsilon_0| 2^{n/2+1}\right)\right).$$

and ϵ_0 *is as defined in Section 2.3.2*

Proof. The probability that a wrong key has absolute bias larger than $|\epsilon_0|$ can be computed as

$$p = \Pr[|\boldsymbol{\epsilon}_w| \geq |\epsilon_0|] = 2(1 - \Pr[\boldsymbol{\epsilon}_w < |\epsilon_0|]) = 2\left(1 - \Phi\left(|\epsilon_0| 2^{n/2+1}\right)\right),$$

since $\boldsymbol{\epsilon}_w \sim \mathcal{N}(0, 2^{-n-2})$. For a decreasing success probability, we require at least 2^{m-a} keys with bias larger than $|\epsilon_0|$. Let $\mathbf{C}$ be the random variable describing the number of such keys, then $\mathbf{C}$ is approximately binomially distributed. Furthermore, if the number of keys 2^m is sufficiently large, $\mathbf{C}$ can be approximated with a normal distribution:

$$\mathbf{C} \sim \mathcal{N}(p2^m, p(1-p)2^m).$$

The probability that $P_S(N)$ is monotonous for some $|\epsilon_0|, m$ and a can hence be computed as

$$\Pr[\mathbf{P_S}(N) \text{ is monotonous}] = \Pr[\mathbf{C} < 2^{m-a}].$$

Using the normal approximation of $\mathbf{C}$, we get:

$$\begin{aligned}\Pr[\mathbf{P_S}(N) \text{ is monotonous}] &= \Phi\left(\frac{2^{m-a} - 2^{-1} - p2^m}{\sqrt{p(1-p)2^m}}\right) \\ &= \Phi\left(\frac{(2^{-a} - 2^{-m-1} - p)2^{m/2}}{\sqrt{p(1-p)}}\right).\end{aligned}$$

□

In the previous lemma, a fixed absolute right-key bias $|\epsilon_0|$ was assumed. When $\boldsymbol{\epsilon}_0$ has a known probability distribution, we can compute the average probability of monotonicity from Lemma 8 by

$$\Pr[\mathbf{P_S}(N) \text{ is monotonous}] = \int_{-1/2}^{1/2} \Phi\left(\frac{\left(2^{-a} - 2^{-m-1} - p(\varepsilon)\right) 2^{m/2}}{\sqrt{p(\varepsilon)(1-p(\varepsilon))}}\right) d\varepsilon,$$

with $p(\varepsilon)$:

$$p(\varepsilon) = 2\left(1 - \Phi\left(|\varepsilon| 2^{n/2+1}\right)\right).$$

Observe that, as the bias $\boldsymbol{\epsilon}_0$ approaches 0 in probability, the success probability is almost surely non-monotonous. If $\mathbf{E}[\mathbf{C}] \geq 2^{m-a}$, then $\mathbf{P_S}(N)$ is non-monotonous on average, *i.e.* over all keys and all choices of plaintexts. This condition can be used to derive the following theorem, which assumes a fixed absolute bias for the right key.

Theorem 4. *The success probability is monotonous on average if and only if*

$$|\epsilon_0| > 2^{-n/2-1}\Phi^{-1}(1-2^{-a-1}).$$

Proof. The condition $\mathbf{E}[\mathbf{C}] \geq 2^{m-a}$ corresponds to the inequality

$$2\left(1-\Phi\left(|\epsilon_0|2^{n/2+1}\right)\right)2^m \geq 2^{m-a},$$

which can be rewritten as

$$1-\Phi\left(|\epsilon_0|2^{n/2+1}\right) \geq 2^{-a-1}$$

$$\iff |\epsilon_0| \leq 2^{-n/2-1}\Phi^{-1}(1-2^{-a-1}).$$

□

Theorem 4 expresses a necessary condition for a nonzero success probability in the average case as $N \to \infty$ (or $N = 2^n$ without replacement). Hence, when the condition does not hold, the advantage a can only be obtained as a false-positive during key recovery with Matsui's Algorithm 2.

Bogdanov and Tischhauser have observed non-monotonous behavior with $|\epsilon_0| = 2^{-10}, a = 12$ and $n = 20$ [42]. Theorem 4 predicts:

$$p = 2\left(1-\Phi(2^{-10}\cdot 2^{11})\right) \approx 0.0455.$$

and

$$|\epsilon_0| \leq 2^{-11}\Phi^{-1}(1-2^{-13}) \approx 2^{-9.125}.$$

Hence, with these parameters, the average attack setup will lead to non-monotonous behavior. By Lemma 8, the probability of monotonicity is $\Phi(-218.78) \approx 0$.

2.5 A Few Observations about Linear Cryptanalysis

We now discuss a few aspects of linear cryptanalysis that are often misunderstood or overlooked. By including it in this Thesis, we try to view it in a new light rather than claim any novel results.

2.5.1 From Linear Trails to Linear Hulls

It is important to note that the problem of obtaining the right value for the bias ϵ is still open. The problem is that even if in many cases it is easy to calculate the probability of a specific linear trail directly from the cipher's structure, this bias remains impossible to measure unless the key is already known. On the other hand, the bias of the linear hull, which is easy for the adversary to measure, is hard to compute theoretically. To better understand this, it is useful to consider the values the bias can take. When a single trail is considered, the magnitude of its bias is fixed, and the key only affects the sign. However, since an adversary has access only to the input and output of the cipher (and not to intermediate rounds), the bias can only be measured over the linear hull, which is composed of an unknown number of trails.

In Matsui's original paper [110], he presented a single trail and used its bias for the attack. This approach worked in the case of DES because the corresponding linear hull consisted only of one significant trail. However, in other ciphers, the fact that the hull may be composed of multiple trails leads to an under- or overestimation of the bias. A counterexample to the case of DES is presented in Section 2.1, showing how such an erroneous estimation of the hull's bias through a single trail affects the success probability of the attack.

2.5.2 The Influence of the Key over the Bias

Another important aspect of bias estimation is the influence of the key. In the case of a linear trail, the magnitude of the bias is known and only the sign is affected by the key. The bias is therefore a random variable that can take only two values. The actual value is determined by a subset of key bits.[3] However, in the case of a linear hull, the bias is the sum of multiple biases coming from the underlying trails. Each of these underlying biases is a random variable with different values coming from different distributions, and resulting in a probability distribution rather than a single value for the hull's bias (*cf.* Section 2.2). Generally, the nature of this distribution is cipher-dependent. For real-life ciphers, the distribution is hard to derive due to the large number of involved non-trivial trails, and is therefore unknown to the adversary. The probabilistic nature of the bias was discussed in [1]. In practice, many works simply use the mean of the absolute bias in place of the actual distribution for attacks, which is generally insufficient.

In [36] Nyberg and Blondeau tried to address these concerns through an estimation of the ELP. The idea behind their approach is to split the involved

[3]This is the property exploited when using Matsui's Algorithm 1.

linear trails into two groups Q and R representing the dominating trails and the "rest". Once the set of dominating trails is properly accounted for, the "rest" can be modeled as random noise. However, in the general case this approach simplifies the problem only slightly. First, it assumes that the adversary can find all dominating trails, which again requires sieving through the space of all possible ones. Second, even if a certain set is already given, it is required that it be exhaustive (*i.e.*, that no dominant trails were overlooked), which is again hard to verify in practice. Finally, although it can readily be shown that the estimation error of the bias decreases with the number of known trails, this is not reflected in [36], which always models R as random noise, regardless of the set's size.

2.5.3 Using Linear Cryptanalysis in Practice

In spite of all of this, progress in the practical application of linear cryptanalysis has not stopped. For example, papers which follow the evolution of linear trails either manually [3] or using automated tools [32, 161] are still published, suggesting that an attack can be leveraged using such trails (ignoring the possible linear hull effect). Similarly, papers ignoring the key dependency [51] are being published, then improved [163], criticized [35, 81], then salvaged [36].

This discussion makes it apparent that there is still much more work to be done before ciphers' security against linear cryptanalysis is understood.

2.5.4 Data Complexity and Key Recovery

Statistical attacks such as linear cryptanalysis are a multi-round game. In the first step, fixing the key for a block cipher is akin to drawing a certain value from the distribution of all possible (right key) biases. Similarly, testing a wrong key is akin to drawing a (wrong key bias) from $\mathcal{N}(0, 2^{-n-2})$. This bias determines the number of data pairs that satisfy the linear hull (under both right and wrong keys) and we call it here the *underlying bias*.

In the second step, the adversary evaluates the linear hull against pairs of plaintexts and ciphertexts to obtain the *sample bias*. Since the underlying bias fully determines the number of data pairs that satisfy the linear hull, using the full codebook results in the underlying bias and the sample bias being the same. However, when only a sample of the codebook is used, the sample bias becomes a compound random variable. Generally speaking, the sample bias is normally distributed around the underlying bias. When more information is

added using new data pairs, the variance of this normal distribution decreases, until it collapses into a single point when the full codebook is used.

It is important to note that this normal distribution has nothing to do with the underlying bias itself, which is fully fixed once the key or the permutation are chosen, but only with the data sample. The nature of this distribution was studied in [35, 42] and in Section 2.3 for different sampling strategies, and is a straightforward process once the underlying bias is known. Key recovery using Matsui's Algorithm 2 is then composed of decrypting one or more rounds using a candidate subkey and obtaining the sample bias associated with each subkey. The wrong-key-randomization hypothesis states that a hull evaluated over data obtained using a wrong key would behave as a random one; hence, the biases for all wrong keys can be modeled as coming from the same distribution. If the underlying bias for the right key is distinctly different from the biases for the wrong keys, it can be distinguished through the sample bias.

2.6 Summary

This chapter dealt with the existing theory of cryptanalysis and presented several contributions. First, we proved the existence of 1-round linear hulls resulting from dependencies between bits in the state. We gave examples for such dependencies and showed that they can either augment or cancel each other, leading to a complication in bias estimation. We devised a method to overcome this by explicitly writing down the Boolean function describing each bit. Then, we corrected errors in the way linear potential is computed in the presence of such 1-round hulls.

In our next contribution we showed how wrong bias estimation affects the success probability of Matsui's Algorithm 1. Such wrong estimation is often the result of the common practice to use the bias of a single trail for the entire hull due to the difficulty to obtain the real bias of the latter. We showed that since each trail may have a positive as well as negative contribution to the hull's bias, an erroneous estimation may result in using either too much or too little data. We also showed that sometimes, increasing the data complexity reduces the success probability of an attack. On the other hand, we showed that when the adversary obtains the exact distribution of the hull's bias by accounting for all invovled trails, Matsui's Algorithm 1 can be used to recover more than a single key bit.

Our next two contributions dealt with Matsui's Algorithm 2. We first highlighted the importance of explicitly choosing a sampling strategy. It is shown that when distinct known-plaintexts are used for linear cryptanalysis, the sample

bias converges faster to the underlying bias and hence less data need to be used to achieve the same success probability. We then showed that the previously described phenomenon of non-monotonous success probability occurs also for distinct known-plaintexts and thus cannot be explained solely by the noise added by duplicates. We offered a new explanation instead, showing that non-monotonicity is the result of an incorrect assumption about the behavior of the right-key bias amongst the biases of wrong keys. We exposed the conditions for non-monotonicity and derived a formula for the probability that an attack would have a monotonous success probability given a set of attack parameters.

Finally, we provided several observations about linear cryptanalysis. These observations are not new and in fact, they can be traced back in one way or another to existing literature. By including them here in an explicit form we hope that they can serve as a roadmap for the still open questions in linear cryptanalysis.

Future work for this chapter is discussed as part of the summary of the next one.

Chapter 3

New Cryptanalytic Techniques

> *"Even among conservative Galactics, science is about slowly improving your models of the world. It's future-oriented. Your children will know more than you do, so the truth you already have can never be called 'perfect'."*
>
> - David Brin, *Startide Rising*

Both the previous chapter and this one deal with the theory of cryptanalysis. The difference is that while Chapter 2 presented and corrected shortcomings in existing theory, this chapter goes beyond it in order to extend existing techniques and develop new ones.

In Section 3.1 we focus our attention on rotational cryptanalysis. Previously, the XOR of round constants was believed to be a countermeasure against rotational cryptanalysis, see *e.g.*, [21, p. 14]. We show that the injection of round constants is akin to introducing an XOR-difference into the state. By abstracting both rotational cryptanalysis and differential cryptanalysis into Rotational-XOR (RX) cryptanalysis we show how a rotational attack can be executed against ARX constructions involving the XOR of round constants. This work was co-authored with Yunwen Liu and published in [13]. Follow-up work to this paper is presented in Sections 4.2 and 5.3, and was published in [10, 109, 133].

In Section 3.2 we present the most important contribution of this Thesis. We show that despite folklore belief, linear approximations with arbitrarily small

biases can be used for a linear attack. This is done by combining data obtained using a single approximation under multiple keys (Section 3.2.4) or multiple approximation under a single key (Section 3.2.5). This work was co-authored with Daniël Bodden and Orr Dunkelman and was made available through ePrint [9].

3.1 Rotational Cryptanalysis in the Presence of Constants

This section deals with the effect of round constants on rotational cryptanalysis. In [96], it was shown that ARX-C, *i.e.*, an ARX construction with the addition of constants, is complete. This means that any function can be implemented using the ARX-C operations. In most cases, the constants are injected into the state either through an XOR operation or through modular addition. When the constant c is rotational-invariant, *i.e.*, $c = (c \lll \gamma)$, for some γ, XORing with c does not change the rotational property of a rotational pair $(x, x \lll \gamma)$. However, when c is not rotational-invariant, the properties of the output require further inspection.

In general, when a constant c that is not rotational-invariant is XORed into a rotational pair $(x, x \lll \gamma)$, the output pair $(x \oplus c, (x \lll \gamma) \oplus c)$, no longer forms a rotational pair. If this pair is given as an input to the modular addition, the basic formula in Proposition 1 of Section 1.1.4 for computing the propagation of the rotational property can no longer be used.

In the sequel, we define a $((a_1, a_2), \gamma)$-*Rotational-XOR-difference* (or in shorthand notation $((a_1, a_2), \gamma)$-RX-difference or simply RX-difference when $(a_1, a_2), \gamma$ are clear or irrelevant) to be the difference of a rotational pair with rotation offset γ under translations a_1 and a_2, *i.e.*, (x, x') have a $((a_1, a_2), \gamma)$-RX-difference if and only if $(x \oplus a_1) \lll \gamma = (x' \oplus a_2)$; we call such a pair an RX-pair. Note that when $a_1 = a_2 = 0$, they simply form a rotational pair. Our goal is to estimate the transition probability of two input RX-differences to an output RX-difference, over modular addition. Without loss of generality, we consider the case where the input rotational pairs are $(x \oplus a_1, y \oplus b_1)$ and $(\overleftarrow{x} \oplus a_2, \overleftarrow{y} \oplus b_2)$, and compute the probability of $\overleftarrow{(x \oplus a_1) \boxplus (y \oplus b_1) \oplus \Delta_1} = (\overleftarrow{x} \oplus a_2) \boxplus (\overleftarrow{y} \oplus b_2) \oplus \Delta_2$.

This section requires some special notation, which is presented in Table 3.1. The main contribution of this section is Theorem 5.

Table 3.1: The notation used for Section 3.1

$x = (x_{n-1}, \cdots, x_1, x_0)$	An n-bit Boolean vector; x_0 is the least significant bit
$\boxplus$	Addition modulo n
$\boxminus$	Subtraction modulo n
$x \| \| y$	The concatenation of x and y
$\|x\|$	The Hamming weight of the Boolean vector x
$x \| y$	The vector bitwise OR operation
$\overleftarrow{x}$	$x \lll 1$
$1_{x \preccurlyeq y}$	A characteristic function which evaluates to 1 if and only if $x_i \leq y_i$ for all i
$SHL(x)$	A non-cyclic left shift of x by one bit
$(I \oplus SHL)(x)$	$x \oplus SHL(x)$
$L(x)^*$	The γ most significant bits of x
$R(x)^*$	The $n - \gamma$ least significant bits of x
$R'(x)^*$	The γ least significant bits of x
$L'(x)^*$	The $n - \gamma$ most significant bits of x

* Note that $x = L(x)||R(x) = L^{'}(x)||R^{'}(x)$.

Theorem 5. *Let $x, y \in \mathbb{F}_{2^n}$ be independent, uniformly distributed random variables. Let $a_1, b_1, a_2, b_2, \Delta_1, \Delta_2$ be constants in $\mathbb{F}_{2^n}$. Then,*

$$\begin{aligned}
&\Pr[\overleftarrow{(x \oplus a_1) \boxplus (y \oplus b_1) \oplus \Delta_1} = (\overleftarrow{x} \oplus a_2) \boxplus (\overleftarrow{y} \oplus b_2) \oplus \Delta_2] \\
&= 1_{(I \oplus SHL)(\delta_1 \oplus \delta_2 \oplus \delta_3) \oplus 1 \preceq SHL((\delta_1 \oplus \delta_3)|(\delta_2 \oplus \delta_3))} \cdot 2^{-|SHL((\delta_1 \oplus \delta_3)|(\delta_2 \oplus \delta_3))|} \cdot 2^{-3} \\
&+ 1_{(I \oplus SHL)(\delta_1 \oplus \delta_2 \oplus \delta_3) \preceq SHL((\delta_1 \oplus \delta_3)|(\delta_2 \oplus \delta_3))} \cdot 2^{-|SHL((\delta_1 \oplus \delta_3)|(\delta_2 \oplus \delta_3))|} \cdot 2^{-1.415},
\end{aligned} \tag{3.1}$$

where

$$\delta_1 = R(a_1) \oplus L^{'}(a_2),$$

$$\delta_2 = R(b_1) \oplus L^{'}(b_2),$$

and

$$\delta_3 = R(\Delta_1) \oplus L^{'}(\Delta_2).$$

Note that when all constants are 0, *i.e.*, $a_1 = a_2 = b_1 = b_2 = \Delta_1 = \Delta_2 = 0$, Theorem 5 predicts $\Pr[\overleftarrow{x \boxplus y} = \overleftarrow{x} \boxplus \overleftarrow{y}]$, which is the normal case for rotational cryptanalysis.

Before moving to prove Theorem 5, we introduce the following two lemmata:

Lemma 9 ([145]). *Let $\zeta_1, \zeta_2, \zeta_3 \in \mathbb{F}_{2^n}$ be constants. Let $x, y \in \mathbb{F}_{2^n}$ be independent random variables. The probability of the differential equation*

$$x \boxplus y = (x \oplus \zeta_1) \boxplus (y \oplus \zeta_2) \oplus \zeta_3 \tag{3.2}$$

is

$$1_{(I \oplus SHL)(\zeta_1 \oplus \zeta_2 \oplus \zeta_3) \preceq SHL((\zeta_1 \oplus \zeta_3)|(\zeta_2 \oplus \zeta_3))} \cdot 2^{-|SHL((\zeta_1 \oplus \zeta_3)|(\zeta_2 \oplus \zeta_3))|}. \tag{3.3}$$

Proof. The complete proof can be found in [145]. □

The following example is provided for a better understanding of Lemma 9.

Example 2. *Let $n = 8$, $\zeta_1 = \mathtt{E}_{16}, \zeta_2 = \mathtt{9}_{16}$ and $\zeta_3 = \mathtt{F7}_{16}$, we have*

$$(I \oplus SHL)(\zeta_1 \oplus \zeta_2 \oplus \zeta_3) = \mathtt{10}_{16},$$

$$SHL((\zeta_1 \oplus \zeta_3)|(\zeta_2 \oplus \zeta_3)) = \mathtt{FE}_{16},$$

and

$$|SHL((\zeta_1 \oplus \zeta_3)|(\zeta_2 \oplus \zeta_3))| = |\mathtt{FE}_{16}| = 7.$$

We evaluate the characteristic function $1_{(I \oplus SHL)(\zeta_1 \oplus \zeta_2 \oplus \zeta_3) \preccurlyeq SHL((\zeta_1 \oplus \zeta_3)|(\zeta_2 \oplus \zeta_3))}$, and see that it is equal to 1 since no bit in $(I \oplus SHL)(\zeta_1 \oplus \zeta_2 \oplus \zeta_3)$ is larger than the respective bit in $SHL((\zeta_1 \oplus \zeta_3)|(\zeta_2 \oplus \zeta_3))$. The probability is then computed to be $2^{-|SHL((\zeta_1 \oplus \zeta_3)|(\zeta_2 \oplus \zeta_3))|} = 2^{-7}$.

Lemma 10. *Let $\zeta_1, \zeta_2, \zeta_3 \in \mathbb{F}_{2^n}$ be constants. For independent random variables $x, y \in \mathbb{F}_{2^n}$, the probability of*

$$x \boxplus y \boxplus 1 = (x \oplus \zeta_1) \boxplus (y \oplus \zeta_2) \oplus \zeta_3 \tag{3.4}$$

is

$$1_{(I \oplus SHL)(\zeta_1 \oplus \zeta_2 \oplus \zeta_3) \oplus 1 \preceq SHL((\zeta_1 \oplus \zeta_3)|(\zeta_2 \oplus \zeta_3))} \cdot 2^{-|SHL((\zeta_1 \oplus \zeta_3)|(\zeta_2 \oplus \zeta_3))|}. \tag{3.5}$$

Proof. [1] Denote the bitwise complement of $x \in \mathbb{F}_{2^n}$ by $\overline{x}$. We have that $x \boxplus \overline{x} = -1$, where $-1 = 2^n \boxminus 1$. Thus

$$x \boxplus y = (-1 \boxminus \overline{x}) \boxplus (-1 \boxminus \overline{y}) = -\overline{x} \boxminus \overline{y} \boxminus 2,$$

and $x \boxplus y \boxplus 1 = \boxminus(\overline{x} \boxplus \overline{y}) \boxminus 1 = \overline{(\overline{x} \boxplus \overline{y})} = (\overline{x} \boxplus \overline{y}) \oplus (-1)$. Rewriting $(\overline{x} \boxplus \overline{y})$ we get

$$(\overline{x} \boxplus \overline{y}) \oplus (-1) = (\overline{x} \oplus \zeta_1) \boxplus (\overline{y} \oplus \zeta_2) \oplus \zeta_3 \oplus (-1),$$

[1] We thank Ernst Schulte-Geers for suggesting a simplified proof to the one appearing in the original paper. The proof given here is an adapted version of his proposal.

which can be written as

$$(x \oplus \overline{\zeta_1}) \boxplus (y \oplus \overline{\zeta_2}) \oplus \overline{\zeta_3} = x \boxplus y.$$

From Lemma 9, the probability of (3.4) is

$$1_{(I \oplus SHL)(\overline{\zeta_1} \oplus \overline{\zeta_2} \oplus \overline{\zeta_3}) \preceq SHL((\overline{\zeta_1} \oplus \overline{\zeta_3})|(\overline{\zeta_2} \oplus \overline{\zeta_3}))} \cdot 2^{-|SHL((\overline{\zeta_1} \oplus \overline{\zeta_3})|(\overline{\zeta_2} \oplus \overline{\zeta_3}))|}. \tag{3.6}$$

Noting that

$$\begin{aligned}
&(I \oplus SHL)(\overline{\zeta_1} \oplus \overline{\zeta_2} \oplus \overline{\zeta_3}) = \\
&(I \oplus SHL)(\zeta_1 \oplus (-1) \oplus \zeta_2 \oplus (-1)\zeta_3 \oplus (-1)) = \\
&(I \oplus SHL)(\zeta_1 \oplus \zeta_2 \oplus \zeta_3 \oplus (-1)) = \\
&(I \oplus SHL)(\zeta_1 \oplus \zeta_2 \oplus \zeta_3) \oplus (I \oplus SHL)(-1)) = \\
&(I \oplus SHL)(\zeta_1 \oplus \zeta_2 \oplus \zeta_3) \oplus 1,
\end{aligned}$$

and that

$$(\overline{\zeta_1} \oplus \overline{\zeta_3})|(\overline{\zeta_2} \oplus \overline{\zeta_3}) = (\zeta_1 \oplus \zeta_3)|(\zeta_2 \oplus \zeta_3),$$

we get that

$$\begin{aligned}
&1_{(I \oplus SHL)(\overline{\zeta_1} \oplus \overline{\zeta_2} \oplus \overline{\zeta_3}) \preceq SHL((\overline{\zeta_1} \oplus \overline{\zeta_3})|(\overline{\zeta_2} \oplus \overline{\zeta_3}))} \cdot 2^{-|SHL((\overline{\zeta_1} \oplus \overline{\zeta_3})|(\overline{\zeta_2} \oplus \overline{\zeta_3}))|} = \\
&1_{(I \oplus SHL)(\zeta_1 \oplus \zeta_2 \oplus \zeta_3) \oplus 1 \preceq SHL((\zeta_1 \oplus \zeta_3)|(\zeta_2 \oplus \zeta_3))} \cdot 2^{-|SHL((\zeta_1 \oplus \zeta_3)|(\zeta_2 \oplus \zeta_3))|},
\end{aligned}$$

which concludes the proof. □

We can now prove Theorem 5.

Theorem 5. *Let $x, y \in \mathbb{F}_{2^n}$ be independent random variables. Let $a_1, b_1, a_2, b_2, \Delta_1, \Delta_2$ be constants in $\mathbb{F}_{2^n}$. Then*

$$\begin{aligned}
&\Pr[\overleftarrow{(x \oplus a_1) \boxplus (y \oplus b_1) \oplus \Delta_1} = (\overleftarrow{x} \oplus a_2) \boxplus (\overleftarrow{y} \oplus b_2) \oplus \Delta_2] = \\
&1_{(I \oplus SHL)(\delta_1 \oplus \delta_2 \oplus \delta_3) \oplus 1 \preceq SHL((\delta_1 \oplus \delta_3)|(\delta_2 \oplus \delta_3))} \cdot 2^{-|SHL((\delta_1 \oplus \delta_3)|(\delta_2 \oplus \delta_3))|} \cdot 2^{-3} + \\
&1_{(I \oplus SHL)(\delta_1 \oplus \delta_2 \oplus \delta_3) \preceq SHL((\delta_1 \oplus \delta_3)|(\delta_2 \oplus \delta_3))} \cdot 2^{-|SHL((\delta_1 \oplus \delta_3)|(\delta_2 \oplus \delta_3))|} \cdot 2^{-1.415}
\end{aligned} \tag{3.7}$$

where

$$\begin{aligned}
\delta_1 &= R(a_1) \oplus L'(a_2), \\
\delta_2 &= R(b_1) \oplus L'(b_2),
\end{aligned}$$

and

$$\delta_3 = R(\Delta_1) \oplus L'(\Delta_2).$$

Proof. Let C^1 be the carry vector of $(x \oplus a_1) \boxplus (y \oplus b_1)$ and let $C^1_{n-\gamma}$ be the carry bit in position $n-\gamma$ (*i.e.*, $C^1_{n-\gamma}$ is the most significant carry produced by $(R(x) \oplus R(a_1)) \boxplus (R(y) \oplus R(b_1)))$. We write $\overleftarrow{(x \oplus a_1) \boxplus (y \oplus b_1) \oplus \Delta_1}$ from (3.7) as the concatenation of its left and right parts:

$$\overleftarrow{(x \oplus a_1) \boxplus (y \oplus b_1) \oplus \Delta_1} =$$

$$\overleftarrow{((L(x) \oplus L(a_1)) \boxplus (L(y) \oplus L(b_1)) \boxplus C^1_{n-\gamma}) \oplus L(\Delta_1)||}$$

$$\overrightarrow{((R(x) \oplus R(a_1)) \boxplus (R(y) \oplus R(b_1))) \oplus R(\Delta_1)} =$$

$$((R(x) \oplus R(a_1)) \boxplus (R(y) \oplus R(b_1))) \oplus R(\Delta_1)||$$

$$((L(x) \oplus L(a_1)) \boxplus (L(y) \oplus L(b_1)) \boxplus C^1_{n-\gamma}) \oplus L(\Delta_1).$$

Similarly, let C^2 be the carry vector of $(\overleftarrow{x} \oplus a_2) \boxplus (\overleftarrow{y} \oplus b_2)$, and C^2_γ the carry bit in position γ (*i.e.*, C^2_γ is the most significant carry produced by $((L(x) \oplus R'(a_2)) \boxplus (L(y) \oplus R'(b_2))$. we can again write $(\overleftarrow{x} \oplus a_2) \boxplus (\overleftarrow{y} \oplus b_2) \oplus \Delta_2$ from (3.7) as the concatenation of its left and right parts:

$$(\overleftarrow{x} \oplus a_2) \boxplus (\overleftarrow{y} \oplus b_2) \oplus \Delta_2 =$$

$$((\overleftarrow{L(x)||R(x)}) \oplus a_2) \boxplus ((\overleftarrow{L(y)||R(y)}) \oplus b_2) \oplus \Delta_2 =$$

$$((R(x)||L(x)) \oplus (L'(a_2)||R'(a_2))) \boxplus ((R(y)||L(y)) \oplus (L'(b_2)||R'(b_2))) \oplus \Delta_2 =$$

$$((R(x) \oplus L'(a_2)) \boxplus (R(y) \oplus L'(b_2)) \boxplus C^2_\gamma) \oplus L'(\Delta_2)||$$

$$((L(x) \oplus R'(a_2)) \boxplus (L(y) \oplus R'(b_2))) \oplus R'(\Delta_2).$$

We get that

$$\overleftarrow{(x \oplus a_1) \boxplus (y \oplus b_1) \oplus \Delta_1} = (\overleftarrow{x} \oplus a_2) \boxplus (\overleftarrow{y} \oplus b_2) \oplus \Delta_2$$

if and only if

$$\begin{aligned} &(R(x) \oplus R(a_1)) \boxplus (R(y) \oplus R(b_1)) \oplus R(\Delta_1) = \\ &((R(x) \oplus L'(a_2)) \boxplus (R(y) \oplus L'(b_2)) \boxplus C^2_\gamma) \oplus L'(\Delta_2), \end{aligned} \tag{3.8}$$

and

$$\begin{aligned}&((L(x) \oplus L(a_1)) \boxplus (L(y) \oplus L(b_1)) \boxplus C^1_{n-\gamma}) \oplus L(\Delta_1) = \\ &((L(x) \oplus R^{'}(a_2)) \boxplus (L(y) \oplus R^{'}(b_2))) \oplus R^{'}(\Delta_2).\end{aligned} \tag{3.9}$$

Substituting

$$R(x^{\dagger}) = R(x) \oplus L^{'}(a_2)$$

$$R(y^{\dagger}) = R(y) \oplus L^{'}(b_2),$$

we can rewrite (3.8) as

$$\begin{aligned}&R(x^{\dagger}) \boxplus R(y^{\dagger}) \boxplus C^2_{\gamma} = \\ &(R(x^{\dagger}) \oplus L^{'}(a_2) \oplus R(a_1)) \boxplus (R(y^{\dagger}) \oplus L^{'}(b_2) \oplus R(b_1)) \oplus R(\Delta_1) \oplus L^{'}(\Delta_2).\end{aligned} \tag{3.10}$$

Similarly, by setting

$$L(x^*) = L(x) \oplus L(a_1)$$

$$L(y^*) = L(y) \oplus L(b_1),$$

(3.9) reduces to

$$\begin{aligned}&L(x^*) \boxplus L(y^*) \boxplus C^1_{n-\gamma} = \\ &((L(x^*) \oplus L(a_1) \oplus R^{'}(a_2)) \boxplus (L(y^*) \oplus L(b_1) \oplus R^{'}(b_2))) \oplus R^{'}(\Delta_2) \oplus L(\Delta_1).\end{aligned} \tag{3.11}$$

Now, we can compute the probabilities of (3.10) and (3.11) based on the values of $C^1_{n-\gamma}$ and C^2_{γ} using Lemmata 9–10:

Case 1: $C^2_{\gamma} = 0$, the probability is the difference propagation probability and can be calculated using Lemma 9.

Case 2: $C^2_{\gamma} = 1$, we solve the differential equations using Lemma 10.

Similarly,

Case 3: $C^1_{n-\gamma} = 0$, the probability is the difference propagation probability and can be calculated by Lemma 9.

Case 4: $C^1_{n-\gamma} = 1$, we solve the differential equations using Lemma 10.

When $\gamma = 1$, $L(\cdot), R^{'}(\cdot)$ represent a single bit, hence,

$$C^1_{n-\gamma} = L(a_1) \oplus L(b_1) \oplus L(\Delta_1) \oplus R^{'}(a_2) \oplus R^{'}(b_2) \oplus R^{'}(\Delta_2).$$

In addition, we note that the carry bit of $L(x) \boxplus L(y)$ is independent of the carry bit of $R(x) \boxplus R(y)$ when x and y are independent random variables. For large n and $\gamma = 1$ we have $\Pr[C^2_\gamma = 0] = 3/4$ and $\Pr[C^1_{n-\gamma} = 0] = 1/2$. Then,

$$\Pr[C^2_\gamma = 0, C^1_{n-\gamma} = 0] = 2^{-1.415}$$

$$\Pr[C^2_\gamma = 0, C^1_{n-\gamma} = 1] = 2^{-1.415}$$

$$\Pr[C^2_\gamma = 1, C^1_{n-\gamma} = 0] = 2^{-3}$$

$$\Pr[C^2_\gamma = 1, C^1_{n-\gamma} = 1] = 2^{-3}.$$

Therefore, the probability is calculated as

$$\Pr[C^2_\gamma = 0, C^1_{n-\gamma}] \cdot \Pr[x \boxplus y = (x \oplus \delta_1) \boxplus (y \oplus \delta_2) \oplus \delta_3] +$$

$$\Pr[C^2_\gamma = 1, C^1_{n-\gamma}] \cdot \Pr[x \boxplus y \boxplus 1 = (x \oplus \delta_1) \boxplus (y \oplus \delta_2) \oplus \delta_3],$$

which concludes the proof. □

The above theorem shows the propagation of RX-differences through modular addition and how to compute its probability. Given inputs $(x \oplus a_1, y \oplus b_1)$ and $(\overleftarrow{x} \oplus a_2, \overleftarrow{y} \oplus b_2)$ with RX-differences $(((a_1, a_2), 1), ((b_1, b_2), 1))$, let $z = ((x \oplus a_1) \boxplus (y \oplus b_1))$ and $z^{'} = ((\overleftarrow{x} \oplus a_2) \boxplus (\overleftarrow{y} \oplus b_2))$. Theorem 5 predicts the probability that z, z' form a $((0, \overleftarrow{\Delta_1} \oplus \Delta_2), 1)$-RX-difference.

3.2 Linear Cryptanalysis Using Multiple Low-bias Approximations

In this section we describe a set of methods for executing a (multi-)linear attack using linear approximations with small biases. Such linear approximations were previously considered unusable because they could not be detected by the classical normal distinguisher discussed in Section 1.1.6. Indeed, one limitation of this classical distinguisher is that it cannot use biases with magnitude smaller than a certain threshold (The threshold is often set to $2^{-n/2}$). This is used to "prove" the resistance of new designs against linear cryptanalysis by claiming

that since no linear approximation with an absolute bias larger than $2^{-n/2}$ exists, no attack can be mounted with a non-negligible success probability. In this section we show how to use linear approximations with arbitrarily low absolute biases (*i.e.*, below $2^{-n/2}$) to overcome this limitation.

We start this section by restating the χ^2 distinguisher which was previously presented in [156] and show that in the classical setting (*i.e.*, with large biases) it is equivalent to the classical normal distinguisher. Then, we show that unlike the classical distinguisher, the χ^2 distinguisher can be used to combine data from multiple sources to avoid data limitations and hence it can be used to detect smaller biases than the classical one. We present this work in two settings:

- A χ^2 linear distinguisher for arbitrarily small biases using a single approximation and multiple keys; and
- a χ^2 linear distinguisher for arbitrarily small biases using multiple approximations and a single key.

3.2.1 Related Work

Interestingly, [36] also mentions that linear approximations with low-bias can be used for an attack through the variance of the distribution. However, they discuss this in the context of a key recovery attack using a single approximation. While theirs is an interesting observation, it still relies on several unrealistic assumptions. First, it assumes that sufficient knowledge of the ELP is given, which implies that the most dominant trails are known, if not the full distribution. More importantly, it assumes that the ELP is always larger than the bias variance of a random approximation, which allows them to define an interval $[-\Theta, \Theta]$ and claim that keys outside this interval are more likely to be right keys. However, it can readily be shown that this is not always the case, which means that in some scenarios (*a priori* unknown due to the difficulty of calculating the ELP), keys inside the interval $[-\Theta, \Theta]$ are actually more likely to be the right keys.

A recent paper [143] takes a different approach for obtaining the distribution of biases, by trying to bound them. They explicitly criticize prevailing methods using order statistics, due to the reliance of such methods on assumptions regarding the distribution of the bias. As a result, they question the applicability of formulae formerly used to derive the data complexity. Clearly, [36] and [143] (as well as others) are in disagreement as to some fundamental aspects of linear cryptanalysis.

Our approach

To avoid the minefield that bias estimation seems to be, we take an approach that is based on the foundation of statistical cryptanalysis, namely, the behavior of random oracles. Our null hypothesis is that a set of observations came from a random oracle, and we use a statistical test that does not require an alternative hypothesis. In doing so, we avoid the need to obtain information about the bias distribution of the specific cipher under consideration.

This is not to say that obtaining the bias distribution is unimportant. When the distribution is known it allows to calculate the ELP, which can then be used to calculate the data requirement and the advantage of an attack. However, being orthogonal to the bias estimation problem, our approach has merits in real world scenarios, even if the bias's distribution is unknown. Since both parameters are not required *a priori* to execute the attack, an adversary can simply test the attack offline (*e.g.*, in a lab), establish how much data is required, then execute it on an online target.

In addition, as per the discussion in Section 2.5.4 and for brevity's sake, we avoid the estimation of the underlying bias through the sample bias using a sample of the codebook. Instead, we use the entire codebook and always obtain the underlying bias directly. We stress that this is not inherent to the method we present here. When combined with [7,35,42], the behavior of a sample bias using a sample of the codebook can directly be obtained. Similarly, our multi-linear distinguisher presented in Section 3.2.5 can be used for a key recovery attack using Matsui's Algorithm 2 in the same manner that such distinguishers are always used even if only part of the codebook is being sampled. A key recovery attack using the multi-key distinguisher of Section 3.2.4 is more complicated, and is left for future research.

3.2.2 The χ^2 Distinguisher

We now present the χ^2 distinguisher and demonstrate that in the classical setting (*i.e.*, for a single high-bias approximation) this distinguisher is as efficient as the classical linear distinguisher.

Recalling Corollary 1, we can convert the normal variable $\mathbf{X}$ into a χ_1^2 variable. The $[-2^{-n/2}, 2^{-n/2}]$ interval which was 2 standard deviations from the mean in each direction (*cf.* Section 1.1.6) now becomes a $[0, 4]$ interval. Computing $F_{\chi_1^2}(4) = 0.95449$ where $F_{\chi_1^2}$ is the Cumulative Distribution Function of the χ_1^2 distribution shows that the two intervals cover an area of the same size. For a

Table 3.2: Bounds for 90% confidence intervals of the χ^2 family of probability distributions. m is the number of degrees of freedom, a is the lower bound of the interval and b is its upper bound. For example, $\Pr[0 \leq \mathbf{X} \leq 5.02 | m = 1] = 0.9$.

m	a	b
1	0	3.84
2	0.1	5.99
4	0.71	9.49
128	102.87	155.4
256	219.95	294.32

Table 3.3: The linear approximations used to verify the equivalence between the classical linear distinguisher and the χ^2 distinguisher. The difference between the trail bias predicted by [161] and the measured bias is due to the linear hull effect. All masks are reported in hexadecimal notation.

No. rounds	Input mask (left,right)	Output mask (left,right)	Trail bias reported in [161]	Measured average absolute bias (ELP)
6	(000D, 56E0)	(0800, 0800)	2^{-8}	2^{-10} ($2^{-17.96}$)
7	(000D, 56E0)	(2040, 2050)	2^{-10}	2^{-12} ($2^{-21.96}$)
8	(000D, 56E0)	(0083, 80C3)	2^{-13}	2^{-15} ($2^{-27.87}$)
9	(000D, 56E0)	(170B, 130A)	2^{-15}	$2^{-15.98}$ ($2^{-29.64}$)

list of confidence intervals for the χ^2 family that are used later in this Thesis, see Table 3.2.

We have also verified this equivalence empirically using versions of Speck32/64 reduced to 6–9 rounds. The linear approximations we used were first published in [161]. We present these approximations in Table 3.3, and report the average absolute bias of the hull which we have obtained empirically. In each experiment we used 2^{11} random keys to encrypt 2^{32} plaintexts for 6–9 rounds and applied the respective linear approximations from Table 3.3. For each key we computed the sample bias, and used both the classical distinguisher and the χ^2 distinguisher. For the classical distinguisher, we checked how many of the 2^{11} experiments have an absolute bias larger than 2^{-16}. For the χ^2 distinguisher, we checked how many times the test statistic was outside the interval $[0, 4]$. Table 3.4 reports the results, which, as expected, show that the two distinguishers are the same.

Table 3.4: Comparing the two distinguishers: we see that the two distinguishers produce exactly the same results and are in fact equivalent.

No. rounds	Bias	No. Successes for the	
		classical Linear distinguisher	χ^2 distinguisher
6	2^{-10}	2048	2048
7	2^{-12}	2048	2048
8	2^{-15}	1900	1900
9	$2^{-15.98}$	1020	1020

3.2.3 Using Datasets from Different Sources

Recall that Corollary 1 allows to convert a counter $\hat{T}$ following a normal distribution into a χ_1^2-variable. Then, since the sum of m independent χ_1^2 variables is distributed according to χ_m^2, we obtain the following:

Corollary 2. *Let* $\hat{\mathbf{T}}_\mathbf{0}, \ldots, \hat{\mathbf{T}}_\mathbf{m-1}$ *be normal independent random variables with*

$$\hat{\mathbf{T}}_\mathbf{0}, \ldots, \hat{\mathbf{T}}_\mathbf{m-1} \sim \mathcal{N}(2^{n-1}, 2^{n-2}).$$

Then a test statistic $\mathbf{T}$ *defined as*

$$\mathbf{T} = \sum_{i=0}^{m-1} \left(\frac{\hat{\mathbf{T}}_\mathbf{i} - 2^{n-1}}{2^{n/2-1}}\right)^2 \tag{3.12}$$

follows the χ_m^2 *distribution.*

Noting that $\mathcal{N}(2^{n-1}, 2^{n-2})$ is exactly the distribution of counters obtained from applying a linear approximation to a random permutation, we can use the χ^2 distinguisher in two scenarios which we soon describe. The intuition in both cases is that although a single counter is insufficient to distinguish a biased distribution from a random one, combining multiple counters increases the statistical distance. In other words, we use a series of non-significant observations to create a significant one. This is inherently different from prior works which combined a series of significant observations to reduce the data complexity.

3.2.4 The Multi-key Setting

The first case we consider is the multi-key scenario. In this setting, data is encrypted using the same family of block ciphers with multiple uniformly selected keys $k_0, \ldots, k_{m-1}$. Under our assumption of using the entire codebook, this means that each of the m keys is used to encrypt 2^n plaintexts. This results in a set of m counters, $\hat{T}_0, \ldots, \hat{T}_{m-1}$, where each counter $\hat{T}_i$ counts the number of times the linear approximation ψ holds for data encrypted under k_i (the same ψ is used for all counters). The goal of the adversary is to distinguish between this case and a case where the data was obtained using m independent random permutations.

Lemma 11. *Let $\pi_0, \ldots, \pi_{m-1}$ be a set of m random permutations, ψ a linear approximation, and $\hat{T}_0, \ldots, \hat{T}_{m-1}$ a set of counters such that $\hat{T}_i$ counts the number of times ψ was satisfied when applied to π_i after using the entire codebook. Then, the test statistic $\mathbf{T}$ which is defined as*

$$\mathbf{T} = \sum_{i=0}^{m-1} \left(\frac{\hat{T}_i - 2^{n-1}}{2^{n/2-1}} \right)^2 \tag{3.13}$$

has a χ^2_m distribution.

Proof. The counters $\hat{T}_0, \ldots, \hat{T}_{m-1}$ are associated with the random permutations $\pi_0, \ldots, \pi_{m-1}$. Since random permutations are independent, the counters are also independent. As per [60] each counter $\hat{T}_i$ is normally distributed according to Lemma 4. Using Corollary 2 on these independent and normally distributed random variables completes the proof. □

The reason $\mathbf{T}$ is modeled as a random variable is that in both cases, the counters $\hat{T}_0, \ldots, \hat{T}_{m-1}$ follow a certain distribution. In the case of a set of random permutations, each value is drawn according to Lemma 4. Similarly, for the case of a block cipher, each counter $\hat{T}_i$ is drawn according to the unknown distribution governing the underlying bias (*cf.* Section 2.5).

With that in mind, we can build a confidence interval of size α for χ^2_m and use it for hypothesis testing. An adversary receives a set of counters $\hat{T}_0, \ldots, \hat{T}_{m-1}$ and needs to decide if they were obtained from a set of random permutations $\pi_0, \ldots, \pi_{m-1}$ or from a family of block ciphers with m different keys, *i.e.*, $E_{k_0}, \ldots, E_{k_{m-1}}$. The adversary builds a test statistic $\mathbf{T}$ according to Lemma 11 and checks if it falls inside the confidence interval. The null hypothesis H_0 is that the counters were obtained using m random permutations, and therefore $\mathbf{T} \sim \chi^2_m$. The alternative hypothesis H_1 is that they were not. The adversary

builds a confidence interval $[a, b]$ of size α for χ^2_m, meaning that if H_0 is true, then $\Pr[a \leq T \leq b] = \alpha$. As is inherent in any hypothesis testing based attack (such as the classical linear attack), the rate of false positives is $1 - \alpha$.

The fact that we use the χ^2 distribution has major benefits over the classical distinguisher. First, by squaring, it avoids sign considerations that are inherent to classical linear cryptanalysis. Moreover, since the average bias of a random permutation as well as a biased one is 0, using more approximations in the classical setting only makes it more similar to the random case. When the random case is modeled as a distribution from the χ^2 family, the mean of that distribution increases as more degrees of freedom are added (*i.e.*, when more counters are used), while a test statistic calculated from a block cipher tends to the second moment of the unknown right bias distribution, *i.e.*, since the mean is 0, to its variance (ELP). Hence, by using a *double-sided* χ^2 *test* we can detect biases **lagging behind the** χ^2 **distribution**.

3.2.5 The Multi-linear Setting

The same arguments in favor of the χ^2 distinguisher that were presented in the previous section can be made for using multiple linear approximations with low biases. However, it is hard to argue that multiple linear approximations relating to the same random permutation are statistically independent. In [124] Nyberg showed, based on the Xiao-Massey Lemma, that a set of linear approximations is statistically independent if and only if they are linearly independent. However, this condition seems too strict, and it might be enough that the approximations are "sufficiently" independent. The adaptation of Lemma 11 to the case of multiple linear approximations is therefore given below only as a hypothesis and left unproven. In Section 3.2.6 we show through experiments that a certain level of dependency can be tolerated.

Hypothesis 3. *Let π be a random permutation, and let $\psi_0, \ldots, \psi_{m-1}$ be a set of m linear approximation, and $\hat{T}_0, \ldots, \hat{T}_{m-1}$ a set of counters such that $\hat{T}_i$ counts the number of times ψ_i was satisfied when applied to π after using the entire codebook. Then, the test statistic* $\mathbf{T}$ *which is defined as*

$$\mathbf{T} = \sum_{i=0}^{m-1} \left(\frac{\hat{T}_i - 2^{n-1}}{2^{n/2-1}} \right)^2 \tag{3.14}$$

has a χ^2_m distribution.

Table 3.5: The 9-round linear approximations used in our experiments

No. rounds	Trail number	Input mask (left,right)	Output mask (left,right)	Trail bias reported in [161]	Measured bias
9	1	(000D, 56E0)	(170B, 130A)	2^{-15}	$2^{-15.98}$
9	2	(000D, 56E0)	(1D0B, 1B0A)	2^{-15}	$2^{-15.97}$

3.2.6 Experimental Verification

In this section we present an experimental verification of our distinguishers. We use SPECK32/64 to present several linear distinguishers for 9 and 10 rounds.

Multi-key distinguishers for Speck

We start discussing our results with a sanity check. In addition to the 9-round linear trail presented by Yao *et al.*, we found another trail of the same length and the same bias. Both trails are presented in Table 3.5. We conducted 2048 experiments with each of the trails, using a confidence interval of size $\alpha = 0.9$. A success in a single experiment is defined to be "the test statistic falls outside the confidence interval" which, for a random variable, should happen with probability 0.1.

When using a single key with a single trail, the number of successes in 2048 experiments is 1069 for Trail 1 and 1104 for Trail 2, corresponding to 52.2% of the experiments and 53.9%, respectively.[2]

When setting $m = 2$ and creating a distinguisher based on multiple keys by combining the zeroth key with the first key, the second with the third, *etc.* we get that the number of successes (out of 1024) is 749 for Trail 1 and 740 for Trail 2 corresponding to 73.1% of the experiments and 72.3%, respectively.

For completeness, we model the expected number of successes in a random permutation over N experiments where the success probability is 0.1 (corresponding to the false positive rate) by

$$W \sim \mathcal{B}(N, 0.1). \tag{3.15}$$

[2]Note that the difference from Table 3.4 is due to the different size of the confidence interval used in the distinguisher.

Table 3.6: Results of the multi-key distinguisher for 9 rounds. The probability reported inside the parentheses is the probability that a binomial random variable will have the reported number of successes or more. We see that the success rate of the distinguisher improves as we consider more approximations obtained using different keys.

m	No. of Experiments	Trail	No. of successes (prob.)
1	2048	1	1069 ($< 2^{-53}$)
	2048	2	1104 ($< 2^{-53}$)
2	1024	1	749 ($< 2^{-53}$)
	1024	2	740 ($< 2^{-53}$)

Fixing N, we can compute the probability that W takes a value which is the number of successes x_i or higher, *i.e.*, $p = \Pr[W \geq x_i]$. The results are summarized in Table 3.6.

We now turn to verify our 10-round distinguisher based on multiple keys. Using the 9-round trails as a basis, we extended each of them into 128 10-round trails using the automated tool presented in Section 4.1. The result is a collection of 256 linear trails. This collection is viewed as two sets of size 128, where the elements in the first set are those linear approximations extended from Trail 1 and the elements of the second set are those linear approximations extended from Trail 2.

We first start by applying the distinguisher to each of the approximations independently, and check if it falls inside the confidence interval induced by χ^2_1. Conducting $2048 \cdot 256 = 524,288$ experiments, the test statistic falls outside the confidence interval 52,215 times, corresponding to about 10% of the experiments. This matches quite closely the expected number of false positives which is $524,288 \cdot 0.1 = 52,428$. We therefore see that each individual approximation cannot be distinguished from a random one.

When setting $m = 256$ (*i.e.*, each approximation is evaluated against data from 256 different keys), the number of successes over 2048 experiments is 224 corresponding to 10.9% of the experiments. The probability that a random binomial variable with probability 0.1 will have 224 or more success in $N = 2048$

Table 3.7: Results of the multi-key distinguisher for 10 rounds. The probability reported inside the parentheses is the probability that a binomial random variable will have the reported number of successes or more. We see that each individual approximation is insufficient for constructing a linear distinguisher, but that a collection of 256 is not.

m	No. of Experiments	Trail	No. of successes (Prob.)
1	524,288	1 + 2	52,215 (= 0.838)
256	2048	1 + 2	224 (= 0.085)

experiments is given by

$$W \sim \mathcal{B}(2048, 0.1)$$

$$\Pr[W \geq 224] = 0.085$$

The results are summarized in Table 3.7.

We see that the success rate of the distinguisher improves as we add more data coming from different keys using the same linear approximation.

Multiple linear cryptanalysis using low-bias approximations

We now present a way to use our distinguisher for multiple approximations, where all the approximations have a bias smaller than $2^{-n/2}$. As before, we start with a sanity check using our 9-round linear approximations. As we saw in Section 3.2.6, when using $\alpha = 0.9$ and $m = 1$, the success rate is 1069 for Trail 1 and 1104 for Trail 2, corresponding to 52.2% of the experiments and 53.9%, respectively.[3] When combining both trails, *i.e.*, setting $m = 2$, the number of successes in 2048 experiments increases to 1429, corresponding to 69.8% of the experiments. The results, as well as the probability to have this number of successes *à la* (3.15) are presented in Table 3.8.

For a 10-round distinguisher for SPECK32 based on multiple linear approximations we used the previously described 256 10-round trails. Setting $\alpha = 0.9$ and $m = 128$ we get that the number of successes in 2048 experiments is 232 when

[3]These numbers are the same as in Section 3.2.6 as they describe the same experiment.

Table 3.8: Results of a distinguisher using multiple approximations for 9 and 10 rounds. The probability reported inside the parentheses is the probability that a binomial random variable will have the reported number of successes or more.

No. Rounds	m	No. of Experiments	Trail	No. of successes (Prob.)
9	1	2048	1	1069 ($< 2^{-53}$)
		2048	2	1104 ($< 2^{-53}$)
9	2	2048	1+2	1429 ($< 2^{-53}$)
10	1	524,288	1 + 2	52,215 (= 0.838)
10	128	2048	1	232 (= 0.026)
		2048	2	222 (= 0.110)
10	256	2048	1 + 2	223 (= 0.097)
10	128+128	2048	1 or 2	438 (= 0.004)

using Trail 1, and 222 when using Trail 2, corresponding to 11.3% and 10.8%, respectively.

Interestingly, when setting $m = 256$ and using the test statistic over all 256 linear approximations, the obtained result is 223 (10.9%), which is still better than using a single trail, but is not as significant as the theory predicts. We conjecture that this behavior is due to dependency between the biases of the two underlying 9-round approximations and leave it for future research.

Instead, we build two test statistics T_1, and T_2, for each group of 10-round linear approximations, and consider the experiments successful if either of the statistics falls outside the confidence interval. We get that when using this test for Speck32, the number of successes in 2048 experiments is 438 (21.4%), wheres the expected false positive rate in this case is $0.1 + 0.1 - 0.1^2 = 0.19$. A summary of these results is presented in Table 3.8.

3.3 Summary and Future Work

In this chapter we developed new cryptanalytic techniques. In the first contribution we showed that the injection of round constants is not a suitable defense against rotational cryptanalysis. We abstracted rotational

and differential cryptanalysis into RX-cryptanalysis, and presented a way to calculate the propagation probability of RX-characteristics.

In the second contribution, we showed that contrary to folklore belief (see *e.g.*, the discussion in [59, Sect. 9.1.1]), linear approximations with low-biases can be used for a linear attack. We recalled a distinguishing method based on the χ^2 distribution, and showed how to use it with multiple low-bias linear approximations to obtain a distinguisher.

Future work

All contributions in the last two chapters stem from incomplete or incorrect models. This is highly unfortunate as linear cryptanalysis is considered by many to be one of the two most important cryptanalytic techniques. As these two chapters show, linear cryptanalysis is still not well understood. The models seem to be good enough in many practical examples, but they definitely do not cover all corners. These cases are hard to identify because it is often the case that an attack is presented only in its theoretical form because it is too expensive to verify. Indeed, it is easy to find examples where attacks with prohibitive complexities are published without any reservations and assuming that everything will work the way the theory predicts, but attacks involving experimental verification do not fully match the predictions.

Future work on linear cryptanalysis should focus on reviewing the models used. The exact conditions for using Matsui's Algorithm 1 should be exposed, and the practice of using the bias of a linear trail for an attack should be abandoned. For this to happen, better techniques for finding the bias of linear hulls must be developed.[4]

However, bias estimation is only one side of the problem. The other side is that bounding the bias does not provide guarantees against linear cryptanalysis. As we showed in this chapter, multiple linear approximations can still be used for an attack, even if each of them has an absolute bias below $2^{-n/2}$. This work is still at its beginning as we only show how to use this property for a distinguishing attack. A key recovery attack for the multi-linear distinguisher seems to be straightforward, but it is still unclear how key recovery in the multi-key case would work. On the other hand, further work is required to understand how the dependencies between different approximations affect the multi-linear distinguisher; especially in light of Parseval's Theorem.

[4]While this is not always a common view (in many fields of life), this Author believes, on principle, that there can be no good reason to use something known to be incorrect just because the right way was not yet discovered.

The understanding that non-monotonic success probability is encountered when the bias for the right key is ranked below more wrong-key-biases than what the advantage allows opens up avenues for improvements in how linear cryptanalysis is being used. For example, we can redefine the advantage such that it is achieved if the right key is amongst a collection of successively ranked keys, not necessarily the first ones.

Similar to linear cryptanalysis, rotational cryptanalysis can be further improved. The current work only considers round constants injected using the XOR operation. A work in progress aims to extend this to round constants injected using modular addition (such is, for example, the case in Threefish, the block cipher underlying the hash function Skein [71]). Another apparent limitation is that this work only considers rotational pairs with rotation offset of $\gamma = 1$ and future work may extend RX-cryptanalysis to $\gamma > 1$. However, it is important to note that $\gamma \neq 1$ was traditionally used in cases when the round constant was rotation invariant with respect to the used γ. With the integration of round constants offered in this chapter, there might never be a need to use $\gamma \neq 1$.

There are two more reasons why rotational cryptanalysis is not regarded as highly as differential and linear cryptanalyses: (i) it is only applicable to ARX constructions, and (ii) it requires two related keys. However, it was already shown before that any function, and in particular any block cipher, can be written using the ARX-C operations. Now that rotational cryptanalysis is extended to cover ARX-C rather than just ARX, it might be beneficial to consider the representation of non-ARX constructions as ARX-C. Similarly, there are indications that RX-cryptanalysis can be made to work in the single key model.

One last area of research that seems to have been neglected in recent years is the development of new cryptanalytic techniques. This Author proposes to do this in two parallel tracks. The first is by combining existing techniques. Differential-linear cryptanalysis [105] is a good example for the potential of such approach. This Author already explored combining rotational cryptanalysis with differential cryptanalysis, rotational cryptanalysis with linear cryptanalysis, and integral cryptanalysis with linear cryptanalysis; yet never in a systematic way. The second approach is to come up with completely new techniques either inspired by existing ones or unrelated.

Chapter 4

Automated Tools for Cryptanalysis

"And once again he felt annoyance with the Universe for making something both essential and unpleasant."

- Isaac Asimov, *The Robots of Dawn*

The core of statistical cryptanalysis techniques such as those discussed in the previous chapters is the existence of undesirable relations between a plaintext, its ciphertext, and possibly the secret key. This chapter is concerned with the automatic search of such relations.

In Section 4.1 we deal with the automated search of good linear trails in ARX constructions. The search strategy reduces the search space by focusing on linear masks that are likely to result in such trails. For each such mask, the tool follows its propagation both forward and backwards, until certain constraints are violated. This work was Daniël Bodden's master thesis [38] under the supervision of this Author. It was published in and won the best student paper award of [8].

The work is included in this Thesis for the sake of completeness as part of the work this Author did during his Ph.D term. Its application became severely limited by this Author's observation that unlike their differential counterparts, linear trails cannot be used to lower bound the bias of a linear hull, and in light of the results presented in Section 2.2.

In Section 4.2 we develop an automated tool for searching RX-characteristics. The tool encodes the constraints for a valid RX-transition and uses a SAT/SMT solver to find characteristics satisfying all the constraints. This work was Glenn De Witte's master thesis [64], for which he won the *Vasco Data Security* prize, under the supervision of Yunwen Liu and this Author. It was published in [10]. De Witte's original work focused on devising the constraints for the block cipher SPECK and was later generalized by Adrián Ranea in his master thesis [132] (co-supervised by Yunwen Liu and this Author) where he developed a new Python-like language for describing ARX ciphers, and a Python parser converting this language into a set of constraints. Ranea's work was published in [133]. Both works were used to find the RX-characteristics presented in Section 5.3. These RX-characteristics were published in [109].

4.1 An Automated Tool for Searching Linear Trails in ARX Constructions

This section deals with an automated search for linear trails in ARX constructions. The search strategy is based on [52, Cor. 1] where it is shown that an input mask containing only a pair of consecutive bits propagates with bias $\frac{1}{4}$ through modular addition to an output mask containing the same bits. This result was later extended in [11] which showed that a sequence of ℓ' consecutive bits can be treated as $\ell = \frac{\ell'}{2}$ independent pairs and their joint bias can be calculated using the Piling Up Lemma. Focusing on these "consecutive pairs" trails allows to greatly reduce the search space by considering only probable candidates.

The tool receives an *OpenCL core*[1] with a description of the cipher to be analyzed and an integer ℓ limiting the search space. The program starts by testing all masks with a single pair of consecutive bits, two pairs of consecutive bits, up to ℓ pairs of consecutive bits. For each starting point it tries to propagate the mask both forward and backwards for as many rounds as possible, using the propagation rules of [28]. The search stops when one of the following events occur:

- The rotation operation splits two consecutive bits to the most- and least-significant positions; or
- the XOR operation creates odd-length sequence of bits (*e.g.*, $0110 \oplus 0011 = 0101$), which can not be further propagated using [52, Cor. 1]; or

[1] OpenCL is a programming language with a C-like syntax developed for writing parallel applications. A core is a unit of software, much like a function in other programming languages.

– the trail bias falls below $2^{-n/2}$ where n is the block size.

The output of the tool is the longest linear trail it could find without violating these constraints.

To test the tool, we executed it on versions of SPECK using a 40-core 3.1 GHz Intel Xeon. For each version with block size n, we ran the tool with the highest possible $\ell = \frac{n}{2}$. The execution time was less than 10 milliseconds for SPECK32 and less than a week for SPECK128, with all other versions ranging in between.

The tool returned linear trails for 7, 8, 11, 10, and 11 rounds for versions of SPECK with block sizes of 32, 48, 64, 96, and 128, respectively. These trails were not the best linear trails published for SPECK. The best linear trails at the time were published in [161] and involved a few odd-length sequences in intermediate states, suggesting that searching only through even-length sequences is not an optimal strategy.

4.2 An Automated Tool for Searching RX-characteristics in ARX Constructions

This section deals with an automated search tool for RX-characteristics. The tool extends the work of Section 3.1 by encoding the RX-transition constraints and searching for an optimal RX-characteristic. We start in Section 4.2.1 by describing the *Boolean Satisfiability Problem* and how SAT-solvers can be used for cryptanalysis. Then, in Section 4.2.2 we show how to encode an ARX-based cryptosystem as a SAT/SMT problem. In Section 4.2.3 we discuss possible search strategies for the tool. We continue by presenting in Section 4.2.4 a language for describing an ARX-based system in a Python-like syntax, and a parser for this language translating it into a SAT/SMT problem. The results of the execution of this tool on the block cipher SPECK are presented in Section 5.3.

In the rest of this chapter, we use the notation of Section 3.1.

4.2.1 The Boolean Satisfiability Problem

A *Boolean formula* is an expression consisting of Boolean variables taking the values TRUE or FALSE, and the logical operators AND, OR and NOT. A Boolean formula is said to be *satisfiable* if there exists an assignment for its variables that makes it evaluate to TRUE. For example the Boolean formula

$$a \text{ AND } (\text{NOT } b)$$

is satisfiable because it returns TRUE when evaluated on the input $(a, b) =$ (TRUE, FALSE).

The Boolean satisfiability (SAT) problem is the problem of determining whether a Boolean formula is satisfiable. In general, the SAT problem is NP-complete [54], which implies that no known algorithm solves a general SAT instance in polynomial time (with respect to the number of variables). In practice, SAT solvers can handle instances with thousands (and sometimes even millions) of variables [162].

A generalization of the SAT problem is the satisfiability modulo theories (SMT) problem. SMT formulae can be expressed with richer languages (theories) than Boolean formulae. In particular, a formula in the bit-vector theory can contain bit-vectors (a vector of Boolean variables) and the usual operations of bit-vectors such as bitwise operations (*e.g.*, XOR, OR, and AND) arithmetic operations (*e.g.*, addition and multiplication), *etc.* A common approach in SMT solvers [44, 75] is to translate the SMT instance into a SAT instance and solve it using a SAT solver.

In addition to richer languages, SMT solvers also support an *objective function*. This function is an additional constraint forcing a variable to satisfy certain conditions. For example, through an objective function, an adversary can ask the solver for solutions not exceeding some probability for the RX-characteristics.

4.2.2 Automated Search for RX-characteristics

A triplet of RX-differences (α, β, ζ) is said to be *valid* if α and β propagate to ζ through $\boxplus$ with non-zero probability, that is, $\Pr[\alpha, \beta \xrightarrow{\boxplus} \zeta] \neq 0$. Using Theorem 5, the triplet (α, β, ζ) is valid if and only if one of the following conditions holds

$$(I \oplus SHL)(\delta_\alpha \oplus \delta_\beta \oplus \delta_\zeta) \preceq SHL((\delta_\alpha \oplus \delta_\zeta)|(\delta_\beta \oplus \delta_\zeta)) \tag{4.1}$$

$$(I \oplus SHL)(\delta_\alpha \oplus \delta_\beta \oplus \delta_\zeta) \oplus 1 \preceq SHL((\delta_\alpha \oplus \delta_\zeta)|(\delta_\beta \oplus \delta_\zeta)). \tag{4.2}$$

The *weight* ω *of a valid triplet* (α, β, ζ) is defined as

$$\omega(\alpha, \beta, \zeta) = -\log_2(\Pr[\alpha, \beta \xrightarrow{\boxplus} \zeta]).$$

Using Theorem 5, the weight can be calculated as follows

$$\omega(\alpha, \beta, \zeta) = \begin{cases} |SHL((\delta_\alpha \oplus \delta_\zeta)|(\delta_\beta \oplus \delta_\zeta))| + 1.415, & \text{if (4.1) holds} \\ |SHL((\delta_\alpha \oplus \delta_\zeta)|(\delta_\beta \oplus \delta_\zeta))| + 3, & \text{if (4.2) holds.} \end{cases} \tag{4.3}$$

The *weight W of an RX-characteristic* is defined as the sum of the weights for each transition (noting that in ARX, only transitions through modular addition incur weight penalty). As in [118], it is assumed that the probability of a characteristic is the multiplication of the probabilities of each modular addition. In this case, the probability of an RX-characteristic p can be calculated as $p = 2^{-W}$.

The main idea of this technique is to use a SAT/SMT solver to determine whether there exists an RX-characteristic up to a certain weight W. If the SAT/SMT solver concludes that the problem is satisfiable, a lower weight is chosen. Otherwise, a higher weight is chosen. This is repeated until the *minimal weight* is found, that is, a weight $\widehat{W}$ such that there exists an RX-characteristic up to weight $\widehat{W}$ but not up to weight $\widehat{W} - \epsilon$, where ϵ is a predetermined value (usually $\epsilon = 1$, *e.g.*, [118]).

SAT/SMT solvers not only determine whether a formula is satisfiable, but also obtain an assignment that makes the formula TRUE if it is. Therefore, the result of this method is a minimal weight $\widehat{W}$, an RX-characteristic with probability in the interval $(\widehat{W} - \epsilon, \widehat{W}]$ and the knowledge that there is no RX-characteristic with probability better that $\widehat{W} - \epsilon$.

Since the RX-differences of the round keys are necessary to propagate the RX-differences through the encryption function, a pair of characteristics is actually considered: one for the key-schedule and one for the encryption. The RX-differences predicted by the key-schedule characteristic are used in the encryption characteristic as the RX-differences of the round keys.

An important step of this technique is the formulation of the decision problem of whether there exists a pair of RX-characteristics up to a certain weight as an SMT problem. The operations of an ARX cipher are performed on n-bit vectors, whereas the formula of a SAT problem can only contain Boolean variables and the operations AND, NOT and OR. Therefore, an SMT problem in the bit-vector theory, which supports bit-vectors variables and the usual operations of bit-vectors, is used instead. Once the SMT problem is written, an SMT solver translates it into a SAT problem and solves it using a SAT solver. An example of an SMT solver is STP [75]. STP is built from a SAT solver and was also used in [118].

Table 4.1: The output RX-differences for rotation and XOR given two pairs (X, X') and (Y, Y') with RX-differences α and β, respectively, and a constant c.

Pair	RX-difference
$((X \ggg c), (X' \ggg c))$	$(\alpha \ggg c, \gamma)$
$((X \lll c), (X' \lll c))$	$(\alpha \lll c, \gamma)$
$((X \oplus Y), (X' \oplus Y'))$	$(\alpha \oplus \beta, \gamma)$
$((X \oplus c), (X' \oplus c))$	$(\alpha \oplus c \oplus (c \lll \gamma), \gamma)$

The SMT problem is written as follows:

- For every pair of n-bit input words of the key schedule and the encryption, an n-bit vector is used to represent the RX-difference of the pair;
- Additional n-bit vectors are used to represent the RX-difference after the addition, XOR and rotation operations when required;
- For each of the XOR and bit rotation operations, Table 4.1 is used to propagate properly the RX-differences;
- For every modular addition operation, (4.1) and (4.2) are used to ensure that the RX-differences are propagated with non-zero probability and (4.3) is used to calculate the weight;
- Finally, two constraints are used to ensure that the weights denoting the transition probability for the key schedule (resp., encryption) is at most W_K (resp., W_E).

4.2.3 Search Strategies

Our program works in two phases:

Phase 1 - finding a good RX-characteristic over the data part. The program starts by searching for an RX-characteristic covering the data part of the cipher with probability not smaller than $2^{-n/2}$, and the key schedule part with probability at least 2^{-mn} for mn the length of the master-key (thus ensuring that

at least one weak-key exists). If a solution adhering to these constraints is found, the objective function for the data part is updated and an RX-characteristic with probability not smaller than $2^{-n/4}$ is sought.

If the program cannot find a solution with probability at least $2^{-n/2}$, the objective function for the data part is relaxed and the program searches for an RX-characteristic with probability at least $2^{-1.5n/2}$. This binary search (over the exponent for the data part) is repeated until no further improvements are possible.

Phase 2 - increasing the size of the weak-key class. After the best RX-characteristic (in terms of its probability) is found, the program sets to increase the size of the weak-key class. Suppose ζ_0 is the probability for the RX-characteristic found in Phase 1, the objective function in Phase 2 is set such that the program finds RX-characteristics with probability ζ_0 for the data part, and probability at least $2^{-mn/2}$ for the key schedule. In a binary search similar to that of Phase 1, the best RX-characteristic for the key schedule is improved, under the constraint that this key RX-characteristic can support an RX-characteristic for the data part with probability ζ_0.

When the program can no longer improve the probability for the key's RX-characteristic, it outputs both RX-characteristics. Using this algorithm it is guaranteed that the data RX-characteristic has optimal probability, and that the corresponding key RX-characteristic allows for a non-empty weak key class and has the best possible probability, conditioned on the existence of a data RX-characteristic with probability ζ_0.

Additional search strategies

Note that, for purposes of obtaining a large number of rounds, the above search strategy prefers RX-characteristics with high probability in the data part over large weak-key classes. A side-effect of this search strategy is that the size of the weak-key class may be smaller than the data complexity. Such distinguishers can be used in the known-key model, but not for key recovery. We therefore devised a second search strategy with the additional constraint that $\zeta_0 \cdot \zeta_1 < 2^{-2n}$ where ζ_0 is as before, ζ_1 is the probability that a uniformly chosen key is weak, and $2n$ is the block size. By ensuring that the data complexity is now smaller than the size of the weak-key class, the additional constraint guarantees that the resulting distinguisher can be used for a key recovery attack.

4.2.4 The ArxPy Tool

ArxPy is a tool for finding optimal RX-characteristics in ARX block ciphers. It takes a Python-like implementation of an ARX block cipher as input, encodes it as a SAT/SMT problem, and gives it as an input to the solver.

To the best of our knowledge all automated tools [103, 107, 157] searching for statistical characteristics are implemented specifically for a particular cipher or for a small set of such. In order to encode an arbitrary cipher, a significant effort is needed.

Given a Python implementation of an ARX block cipher, ArxPy is executed with a simple shell command. Therefore, the only effort to use ArxPy is implementing the ARX block cipher in Python, which is negligible due to the low development time and code complexity of the Python programming language. On top of that, ArxPy is open source and has a modular architecture. Therefore, it can be easily adapted for specific needs.

Structure of Python implementations of ARX block ciphers

ArxPy expects a certain structure in the Python implementation of an ARX block cipher. Such implementation will be called an *ARX implementation*. Any iterated ARX block cipher can be considered, as long as all its operations are performed on words of the same size.

A minimal ARX implementation contains at least one global variable, `wordsize`, and two functions, `key_schedule` and `encryption`.

- The global variable `wordsize` contains the word size (in bits) of the ARX block cipher.
- The function `key_schedule` implements the key expansion algorithm of the cipher. This function receives m arguments as input, representing the m words of the key, and has no return value; the round keys are stored in the list-like object `round_keys`.
- The function `encryption` describes the encryption algorithm of the cipher. The arguments of this function represent the words of the plaintext. The intermediate values of each round are stored in the list-like object `rounds`, except for the last one which is used as the return value. This function can access the list-like object `round_keys`.

In order to implement the encryption and the key schedule, the Python operators `+`, `>>`, `<<` and `^` are used to describe modular addition, right and left rotations,

Algorithm 4.1 An ARX implementation of SPECK32. The total length of code is 23 lines.

```
wordsize = 16
number_of_rounds = 22
alpha = 7
beta = 2

# round function
def f(x, y, k):
    x = ((x >> alpha) + y) ^ k
    y = (y << beta) ^ x
    return x, y

def key_schedule(l2, l1, l0, k0):
    l = [None for i in range(number_of_rounds + 3)]

    round_keys[0] = k0
    l[0:3] = [l0, l1, l2]

    for i in range(number_of_rounds - 1):
        l[i+3], round_keys[i+1] = f(l[i], round_keys[i], i)

def encryption(x0, y0):
    rounds[0] = f(x0, y0, round_keys[0])

    for i in range(1, number_of_rounds):
        x, y = rounds[i - 1]
        round_output = f(x, y, round_keys[i])

        if i < number_of_rounds - 1:
            rounds[i] = round_output
        else:
            return round_output
```

and XOR, respectively. Augmented assignments, such as `+=` or `^=`, are not allowed.

Apart from the variable `wordsize` and the functions `key_schedule` and

Algorithm 4.2 An example for the `test` function and a `fix_differences` function for SPECK32.

```
def test():
    key = (0x1918, 0x1110, 0x0908, 0x0100)
    plaintext = (0x6574, 0x694c)
    ciphertext = (0xa868, 0x42f2)

    key_schedule(*key)
    assert ciphertext == encryption(*plaintext)

def fix_differences():
    for i in range(number_of_rounds):
        round_keys.fix_difference(i, 0)
```

`encryption`, additional variables and functions can be defined to improve the readability and modularity of the implementation. Algorithm 4.1 contains an ARX implementation of SPECK32.

There are several considerations about `rounds` and `round_keys` to take into account:

- They are not created nor declared in the ARX implementation. They are created by the ARXPY parser; and
- they only support the operator `[]` to access their elements. Slices and negative indices are not supported; and
- they can store either single values or lists of values, but each position can only be assigned once. Furthermore, storing a list of values must be done in a single assignment and cannot be done element-wise.

Apart from the functions `encryption` and `key_schedule`, an ARX implementation may contain two more special functions: the function `test` and the function `fix_differences`.

Test vectors can be added to an ARX implementation by using the Python statement `assert` inside the function `test`. It is also possible to fix the RX-differences of the round keys and the outputs of each round by using

the method `fix_difference` of `round_keys` and `rounds` inside the function `fix_differences`. Algorithm 4.2 shows an example of these functions for SPECK32, where all RX-differences of the round keys are fixed to 0.

Running the program

ARXPY uses the Python library SymPy [114] and the SMT solver STP [75]. They, together with Python3, must be installed in order to execute ARXPY.

The shell command to run ARXPY is the following:

```
python3 arxpy.py <ARX_implementation> <output>
```

where `<ARX_implementation>` is the name of the file containing an ARX implementation and `<output>` is the name of the output file.

To find an optimal RX-characteristic, ARXPY searches for characteristics using the search strategy described above. During an execution of ARXPY, intermediate characteristics are written to the output file. When the execution is complete, the last characteristic in the output file is the optimal one.

Implementation

ARXPY has been implemented in three modules: the ARX block cipher parser, the SMT writer and the characteristic finder. This section briefly explains these three modules.

The parser module takes an ARX implementation and generates a sequence of symbolic expressions of the output values of each round and the round keys. This is done by modifying the source code of the ARX implementation dynamically and executing the functions `key_schedule` and `encryption` symbolically. The Abstract Syntax Tree (AST) of the ARX implementation is used to modify the source code dynamically, whereas SymPy, a Python library for symbolic mathematics, is used to generate and handle the symbolic expressions.

The writer module takes the symbolic expressions generated by the parser as an input and outputs an SMT problem. The SMT problem is described in the SMT-LIB v2 [18] language, an input format supported by many SMT solvers. This is done by extracting the sequence of ARX operations of the encryption algorithm and the key schedule and translating these operations into equations as described in Section 4.2.2. The sequence of operations is obtained from the

symbolic expressions by traversing them as trees and extracting their nodes with methods provided by SymPy.

The finder module implements the search strategy to find the optimal RX-characteristic. The binary search strategy described in Section 4.2.3 is used to minimize the weight of the characteristic.

4.3 Summary and Future Work

This chapter dealt with automated tools for cryptanalysis. We developed two tools to accommodate the search for linear trails and RX-characteristics in ARX constructions. The first tool searches for linear trails by reducing the search space from all masks to those likely to produce a highly biased trail. This is achieved by starting with an intermediate mask having even-length sequences of consecutive bits, and trying to extend it both backwards and forward until certain constraints are violated.

The second tool searches for RX-characteristics by encoding the problem into a set of logical constraints and solving them using a SAT/SMT solver. The tool we developed receives a simple Python-like description and deals with the rest.

The benefit from building automated tools is twofold: first, it complements the human effort in searching for a good property. This effort is menial, and since computers, unlike humans, excel in performing such tasks, it is reasonable to assume that it will expose properties previously missed by humans. Secondly, using such tools frees their operators to developing new ideas; a task in which they outperform their computers.

Future work

At the time of writing this, automated search for statistical and structural properties is an active research field. This promising trend is likely to continue even without recommendations from this Author. Yet, if such is still desired, it would be useful if future research would focus on building general purpose search tools rather than ones for specific algorithms, thus allowing researchers to outsource this work to computers with minimal effort.

Chapter 5

Application of Cryptanalysis

> *"A process cannot be understood by stopping it. Understanding must move with the flow of the process, must join it and flow with it." (Paul-Muad'dib Atreides)*
>
> - Frank Herbert, *Dune*

This chapter deals with the application of cryptanalytic methods to existing algorithms. In a way, this is the cryptanalyst's real job.

In Section 5.1 we present an attack on Pure omd (p-omd). The attack uses a flaw in the security proof of p-omd to create a state collision by using the same nonce multiple times. This work was co-authored with Bart Mennink and was first published in [14]. As a result, the authors of p-omd dropped the claim for nonce-misuse resistance. A more thorough version was subsequently published in [15].

In Section 5.2 we present multiple attacks for Gost2. The 3 main attacks of this work are a fixed-point attack, a reflection attack and an impossible reflection attack. Less notable attacks are related-key differential attacks. This work was co-authored with Achiya Bar-On and Orr Dunkelman and was published in [6].

In Section 5.3 we present several RX-distinguishers for the Speck family of block ciphers. The distinguishers were found using the automated tool described in Section 4.2. This work appeared in part in [10] and was co-authored with Glenn De Witte and Yunwen Liu. A more complete manuscript was co-authored with Glenn De Witte, Yunwen Liu, and Adrián Ranea and was published in [109]. This Author was a main contributor to this work.

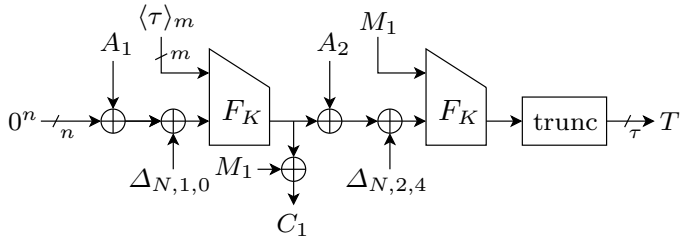


Figure 5.1: P-OMD for the specific case of $|A| = 2n$ and $|M| = m$.

5.1 A Nonce-misusing Attack on p-OMD

Recalling the specification of P-OMD given in Section 1.2.4 (*cf.* Figure 5.1) we consider a nonce-misusing adversary that operates as follows:

(i) Fix $N = \varepsilon$ and choose arbitrary $M \in \{0,1\}^m$ and $A_1, A_2, A_1' \in \{0,1\}^n$ such that $A_1 \neq A_1'$;

(ii) Query $\text{P-OMD}_K(N, A_1A_2, M) \to (C, T)$;

(iii) Query $\text{P-OMD}_K(N, A_1'A_2, M) \to (C', T')$;

(iv) Set $A_2' = C \oplus C' \oplus A_2$;

(v) Query forgery $\text{P-OMD}_K^{-1}(N, A_1'A_2', C', T)$.

For the first and second evaluations of P-OMD, the state difference right before the second F-evaluation is $C \oplus C'$. The forgery is formed simply by adding this value to A_2. Consequently, the first call to P-OMD and the forgery attempt have the exact same input to the second F-evaluation, and thus have the same tag. Therefore, the forgery attempt succeeds as

$$\text{P-OMD}_K^{-1}(N, A_1'A_2', C', T) = M$$

by construction.

The issue appears in the proof of [135] in Lemma 4 case 4, and more specifically the probability analysis of $\Pr[\mathsf{intcol} \mid \mathsf{E}_4]$. The authors claim that an adversary can indeed find an internal collision, but that any such collision happens with a

birthday-bound probability only. This reasoning, however, assumes that the input to every F-call is random, which is not the case given that the adversary can re-use the nonce and thus observe and modify the state using encryption queries.

5.2 Cryptanalysis of GOST2

In this section, we show how to adapt previous attacks against Gost to Gost2. This shows that Gost2 fails to deter the exact attacks it was supposed to avoid, thus casting doubt on its design methodology, and most notably on its key schedule.

We start by discussing some properties of Feistel structures. The first of which is the *reflection property* [94]. A reflection-point is a state $S = (L, R)$ such that $L = R$. We use the following lemma which is a well-known property of Feistel networks:

Lemma 12. *if S is a reflection-point, then for any subkey k, $R_k(S) = R_k^{-1}(S)$.*

Lemma 12 leads to the following corollary:

Corollary 3. *if S is a reflection-point, then for any sequence of subkeys K_i, $K_{i+1}, \ldots, K_{i+j}$, it holds that*

$$R_{K_i}(R_{K_{i+1}}(\ldots(R_{K_{i+j}}(S)))) = R_{K_i}^{-1}(R_{K_{i+1}}^{-1}(\ldots(R_{K_{i+j}}^{-1}(S)))).$$

We use Corollary 3 to present in Section 5.2.1 a reflection attack for a weak-key class of size 2^{224} with time complexity of 2^{192} using 2^{32} known-plaintexts. Then, in Section 5.2.2 we present a complementary attack (*i.e.*, an attack that works when the key is chosen amongst the remaining $2^{256} - 2^{224}$ "strong" keys) that reduces the time of exhaustive search by a factor of $2e$.

In Section 5.2.3 we present a fixed-point attack on the full Gost2 with time complexity 2^{237}. A fixed-point is an intermediate state S such that

$$S = R_{K_i}(R_{K_{i+1}}(\ldots(R_{K_{i+j}}(S))))$$

for some sequence of keys $K_i, \ldots, K_{i+j}$ used in rounds i to $i + j$. As we can see from Table 1.1, rounds 10–15 use the same keys as rounds 16–21. This leads to the following observation used in Section 5.2.3:

Observation 1. *If S_{10} is a fixed-point with respect to rounds 10–15 (*i.e. *$S_{10} = S_{16}$), then $S_{10} = S_{16} = S_{22}$.*

We conclude by briefly discussing several related-key differential attacks, and how they affect the security of GOST2.

5.2.1 A Reflection Attack for a Weak-key Class of GOST2

In this attack, we make use of a reflection-point in an intermediate state. As can be seen from Table 1.1, the order of keys in rounds 18–31 is:

$$K_{18}^7, K_{19}^0, K_{20}^1, K_{21}^2, K_{22}^3, K_{23}^4, K_{24}^6, K_{25}^5, K_{26}^4, K_{27}^3, K_{28}^2, K_{29}^1, K_{30}^0, K_{31}^7.$$

We assume that the key belongs to a weak-key class where $K_{24}^5 = K_{25}^6$. Then, if the intermediate state before round 25 is a reflection-point (*i.e.* $S_{25} = (x, x)$), we get, due to Corollary 3, that $C = S_{32} = S_{18}$, and thus the number of effective encryption rounds is 18 rather than 32. The probability of any state in GOST2 to be a reflection-point is 2^{-32}, which means that the probability that an adversary observing 2^{32} plaintexts would encounter at least one reflection-point in S_{25} is approximately 0.632.

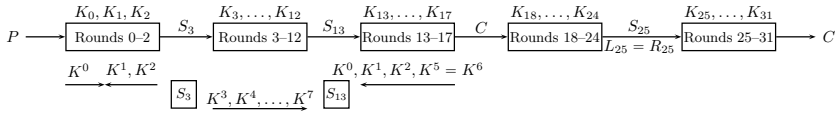


Figure 5.2: A schematic description of the reflection attack

The description of the attack is as follows (*cf.* Figure 5.2): The adversary observes 2^{32} pairs of plaintexts and ciphertexts denoted by (P_i, C_i). Then, they use an outer loop and two inner loops. In the outer loop they iterate over the values of S_3 and $K_{16}^5 = K_{17}^6$. Then, in the first inner loop, they iterate over the observed plaintext and ciphertext pairs (P_i, C_i), and over all possible values for K_0. They compute $S_1 = R_{K_0}(P_i)$ and use the outer loop's interim S_3 to retrieve K_1 and K_2 which can be computed through the 2-round Feistel network without additional complexity by solving the equations

$$K_1 = S^{-1}((L_1 \oplus L_3) \ggg 11) \boxminus R_1$$

and

$$K_2 = S^{-1}((R_1 \oplus R_3) \ggg 11) \boxminus L_3$$

with respect to known R_3 and L_3. They then obtain

$$S_{13} = R_{K_{13}}^{-1}(R_{K_{14}}^{-1}(R_{K_{15}}^{-1}(R_{K_{16}}^{-1}(R_{K_{17}}^{-1}(C_i = S_{18})))))$$

Algorithm 5.1 Pseudocode of the reflection attack against GOST2

Input: 2^{32} pairs of known-plaintexts and ciphertexts - $\{P_i, C_i\}$.

for $S_3, K_{16}^5 = K_{17}^6$ **do** {**Outer loop**}
  for $(P_i, C_i), K_0$ **do** {**Inner loop 1**}
    $K_1, K_2 \leftarrow \text{Solve}(P_i, S_3, K_0)$
    $S_{13} \leftarrow R_{K_{13}}^{-1}(R_{K_{14}}^{-1}(R_{K_{15}}^{-1}(R_{K_{16}}^{-1}(R_{K_{17}}^{-1}(C_i = S_{18})))))$
    $T[S_{13}] \leftarrow (P_i, K_0, K_1, K_2)$
  end for
  for K_3, K_4, K_7 **do** {**Inner loop 2**}
    $S_{13} \leftarrow R_{K_{12}}(R_{K_{11}}(R_{K_{10}}(R_{K_9}(R_{K_8}(R_{K_7}(R_{K_6}(R_{K_5}(R_{K_4}(R_{K_3}(S_3)))))))))))$
    $(P_i, K_0, K_1, K_2) \leftarrow T[S_{13}]$
    $\text{TRY}(K^0, K^1, K^2, K^3, K^4, K^5, K^6, K^7)$
  end for
end for

using the newly obtained $K_2 = K_{15}$ and $K_1 = K_{14}$ and the interim $K_0 = K_{13}, K_{16}^5 = K_{17}^6$. Finally, they store the tuple $(P_i, K_0^0, K_1^1, K_2^2)$ in $T[S_{13}]$.

In the second inner loop, which is run sequentially after the previous one, they iterate over all possible values of K_3^3, K_4^4, and K_7^7 for the encryption of S_3 through rounds 3–12 to obtain S_{13}. Then, they fetch the tuple (P_i, K_0, K_1, K_2) from $T[S_{13}]$. Since, on average, each S_{13} maps to a single tuple, the adversary now holds a candidate key $K = (K^0, K^1, K^2, K^3, K^4, K^5, K^6, K^7)$ which can be tested using a trial encryption. A pseudo-code describing the attack is presented in Algorithm 5.1.

The data complexity of the attack is 2^{32} known-plaintexts, and hence the expected number of reflection-points is 1. The outer loop iterates over 2^{64} possible values for S_3 and over 2^{32} possible values for $K_{16}^5 = K_{17}^6$, making its time complexity 2^{96}. The first inner loop iterates over the 2^{32} known pairs of plaintexts and ciphertexts, and over 2^{32} candidates for K_0, making its time complexity 2^{64}. The second inner loop iterates over 2^{32} candidates for each of the values of K_3, K_4, and K_7, *i.e.* it runs 2^{96} times. The total time complexity is therefore $2^{96} \cdot (2^{64} + 2^{96}) \approx 2^{192}$. The size of the table T is 2^{64} rows, each of 160 bits, resulting in memory complexity of $2^{68.32}$ bytes.[1] The memory

[1]The original publication included an error suggesting that the memory complexity is $2^{68.58}$ due to a need to save a 192-bit value in each row of the table. This error was corrected here.

complexity can be reduced to $2^{67.58}$ bytes by storing only P_i and K_0 in the first inner loop and recomputing K_1 and K_2 in the second inner one.

5.2.2 An Impossible Reflection Attack on the Full GOST2

In this section we present an impossible reflection attack on the full GOST2. This attack is a complementary attack to the one presented in Section 5.2.1 and uses the fact that $K^5 \neq K^6$. When this is the case and the S-boxes are bijective, the event $S_{18} = C$ is impossible, as it implies that $K^6_{24} = K^5_{25}$.

The attack uses an outer loop and two sequential inner loops. In the outer loop the adversary iterates over all K^5 and $K^6 \neq K^5$. In the first inner loop, they iterate over all possible S_3, and over all pairs (P_i, C_i) of plaintexts and ciphertexts, to find the values K_0, K_1, and K_2 leading from P_i to this S_3. Then, they assume towards contradiction that $S_{18} = C_i$ and decrypt $S_{18} = C_i$ back to S_{13}. They store in a temporary table $T1$ the values K^0_0, K^1_1, and K^2_2 indexed by $S_3||S_{13}$.

In the second inner loop the adversary iterates over S_3, K^3_3, K^4_4, and K^7_7, and tries to encrypt S_3 to S_{13}. If $S_3||S_{13}$ is in the table, the keys associated with their entry along with the values of the iterators (*i.e.* the current K_3, K_4, K_5, K_6, and K_7) are discarded as impossible keys. Finally, the adversary tries all remaining keys to find the right one. A full description of the attack can be found in Algorithm 5.2.

The probability that a key is impossible is e^{-1} and thus, the number of impossible keys is $e^{-1} \cdot 2^{256} \approx 2^{254.56}$ (the number of remaining keys is $(1 - e^{-1}) \cdot 2^{256} \approx 2^{255.34}$). We can reduce the number of possible keys by another factor of 2 by using GOST2's complementation property described in Section 5.2.4, leading to a time complexity of $2^{-1} \cdot (1 - e^{-1}) \cdot 2^{256} \approx 2^{254.34}$. The data complexity is 2^{64} known-plaintexts (2^{63} chosen-plaintexts when using the complementation property), and the memory complexity is $(1-e^{-1}) \cdot 2^{192}$ for storing the impossible keys.

5.2.3 A Fixed-point Attack on the Full GOST2

We now show how to mount a fixed-point attack against GOST2 using Observation 1. The probability that S_{10} is a fixed-point with respect to rounds 10–15 is 2^{-64}, and hence an adversary observing 2^{64} plaintexts (*i.e.* the entire codebook) expects to observe one such point. A second observation is that knowing the input and output to a 3-round Feistel network, an adversary can iterate over some of the bits in the first and last rounds and check if they match

Algorithm 5.2 Pseudocode of the impossible reflection attack against GOST2

Input: 2^{64} pairs of known-plaintexts and ciphertexts.
  {**Outer loop**}
  for K_5, K_6 **do**
    for $S_3, (P_i, C_i)$ **do** {**Inner loop 1**}
      for K_2 **do**
        $S_2 \leftarrow R_{K_2}^{-1}(S_3)$
        $T[L_2] \leftarrow K_2$
      end for
      for K_0 **do**
        $S_1 \leftarrow R_{K_0}(P_i)$
        $K_2 \leftarrow T[R_1]$
        $K_1 \leftarrow \text{SOLVE}(P_i, S_3, K_0, K_2)$
        $S_{18} \leftarrow C_i$
        $S_{13} \leftarrow R_{K_{13}}^{-1}(R_{K_{14}}^{-1}(R_{K_{15}}^{-1}(R_{K_{16}}^{-1}(R_{K_{17}}^{-1}(S_{18})))))$
        $T1[S_3||S_{13}] \leftarrow (K_0, K_1, K_2)$
      end for
    end for
    for S_3, K_3, K_4, K_7 **do** {**Inner loop 2**}
      $S_{13} \leftarrow R_{K_{12}}(R_{K_{11}}(R_{K_{10}}(R_{K_9}(R_{K_8}(R_{K_7}(R_{K_6}(R_{K_5}(R_{K_4}(R_{K_3}(S_3)))))))))))$
      $(K_0, K_1, K_2) \leftarrow T1[S_3||S_{13}]$
      Discard$(K_0, K_1, K_2, K_3, K_4, K_7)$
    end for
    for all undiscarded $(K_0, K_1, K_2, K_3, K_4, K_7)$ **do**
      TRY$(K^0, K^1, K^2, K^3, K^4, K^5, K^6, K^7)$ {(K_5, K_6) **are taken from the outer loop**}
    end for
  end for

in the middle round to filter out wrong keys. In this attack, we guess the 12 least significant bits of K_0 and K_2, which are denoted by $K_0[0–11]$ and $K_2[0–11]$, respectively, and match bits 11–22 in R_1. This gives $2^{12} \cdot 2^{12} \cdot 2^{-12} = 2^{12}$ suggestions for 24 bits of the key. By additionally guessing the carry bit in position 11, the adversary can compute key bits 12–19 in K_1 denoted by $K_1[12–19]$.

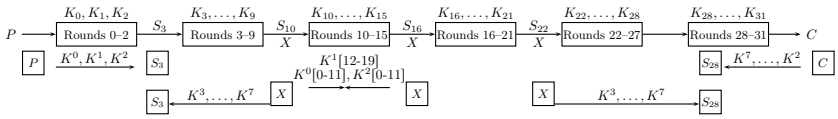


Figure 5.3: A schematic description of GOST2's fixed-point attack

The attack procedure is as follows (*cf.* Figure 5.3): In the outer loop, the adversary iterates over K_7, $K_0[0\text{–}11]$, $K_2[0\text{–}11]$, and $K_1[11]$. In the first inner loop they iterate over 2^{64} pairs of plaintexts and ciphertexts, and the remaining bits of K_0, K_1, and K_2. They decrypt C_i through rounds 31–28 to obtain S_{28} and encrypt P_i through rounds 0–2 to obtain S_3. They then use a table T to store in row $S_3||S_{28}$ the subkey values $K^0_{28}[12\text{–}31], K^1_{29}[0\text{–}10,12\text{–}31]$, and $K^2_{30}[12\text{–}31]$ supposedly leading from S_3 to S_{28}. Since T needs to store 2^{71} possible key values for each of the 2^{64} data pairs using 2^{128} rows, each row contains $2^{71} \cdot 2^{64} \cdot 2^{-128} = 2^7$ candidates on average where each 71-bit candidate occupies $2^{3.15}$ bytes.

In the second inner loop the adversary iterates over all possible S_{10} and all possible K_5 and K_6 to encrypt S_{10} through rounds 10–12 and obtain S_{13}. From S_{13} and $S_{16} = S_{10}$ they check if the values of $K_0[0\text{–}11]$ and $K_2[0\text{–}11]$ in the outer loop lead from S_{13} to S_{16}. For the values surviving this filter, they obtain $K_1[12\text{–}19]$. Then, by iterating over K_3 and K_4 they decrypt S_{10} through rounds 9–3 to obtain S_3 and encrypt $S_{22} = S_{10}$ through rounds 22–28 to obtain S_{28}. Using the interim S_3 and S_{28} they fetch 2^7 possible suggestions from T for $K^0_{28}[12\text{–}31], K^1_{29}[0\text{–}10,12\text{–}31], K^2_{30}[12\text{–}31]$, on average. They use $K_1[12\text{–}19]$ to further discard wrong values and try the remaining keys, *i.e.* a single key on average. A pseudocode of the attack procedure is presented in Algorithm 5.3.

The attack uses 2^{64} known-plaintexts and it builds a table of size $2^{128+7+3.15} = 2^{138.15}$ bytes. The time complexity is dominated by the total number of iterations. The number of iterations in the outer loop is 2^{32+25}. The time it takes to build the table in the first phase is 2^{64+71}. The second phase takes $2^{64+32+32}$ iterations for the outer part, which uses a filter to keep 2^{-12} of the keys. The inner part of the second phase, which is executed for the surviving keys requires another 2^{32+32} iterations. The probability that a row in the table matches the 2^{128} bits of S_3 and S_{28} that were obtained in this part is $(2^{-64})^2 = 2^{-128}$, *i.e.* each call to the table fetches 2^7 possible keys, which are filtered according to $K_1[12\text{–}19]$, leaving a single key to test. The total time complexity of the attack is then $2^{32+25} \cdot (2^{64+71} + 2^{64+64} \cdot 2^{-12} \cdot 2^{64}) \approx 2^{237}$.

Algorithm 5.3 Pseudocode of the fixed-point attack against GOST2

Input: 2^{64} pairs of known-plaintexts and ciphertexts.

for $K^7, K_0[0–11], K_2[0–11], K_1[11]$ **do** {**Outer loop**}
  for $(P_i, C_i), K_{28}^0[12–31], K_{29}^1[0–10,12–31], K_{30}^2[12–31]$ **do** {**First phase**}
    $S_{28} \leftarrow R_{K_{28}}^{-1}(R_{K_{29}}^{-1}(R_{K_{30}}^{-1}(R_{K_{31}}^{-1}(C_i))))$
    $S_3 \leftarrow R_{K_2}(R_{K_1}(R_{K_0}(P_i)))$
    $T[S_3||S_{28}] \leftarrow T[S_3||S_{28}] \cup (K_{28}^0[12–31], K_{29}^1[0–10,12–31], K_{30}^2[12–31])$
  end for
  for $S_{10} = S_{16} = S_{22}, K_5^5, K_6^6$ **do** {**Second phase - outer part**}
    $S_{13} \leftarrow R_{K_{12}}(R_{K_{11}}(R_{K_{10}}(S_{10})))$
    if FILTER$(S_{16}, S_{13}, K_0[0–11], K_2[0–11]) == TRUE$ **then**
      for K_3^3, K_4^4 **do** {**Second phase - inner part**}
        $(K_0[0–11], K_1[12–19], K_2[0–11]) \leftarrow$ SOLVE$(S_{16}, S_{13}, K_0[0–11], K_2[0–11], K_1[11])$
        $S_3 \leftarrow R_{K_3}^{-1}(R_{K_4}^{-1}(R_{K_5}^{-1}(R_{K_6}^{-1}(R_{K_7}^{-1}(R_{K_8}^{-1}(R_{K_9}^{-1}(S_{10})))))))$
        $S_{28} \leftarrow R_{K_{27}}(R_{K_{26}}(R_{K_{25}}(R_{K_{24}}(R_{K_{23}}(R_{K_{22}}(S_{22}))))))$
        for each $(K_0[12–31], K_1[0–10,12–31], K_2[12–31])$ **in** $T[S_3||S_{28}]$ **do**
          FILTER$(K^1$[12-19])
        end for
        TRY$(K^0, K^1, K^2, K^3, K^4, K^5, K^6, K^7)$
      end for
    end if
  end for
end for

5.2.4 Related-key Differential Properties in GOST2 and their Effect on its Security

An interesting omission by the authors of [69] is mitigation against related-key differential attacks. Indeed, in their introduction they discuss the works of [68, 83], but not any of the works of [57, 100, 146]. It seems that some attacks were not addressed in the design of GOST2, and that the cipher is still vulnerable to these attacks. We mention below three possible related-key differential attacks.

A 32-round related-key differential distinguisher with probability 1 (a complementation property)

Assume two related keys differing only in the most significant bit of each of the subkeys. A pair of plaintexts also differing only in the most significant bit of each of the two words will result in a pair of ciphertexts differing in the same bits with probability 1. This characteristic can be used to distinguish the full GOST2 from a random permutation in the related-key model with probability 1, using only 2 chosen-plaintexts. It can also be used to speed up exhaustive search on the full GOST2 by a factor of 2. Moreover, reduced-round variants of this observation can speed up all of our attacks by a factor of 2.

A 16-round related-key differential distinguisher with probability 1

Assume that a pair of plaintexts (P, P') whose intermediate states (S_8, S'_8) differ only in bit 31 (*i.e.* the most significant bit of the right half). Then, a pair of related keys having a difference in the most significant bit of the subkeys indexed by an odd number (*i.e.* K_1, K_3, K_5, and K_7), leads to the same difference before round 24 with probability 1. We can extend this differential characteristic 4 rounds backwards and 3 rounds forward with probability $(\frac{3}{4})^4$. This gives a truncated input difference with 8 known bits in S_4 leading to a truncated output difference with 28 known bits in S_{27}. By guessing K_0, K_1, K_2, K_3 and K_7 we can bridge the distance between the plaintext and S_4, and between the ciphertext and S_{27}. The attack can recover the full key using 2^{36} chosen-plaintexts, 2^{38} bytes of memory, 2^{226} time, and 2 related keys.

A 30-round related-key differential distinguisher with probability 2^{-30}

As was shown by Ko *et al.* in [100], a difference in the second most significant bit can be canceled by a related-key having a difference in the same position with probability $\frac{1}{2}$. Ko *et al.* also show how to concatenate this transition to obtain a related-key differential characteristic for 30 rounds with probability 2^{-30}. We observe that the same property holds also for GOST2, and that the key recovery attack of [100] can be adapted in a straightforward manner.

5.3 RX-cryptanalysis of Reduced-round Speck

Using the theory developed in Section 3.1 we developed the automated search tool described in Section 4.2. The tool was executed on round reduced versions

Table 5.1: Comparison of RX-characteristics with $\gamma = 1$ and previous differentials for different versions of SPECK. Entries marked with † were found through the adjusted search strategy. Entries marked with †† can only be used in the open-key model.

Version	Rounds	Data Prob.	Key Class Size	Ref.
32/64	9	2^{-30}	2^{64}	[67]
32/64	10	$2^{-19.15}$	$2^{28.10}$	Section 5.3.1
32/64	11††	$2^{-22.15}$	$2^{18.68}$	Section 5.3.1
32/64	12††	$2^{-25.57}$	$2^{4.92}$	Section 5.3.1
48/96	10	2^{-40}	2^{96}	[67]
48/96	11	2^{-45}	2^{96}	[74]
48/96	11	$2^{-23.15}$	$2^{14.93}$	Section 5.3.2
48/96	11†	$2^{-24.15}$	$2^{25.68}$	Section 5.3.2
48/96	12	$2^{-26.57}$	$2^{27.5}$	Section 5.3.2
48/96	12†	$2^{-26.57}$	$2^{43.51}$	Section 5.3.2
48/96	13††	$2^{-31.98}$	$2^{24.51}$	Section 5.3.2
48/96	14††	$2^{-37.40}$	$2^{0.34}$	Section 5.3.2
48/96	15††	$2^{-43.81}$	$2^{1.09}$	Section 5.3.2

of SPECK. We present an overview of the distinguishers found in Table 5.1 and compare them with the previously best ones.

5.3.1 RX-characteristics of Speck32/64

Table 5.2 shows two RX-characteristics covering 11 and 12 rounds found by our program. The best characteristic thus far covered 9 rounds of SPECK with probability 2^{-30}. Our 10-round characteristic has a much better probability of $2^{-19.15}$ for a weak-key class of size $2^{28.10}$. The table also shows that even our 12-round characteristic has probability $2^{-25.57}$, which is still higher than the previously known 9-round differential characteristic, although ours works for a weak-key class of about 30 keys. As was explained in Section 4.2, different tradeoffs between the weak-key class and the characteristic's probability are possible.

We extended the search to find 13-round characteristics and found that none exist, suggesting that a 12-round RX-characteristic is the longest possible for SPECK32/64.

Table 5.2: An 11-round (left) and a 12-round (right) RX-characteristics for SPECK32/64.

Round	Key RX-difference	Input RX-difference
0	0000	(0000||0000)
1	0000	(0000||0000)
2	0000	(0000||0000)
3	0001	(0000||0000)
4	0000	(0000||0000)
5	0003	(0000||0000)
6	0200	(0000||0000)
7	0205	(0200||0200)
8	0801	(0000||0800)
9	2001	(0000||2000)
10	AA0B	(0000||8000)
11		(2A0B||2A09)
Prob.	$2^{-45.32}$	$2^{-22.15}$

Round	Key RX-difference	Input RX-difference
0	0000	(0050||2000)
1	0100	(8000||0000)
2	0001	(0000||0000)
3	0000	(0000||0000)
4	0001	(0000||0000)
5	0000	(0000||0000)
6	0001	(0000||0000)
7	0200	(0000||0000)
8	0206	(0200||0200)
9	0800	(0000||0800)
10	2001	(0000||2000)
11	A40E	(0000||8000)
12		(240E||240C)
Prob.	$2^{-59.08}$	$2^{-25.57}$

5.3.2 RX-characteristics of Speck48/96

For SPECK48/96, we found RX-characteristics covering up to 15 rounds. Some of these characteristics are shown in Table 5.3 and Table 5.4. The distinguishers extend the previously best differential characteristic which covers 11 rounds with probability 2^{-45}. Note that the sizes of the weak key classes for the 14- and 15-round characteristics are marginal. However, due to resource constraints we ended the program before it completed its search. Hence, the characteristics presented here are not guaranteed to be optimal in length (*i.e.*, 16-round RX-characteristics may exist) nor in probability (*i.e.*, RX-characteristics with higher probabilities or a larger weak-key class may exist for the same number of rounds). In addition, the probabilities of the round function part in the 14- and 15-round characteristics are relatively high, which may imply that distinguishers with larger weak key classes can be found using a different trade-off.

5.4 Summary and Future Work

In this chapter we applied cryptanalytic techniques to existing constructions. The first contribution is a trivial forgery attack against P-OMD using only 3

Table 5.3: A 12-round (left) and a 13-round (right) RX-characteristics for Speck48/96.

Round	RX-difference in Key	RX-difference in Input
0	000008	(000000||000008)
1	000240	(000000||000040)
2	000000	(000200||000000)
3	000000	(000000||000000)
4	000000	(000000||000000)
5	000000	(000000||000000)
6	000001	(000000||000000)
7	000001	(000000||000000)
8	000001	(000000||000000)
9	010010	(000001||000001)
10	100089	(000010||000018)
11	8904de	(000080||000040)
12		(09049e||09069e)
Prob.	$2^{-52.49}$	$2^{-26.57}$

Round	RX-difference in Key	RX-difference in Input
0	000008	(000000||000008)
1	000240	(000000||000040)
2	000000	(000200||000000)
3	000000	(000000||000000)
4	000000	(000000||000000)
5	000000	(000000||000000)
6	000001	(000000||000000)
7	000001	(000000||000000)
8	000000	(000000||000000)
9	010018	(000001||000001)
10	1000f1	(000018||000010)
11	880801	(080080||080000)
12	c04911	(000000||400000)
13		(004911||004913)
Prob.	$2^{-71.49}$	$2^{-31.98}$

Table 5.4: A 14-round (left) and a 15-round (right) RX-characteristics for Speck48/96.

Round	RX-difference in Key	RX-difference in Input
0	000008	(000000||000008)
1	000240	(000000||000040)
2	000000	(000200||000000)
3	000000	(000000||000000)
4	000000	(000000||000000)
5	000000	(000000||000000)
6	000001	(000000||000000)
7	000001	(000000||000000)
8	000000	(000000||000000)
9	010018	(000000||000000)
10	1000e0	(010019||010019)
11	680021	(0801e8||000120)
12	000009	(000900||000000)
13	202844	(000000||000000)
14		(202844||202844)
Prob.	$2^{-95.66}$	$2^{-37.40}$

Round	RX-difference in Key	RX-difference in Input
0	000008	(000000||000008)
1	000240	(000000||000040)
2	000000	(000200||000000)
3	000000	(000000||000000)
4	000000	(000000||000000)
5	000000	(000000||000000)
6	000001	(000000||000000)
7	000001	(000000||000000)
8	000001	(000000||000000)
9	010011	(000001||000001)
10	100080	(000010||000018)
11	990391	(000089||000049)
12	480103	(000248||000000)
13	000301	(000100||000100)
14	91101d	(000000||000800)
15		(91181d||91581d)
Prob.	$2^{-94.91}$	$2^{-43.81}$

messages. Although not presented as such, this attack was found by searching for a state collision using the differential cryptanalysis framework. As a result of this attack, the P-OMD designers changed their security claims and no longer argue that P-OMD is nonce-misuse resistant.

The second contribution is a collection of attacks against GOST2. Inspired by attacks on the original GOST, we showed how 3 self-similarity attacks can be translated from GOST to GOST2, and that previously known related-key differential attacks for GOST are still valid, unmodified, for GOST2.

We also applied RX-cryptanalysis to the SPECK family of block ciphers and found, using one of the automated tools of the previous chapter, distinguishers longer than everything previously known, albeit in a weaker model.

Future work

It is striking to see how remarkably easy it is to find weaknesses in cryptosystems. The general idea behind the acceptance of cryptosystems, and especially symmetric-key primitives is that of Linus's law: "given enough eyeballs, all bugs are shallow". Seeing that not all "bugs" are shallow we must conclude that there are not enough eyeballs.

Future research must sift cryptosystems better and do so more systematically. While sometimes requiring ingenuity, the general problem of systematically reviewing existing cryptosystems against known and newly developed techniques is a classical task for entry level research. A first step in this direction is to compile an exhaustive list of techniques and their extensions. This list can then be translated to a set of standardized procedures a designer can use to "prove" to the world that their design is secure.[2] Then, after performing a survey identifing which algorithms are being used in the wild beyond the obvious AES and CHACHA+POLY, researchers, aided by automated tools, can evaluate existing algorithms systematically.

[2]Part of this work was already done in *ISO/IEC 15408—Common Criteria for Information Technology Security Evaluation* which includes a set of criteria for a system to be considered secure but not the evidence required from a designer to pronounce the system as such.

Chapter 6

Design of Symmetric-key Mechanisms

"Isn't it a pleasure to study and practice what you have learned?"

- Confucius, *Analects*

This chapter deals with the design of new cryptosystems. Whereas previous chapters were concerned with finding weaknesses and undesirable properties, this chapter seeks to learn from such weaknesses and apply these lessons to build better secured systems.

In Section 6.1 we present a security analysis of the TESLA variant used in the European Union's Galileo global navigation satellite system (GNSS) in order to derive proper security parameters against various adversaries. This work was submitted as a report to the EU commission [17] and was co-authored with Vincent Rijmen. Only those parts of the report where this Author was a main contributor were included.

In Section 6.2 we present our security model for Sections 6.3–6.5. For brevity sake, in addition to introducing some specific notation for these sections, we recall some notation already used.

In Sections 6.3–6.4 we present the Spoed and Spoednic modes of operation. Both modes are inspired by p-omd (*cf.* Section 1.2.4) and seek to fix the attack presented in Section 5.1 while preserving p-omd's clever trick to process the *Associated Data (AD)* almost for free. Spoed (*Simplified* p-omd *Encryption*

Decryption) aims to simplify the security proof of P-OMD against nonce-respecting adversaries. SPOEDNIC (*Simplified* P-OMD *Encryption Decryption (with) Nonce-misuse Integrity Conserved*) takes a further step forwards and tries to salvage the nonce-misuse resistance initially claimed for P-OMD. Both modes are proven to be secure, and consequently, the simplified proof allows to slightly improve the security bound. Both Sections 6.3–6.4 (together with Section 5.1 and Appendix A) were published in [15]. This Author was a main contributor to this work which was co-authored with Bart Mennink.

In Section 6.5 we show how to add RUP security to some existing modes of operation. This work was initially motivated by the *crypto-tagging attack* [154] on the Tor network. We sought to identify the causes allowing for the attack and mitigate them. We show that with minor modifications, some existing modes of operation can be made RUP-secure (incidentally, adding RUP security will also prevent the crypto-tagging attack). This work was co-authored with Orr Dunkelman and Atul Luykx and was published in [12]. Only those parts where this Author was a main contributor were included in this Thesis.

6.1 GALILEO Security Assessment

The GALILEO project is a *Global Navigation Satellite System (GNSS)* managed and deployed by the European Union. A common threat against GNSS systems are spoofing attacks. These are attacks where an adversary sends a false signal to the receiver. If the false signal is strong enough to overshadow the real signal, the receiver mistakes to accept it as real and provides the user with wrong location data. For a real-life example of such an attack believed to be performed by the Russian government see [45].

We evaluated the security of GALILEO's authentication service. This evaluation resulted in a set of recommended security parameters given in Section 6.1.8.

6.1.1 The TESLA-based GALILEO System

The *Timed Efficient Stream Loss-tolerant Authentication (*TESLA*)* scheme is an authentication scheme for continuous data broadcasting designed by Perrig *et al.* in [128]. The idea behind the scheme is to use a one-way function to create a chain of authentication keys. The last key is then released and authenticated off-channel. In each step, a package of data is authenticated using the previous key in the chain, followed by the key itself. Each receiver can verify the validity of the newly released key by first recomputing the one-way function, and then

checking if the result matches the last valid key. Once the key is accepted as valid, the data can be authenticated using this key.

A common method to obtain a one-way function is through a cryptographically secure hash function. A hash function is a function taking inputs of arbitrary length, producing an output of a fixed length n. If $F(\cdot)$ is a one-way function, then a preimage attack is an attack that, given Y, finds an X such that $Y = F(X)$ (note that X does not have to be the same X through which Y was generated). A cryptographically secure hash function has to satisfy, amongst other security requirements, that the work effort required for finding a preimage is 2^n. Standardized hash functions (*e.g.*, SHA-256 and SHA-3), although not *proven* to be secure, are the subject of a long ongoing research, and are believed to be so.

The TESLA variant used for the GALILEO system differs from the original scheme in several ways. The biggest difference between the schemes is that multiple satellites disseminate keys from the same chain in every step of GALILEO, rather than just one for the original TESLA. Another difference is that instead of using the MD5 hash function, GALILEO uses SHA-256 truncated to 80–128 bits, depending on the desired security level. Finally, the input used to generate the next key in the chain has a time dependent component, and each chain is associated with a fixed constant α used to thwart *Time-Memory Tradeoff* attacks.

In GALILEO, the level of truncation for the hash function's output determines the key length used for authentication of the MAC data. The purpose of this section is to analyse how different output sizes (*i.e.* key lengths) affect the security of the scheme. We will consider three scenarios: an online attack, an offline attack, and a hybrid attack between them. In the online attack, the adversary tries to find a preimage for the last released key. In the offline attack, the adversary has some time to prepare, hoping to observe one of their precomputed values at the right time. In the hybrid attack, the adversary can prepare, but only for a limited amount of time determined by the length of the chain.

6.1.2 Preliminaries

The preimage attack is a natural attack against the TESLA key chain. If, after a key is released, an adversary can find a preimage for that key, they can use this value to authenticate spoofed messages. It is important to note that a preimage can be found for any one-way function using exhaustive search, *i.e.* given an output value Y, try sufficiently enough inputs until the right output is obtained.

We start by introducing some notation. Let n be the key length. To generate a chain we first choose uniformly a *seed key* k_0 in the interval $[0, 2^n - 1]$. Further, let f be a one-way function used for generating a key in the chain. The next key is generated as $k_1 = f(k_0)$, and successive keys k_i are generated as $k_i = f(k_{i-1})$. The last key in the chain is k_{2^z}. When the chain is fully generated, $f(k_{2^z})$ is released and authenticated by external means. We refer to this key as the *initial KRoot*.

The time domain is split into sections of length T (*e.g.*, 10 seconds), which we will refer to as *MACK sections*. During the first MACK section each GALILEO satellite authenticates the navigation data it disseminates using k_{2^z}. At the end of this MACK section, k_{2^z} is released along with its $s - 1$ predecessors $k_{2^z-1}, \ldots, k_{2^z-s+1}$. Note that each of these keys allows to obtain k_{2^z}, and that none of them can be used to sign further messages. To verify that the navigation data was signed by the right key, k_{2^z} is hashed and compared against the initial KRoot. In the next section, messages are signed using k_{2^z-s}, which is released along with $k_{2^z-s-1}, \ldots, k_{2^z-2\cdot s+1}$ at the end of the MACK section. It is up to the receiver to decide if they wish to compare the hash of k_{2^z-s} to k_{2^z}, k_{2^z-s+1}, or to the initial KRoot. For simplicity, we will assume that the hash of a newly disseminated key $k_{2^z-\alpha\cdot s}$ is compared against $k_{2^z-\alpha\cdot s+1}$ in a single hashing step,[1] and refer to the key against which the new key is compared as the *effective key* and to any key that is not part of the chain in a certain MACK section as an *ineffective key*. Further, we refer to a key k_j, leading to an effective key after m hashing steps as a *preimage of degree* m or as the *effective key of MACK section* j.

6.1.3 Preimage Analysis

Given a key k_i, we can analyse its number of preimages as a binomial variable. The first observation is that an effective key k_i is part of a chain and must have at least one preimage *ipso facto*. The rest of the $2^n - 1$ values are left to be mapped at random by the hash function which, by definition, maps each of them to a fixed value with probability 2^{-n}.

For an arbitrary key k_i, we get that the number of preimages can be modeled as a binomial random variable X having

$$X \sim \mathcal{B}(2^n - 1, 2^{-n}) \tag{6.1}$$

[1] Since any already-released key can be obtained from $k_{2^z-\alpha\cdot s+1}$, this assumption can only help the adversary.

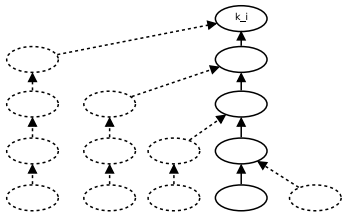


Figure 6.1: A tree of chains ending in k_i. Keys along the "real" chain are drawn with a solid line, while ineffective keys are drawn with a dotted line.

with expected value $\mu_X = (2^n - 1) \cdot 2^{-n} \approx 1$.[2] For an ineffective key, the expected number of preimages is μ_X. For an effective key, taking into account the previous key in the chain, the expected number of preimages is $\mu_X + 1$, *i.e.*, the previous key in the chain and possibly other ineffective keys.

By induction, assume that an effective key k_i has m preimages of degree $m-1$. Then, the number of preimages of degree m for k_i is $m+1$. Per the assumption that k_i is an effective key, one of the m preimages of degree $m-1$ is a key in the chain leading to k_i. This key has two preimages on average. Each of the other $m-1$ preimages of degree $m-1$ has a single preimage on average. We conclude that the number of preimages of degree m is $1 \cdot (m-1) + 2 \cdot 1 = m+1$. Using (6.1) as the base case of the induction, we conclude with the following proposition:

Proposition 2. *An effective key k_i has $\ell \cdot s + 1$ preimages of degree $\ell \cdot s$, on average.*

An example for this behavior is depicted in Figure 6.1.

6.1.4 The Security Model

An important point to discuss is the type of preimage required for a spoofing attack. Since the s keys that are released in every MACK section all contain the same information from a cryptographic point of view, finding a preimage within them cannot be used for an attack. In the sequel we denote the effective key by

[2]This may seem contradictory to [72] at first. The contradiction is resolved by noting that the TESLA variant we investigate in this Thesis appends a timestamp to each link of the key before inputting it to the one-way function. This ensures that a new one-way function is used for every input.

k_i. A forgery attack of length 1 requires a single hash (*i.e.* $k_i = f(k_j)$), a forgery attack of length 2 requires $s+1$ hashes (*i.e.* $k_i = f^{s+1}(k_j)$), and a forgery attack of length $\ell \cdot s$ requires $(\ell-1)\cdot s+1$ hashes (*i.e.* $k_i = f^{((\ell-1)\cdot s+1)}(k_j)$).

The adversary we consider tries to keep the victim in a spoofed state during ℓ MACK sections of length T. Given an already released effective key k_i an attack is a chain of length $(\ell-1)\cdot s+2$ such that k_i is the last key of the chain. We consider three scenarios: an online attack, an offline attack, and a hybrid attack and give a bound of the expected effort an adversary needs to spend to forge a chain of length $(\ell-1)\cdot s+2$.

6.1.5 The Online Attack

In the online attack, the adversary tries to spoof messages during ℓ MACK sections by targeting the latest key released. As we saw in the preimage analysis, there are $(\ell-1)\cdot s+2$ starting points converging after $(\ell-1)\cdot s+1$ steps to a specific effective key. In order to fully construct such a chain, $(\ell-1)\cdot s+1$ hashes are required. Using a random starting point $\bar{k}_{i-((\ell-1)\cdot s+1)}$, the probability that such a candidate chain ends in a specific value k_i (*i.e.* an effective key) is $p = \frac{(\ell-1)\cdot s+2}{2^n}$, when n is the key size. The expected number of starting points that need to be checked before finding a good one is inversely proportional to p, *i.e.* $\frac{1}{p} = \frac{2^n}{(\ell-1)\cdot s+2}$. Given $\hat{t}$ the number of hashes the adversary can execute during a MACK section (*i.e.*, the adversary's computational power, measured in units of $\frac{\#\text{ of hashes}}{\text{MACK section length}}$) of length T, it takes $\frac{(\ell-1)\cdot s+1}{\hat{t}}$ MACK sections to generate a single chain, or

$$\frac{2^n}{(\ell-1)\cdot s+2} \cdot \frac{(\ell-1)\cdot s+1}{\hat{t}} = \frac{2^n}{\hat{t}} \cdot \frac{(\ell-1)\cdot s+1}{(\ell-1)\cdot s+2} \tag{6.2}$$

MACK sections to test enough starting points and find a spoofed chain of length $(\ell-1)\cdot s+1$. Note that when the spoofed chains are very short (*i.e.* $\ell \cdot s = 1 \Rightarrow \ell = s = 1$), then $\frac{(\ell-1)\cdot s+1}{(\ell-1)\cdot s+2} = 0.5$ which reduces (6.2) to

$$\frac{2^{n-1}}{\hat{t}} \ . \tag{6.3}$$

6.1.6 The Hybrid Attack

The online attack targets an already released key and updates the target frequently as new effective keys are being released. The next two attacks target a yet to be released key. In the hybrid attack, the adversary waits until the value

α is released. As soon as it is published, the attacker starts generating chains of length $(\ell - 1) \cdot s + 2$ with endpoints $k_i^0, \ldots, k_i^{m-1}$ and stores them in a large database. Once the target key k_i is released, the adversary searches the database for a chain with this endpoint. Since an endpoint is univocally associated to a chain (through the α value) and a position (through the timestamp), the chains in the database are only valid with respect to a certain MACK section. If the target key k_i is not in the database at the end of MACK section i, the database is not reusable and the work effort spent on building it is lost.

To build the database, the adversary chooses a MACK section i, and a chain length $(\ell-1)\cdot s+2$ they wish to attack. They choose a starting point $\bar{k}^0_{i-((\ell-1)\cdot s+1)}$ and compute its endpoint $\bar{k}_i^0 = f^{(\ell-1)\cdot s+1}(k^0_{i-((\ell-1)\cdot s+1)})$. They store the pair $(\bar{k}_i^0, \bar{k}^0_{i-((\ell-1)\cdot s+1)})$ in a look-up table, and repeat this process for many starting points.

When the target key k_i is finally released, the adversary searches the table to see whether any value $\bar{k}_i^j = k_i$. If there is such a value, then its corresponding value in the table is the beginning of a spoofed chain. As before, since a chain of length $(\ell - 1) \cdot s + 2$ has $(\ell - 1) \cdot s + 2$ possible starting points converging to the same endpoint, the probability to randomly select a "good" starting point is $\frac{(\ell-1)\cdot s+2}{2^n}$. The analysis is similar to that of the online attack, for $\hat{t}$ the number of hashes per MACK section, it takes $\frac{(\ell-1)\cdot s+1}{\hat{t}}$ to generate a single chain, and the time it takes to try the required $\frac{2^n}{(\ell-1)\cdot s+2}$ starting points is $\frac{2^n}{(\ell-1)\cdot s+2} \cdot \frac{(\ell-1)\cdot s+1}{\hat{t}}$, measured in time units of length T.

This attack is a hybrid between the online attack described in the previous section and the offline attack described in the next one. Like the online attack, the attacker needs to know the α value of the chain being attacked. On the other hand, similar to the offline attack, the adversary builds a database of endpoints, and waits for the target key to be released at a future time.

6.1.7 The Offline Attack

We describe now how an adversary that does not know α can carry out a fully offline attack. We recall that α is a random value associated with each chain and is released close to the time when the chain enters into effect. An adversary who is interested to start preparing before α was released, must guess α in addition to guessing an appropriate starting point.

The probability of choosing the right α is $2^{-|\alpha|}$ and the probability to choose a good starting point is $\frac{(\ell-1)\cdot s+2}{2^n}$ as before. The combined probability to

choose both α and a good starting point is therefore $\frac{(\ell-1)\cdot s+2}{2^{n+|\alpha|}}$ as the two probabilities are independent. An adversary capable of computing $\hat{t}$ hashes per MACK section of length T needs $\frac{(\ell-1)\cdot s+1}{\hat{t}}$ MACK sections to generate a single chain, and the time it takes to try the required $\frac{2^{n+|\alpha|}}{(\ell-1)\cdot s+2}$ starting points is $\frac{2^{n+|\alpha|}}{(\ell-1)\cdot s+2} \cdot \frac{(\ell-1)\cdot s+1}{\hat{t}} = 2^{|\alpha|} \cdot \frac{2^n}{(\ell-1)\cdot s+2} \cdot \frac{(\ell-1)\cdot s+1}{\hat{t}}$, measured in time units of length T.

6.1.8 Conclusion and Recommendations for Security Parameters in GALILEO

This section derives bounds for the effort required by an adversary to attack the Galileo/TESLA scheme. We looked into three cases: an online attack, a hybrid attack, and an offline attack. We see that the online attack and the hybrid attack converge to the same bound. *I.e.*, an adversary who wants to keep a victim in a spoofed state during ℓ MACK sections, with a system using a key of n bits, where s keys are released in each time unit, and hardware capable of hashing $\hat{t}$ values during a MACK section of length T would need

$$\frac{2^n}{\hat{t}} \cdot \frac{(\ell-1)\cdot s+1}{(\ell-1)\cdot s+2} \tag{6.4}$$

MACK sections on average to find a spoofed chain using the online and hybrid attacks. For the offline attack (6.4) becomes

$$2^{|\alpha|} \cdot \frac{2^n}{\hat{t}} \cdot \frac{(\ell-1)\cdot s+1}{(\ell-1)\cdot s+2}. \tag{6.5}$$

As of Feb. 2017 the parameters considered for Galileo were $T = 10$ seconds, $s = 36$, $n = 82$. Assuming $\ell = 1$ and $\hat{t} = 2^{57}$ we plug these values into (6.4) and see that it would take

$$\frac{2^{82}}{2^{57}} \cdot \frac{(1-1)\cdot 36+1}{(1-1)\cdot 36+2} = 2^{24}$$

MACK sections to find a spoofed chain of length 1, which correspond to about 5.32 years. When $\ell = 2$, the time it takes to find a single forged chain almost doubles to ≈ 10.33 years. For other values, see Table 6.1. We note that the biggest step is when moving from $\ell = 1$ to $\ell = 2$, and that successive steps are increasingly smaller.

Table 6.1: A table describing the average length in years for spoofing a single chain for the adversarial effort considered in Section 6.1.8 (*i.e.*, $s = 36$, $n = 82$, and $\hat{t} = 2^{57}$).

ℓ	Years
1	5.32
2	10.33
3	10.48
4	10.53
5	10.56
6	10.57
7	10.59
8	10.59
9	10.60

As on Nov. 2016, the hardware cost for $2^{46.51}$ hashes per 10 seconds is about 1,600-4,550 US$ [33], putting the hardware cost alone between 2,000,000–6,500,000 US$ for $\ell = 1$ and double that number for larger ℓ. Further costs include energy consumption, cooling costs, and sufficient and efficient storage devices for the hybrid and offline attacks. This puts the attack way beyond the reach of most reasonable adversaries.

Note that extending the key by a single bit requires the adversary to double its efforts to maintain the same success rate. *I.e.*, once the system is switched to 83-bit keys, the same hardware will be able to attack the system within 10–20 years, instead of 5–10. Alternatively, the cost of attacking the system within 5–10 years is doubled to 4,000,000–13,000,000 US$.

It is important to point out that **the cryptoperiod of the chain (*i.e.*, its length) does not play a role in the success rate analysis,** as all it does is to slightly delay the adversary by limiting the length of the chains they can spoof, thus putting an upper bound on $(\ell - 1) \cdot s + 1$. For reasonable values of ℓ and s we get $\frac{(\ell-1)\cdot s+1}{(\ell-1)\cdot s+2} \approx 1$, which is already covered by (6.3). Therefore, once a certain level of confidence is decided, we recommend using (6.3) for choosing the key size.

The place where the cryptoperiod of the chain does play a role is in disaster mitigation. As can be seen in Table 6.1, when ℓ grows, the expected effort (in years) required for a successful attack is ≈ 10.6, independently of the length of ℓ. Setting the cryptoperiod of the chain to a certain length puts an upper

bound on ℓ. In other words, Table 6.1 shows that it takes about the same effort for the adversary to spoof a day's worth of chain as it is to spoof a year's worth. Replacing the chain periodically ensures that even if a spoof occurred, the attack is valid at most until the chain is replaced.

6.2 Security Model for Sections 6.3–6.5

Before we move to Sections 6.3–6.5 we recall some required notation and present new ones. Throughout the rest of this chapter, $n \geq 1$ denotes the state size. By $\oplus$ we denote the exclusive-or (XOR) operation, and by $\otimes$ or $\cdot$ finite field multiplication over 2^n. Concatenation is denoted using $\|$. Denote by $\{0,1\}^*$ the set of binary strings of arbitrary length and by $\{0,1\}^n$ the set of strings of size n. Denote by $(\{0,1\}^n)^+$ the set of strings of length a *positive* multiple of n. For an arbitrary string X, $|X|$ denotes its length, and $\langle |X| \rangle_n$ denotes its encoding in $n \geq 1$ bits. By $\text{left}_n(X)$ (resp. $\text{right}_n(X)$) we denote its n leftmost (resp. rightmost) bits. We use little-endian notation, which means the three notations "bit position 0", "the rightmost bit", and "the least significant bit" all refer to the same bit.

6.2.1 Authenticated Encryption

Let $\Pi = (\mathsf{E}, \mathsf{D})$ be an authenticated encryption scheme with associated data (AEAD), where

$$\mathsf{E} : (K, N, A, M) \mapsto (C, T) \text{ and}$$

$$\mathsf{D} : (K, N, A, C, T) \mapsto M/\perp$$

are the encryption and decryption functions of Π. Let \$ be a random function that returns $(C,T) \xleftarrow{\$} \{0,1\}^{|M|} \times \{0,1\}^\tau$ on every new tuple (N, A, M). In other words, E and \$ have the same interface, but the latter outputs a uniformly drawn ciphertext and tag for every new input.

An adversary $\mathbf{A}$ is a probabilistic algorithm that has access to one or more oracles $\mathcal{O}$, denoted $\mathbf{A}^{\mathcal{O}}$. By $\mathbf{A}^{\mathcal{O}} = 1$ we denote the event that $\mathbf{A}$, after interacting with $\mathcal{O}$, outputs 1. In below games, the adversaries have oracle access to E_K or its counterpart \$, and possibly D_K. The key K is uniformly drawn from $\{0,1\}^k$ at the beginning of the security experiment. We say that $\mathbf{A}$ is *nonce-respecting* (nr) if it never queries its encryption oracle under the same nonce twice, and *nonce-misusing* (nm) if it is allowed to make multiple encryption queries with the same nonce. The security definitions below follow [5, 23–25, 73, 84, 90, 91].

We define the advantage of $\mathbf{A}$ in breaking the confidentiality of Π as follows:

$$\mathbf{Adv}^{\mathrm{conf}}_{\Pi}(\mathbf{A}) = \left|\mathbf{Pr}\left(K \xleftarrow{\$} \{0,1\}^k,\ \mathbf{A}^{\mathsf{E}_K} = 1\right) - \mathbf{Pr}\left(\mathbf{A}^{\$} = 1\right)\right| .$$

For $\mathsf{n} \in \{\mathsf{nr}, \mathsf{nm}\}$, we denote by $\mathbf{Adv}^{\mathrm{conf}}_{\Pi}(\mathsf{n}, q, \ell, \sigma, t)$ the maximum advantage over all n-adversaries that make at most q queries, each of length at most ℓ message blocks and together of length at most σ message blocks, and that run in time t.

For integrity, we consider an adversary that tries to forge a ciphertext, which means that D_K returns a valid message (other than $\perp$) on input (N, A, C, T) and no previous encryption query $\mathsf{E}_K(N, A, M) = \mathsf{E}^N_K(A, M)$ returned (C, T) for any M. Formally:

$$\mathbf{Adv}^{\mathrm{int}}_{\Pi}(\mathbf{A}) = \mathbf{Pr}\left(K \xleftarrow{\$} \{0,1\}^k,\ \mathbf{A}^{\mathsf{E}_K, \mathsf{D}_K} \text{ forges}\right) .$$

For $\mathsf{n} \in \{\mathsf{nr}, \mathsf{nm}\}$, we denote by $\mathbf{Adv}^{\mathrm{int}}_{\Pi}(\mathsf{n}, q_{\mathsf{E}}, q_{\mathsf{D}}, \ell, \sigma, t)$ the maximum advantage over all n-adversaries that make at most q_{E} encryption and q_{D} decryption queries, each of length at most ℓ message blocks and together of length at most σ message blocks, and that run in time t. We remark that the nonce-respecting condition *only* applies to encryption queries: the adversary is always allowed to make decryption queries for "old" nonces, and to make an encryption query using a nonce which is already used in a decryption query before.

6.2.2 Separated AE Schemes

Following the RUP-model [4], we focus in Section 6.5 on designing *separated AE schemes*, where the decryption algorithm is split into plaintext computation and verification algorithms, to ensure that the decryption algorithm does not "hide" weaknesses behind the error symbol. Furthermore, our construction will communicate nonces in-band, meaning it will encrypt them and consider them as part of the ciphertext. As a result, the nonce no longer needs to be synchronized or communicated explicitly, as sufficient information is contained in the ciphertext. This changes the syntax slightly from that of Section 6.2.1, since now the decryption and verification algorithms no longer accept an explicit nonce input.

Formally, a separated AE scheme which communicates nonces in-band is a triplet of functions—encryption SEnc, decryption SDec, and verification SVer—where

$$\mathsf{SEnc} : (\mathsf{K}, \mathsf{N}, \mathsf{M}) \mapsto \mathsf{C}\,, \tag{6.6}$$

$$\mathsf{SDec} : (\mathsf{K}, \mathsf{C}) \mapsto \mathsf{M}\,, \tag{6.7}$$

$$\mathsf{SVer} : (\mathsf{K}, \mathsf{C}) \mapsto \{\bot, \top\}\ , \tag{6.8}$$

with K the keys, N the nonces, M the messages, and C the ciphertexts. Recall that the symbols $\top$ and $\bot$ represent success and failure of verification, respectively, and we assume that neither are elements of M. It must be the case that for all $K \in \mathsf{K}$, $N \in \mathsf{N}$, and $M \in \mathsf{M}$,

$$\mathsf{SDec}_K(\mathsf{SEnc}_K^N(M)) = M \quad \text{and} \quad \mathsf{SVer}_K(\mathsf{SEnc}_K^N(M)) = \top\,. \tag{6.9}$$

From a separated AE scheme $(\mathsf{SEnc}, \mathsf{SDec}, \mathsf{SVer})$ one can reconstruct the following conventional AE scheme (E, D):

$$\mathsf{E}_K^N(M) := \mathsf{SEnc}_K^N(M) \tag{6.10}$$

$$\mathsf{D}_K(C) := \begin{cases} M = \mathsf{SDec}_K(C) & \text{if } \mathsf{SVer}_K(C) = \top \\ \bot & \text{otherwise}\ , \end{cases} \tag{6.11}$$

where we assume that the AE scheme communicates nonces in-band as well.

Separated AE schemes must provide both chosen-ciphertext confidentiality and authenticity. Both of these security aspects are captured in the *RUPAE* measure of Barwell, Page, and Stam [19]. We adopt a stronger version of their definition, by requiring from the decryption algorithm to look "random" as well. Let Π denote a separated AE scheme $(\mathsf{SEnc}, \mathsf{SDec}, \mathsf{SVer})$, then the *RUPAE*-advantage of adversary $\mathbf{A}$ against Π is

$$\mathsf{RUPAE}_\Pi(\mathbf{A}) := \underset{\mathbf{A}}{\Delta}(\mathsf{SEnc}_K, \mathsf{SDec}_K, \mathsf{SVer}_K\,;\, \$_{\mathsf{SEnc}}, \$_{\mathsf{SDec}}, \bot)\ , \tag{6.12}$$

where $\mathbf{A}$ is *nonce-respecting*, meaning the same nonce is never queried twice to SEnc, and

$$\underset{\mathbf{A}}{\Delta}(\alpha\,;\,\beta) = |\Pr[\mathbf{A}^\alpha = 1] - \Pr[\mathbf{A}^\beta = 1]|.$$

Nonces may be repeated with SDec and SVer. Furthermore, $\mathbf{A}$ cannot use the output of an O_1^N query (*i.e.*, a query to the first oracle in the tuple) as the input to an O_2^N or O_3^N query with the same nonce N, otherwise such queries result in trivial wins.

6.2.3 Encryption Schemes

An *encryption scheme* $(\mathsf{Enc}, \mathsf{Dec})$ is a separated AE scheme without SVer. The basic security requirement for encryption schemes is chosen-plaintext confidentiality, but this is not sufficient for our purpose. In particular, a mode like CBC [120] will not work, since during decryption a change in the nonce will only affect the first decrypted plaintext block. What we need are encryption schemes where during decryption a change in the nonce will result in the entire plaintext changing. Modes such as CTR [120], OFB [120], and the encryption of OCB [102, 138, 139] suffice. In particular, it is necessary that both encryption and decryption algorithms give uniform random output when distinct nonces are input across both encryption *and* decryption. For example, with CTR mode, decryption is the same as encryption, and if nonces are never repeated across both algorithms then its output will always look uniformly random.

We use Shrimpton and Terashima's [151] SRND measure for encryption schemes, which was introduced by Halevi and Rogaway [78]:

$$\mathsf{SRND}(\mathbf{A}) := \underset{\mathbf{A}}{\Delta}\left(\mathsf{Enc}_K, \mathsf{Dec}_K \,;\, \$_{\mathsf{Enc}}, \$_{\mathsf{Dec}}\right) , \tag{6.13}$$

where K is chosen uniformly at random from K, and $\mathbf{A}$ must use a different nonce for *every* query it makes, to both of its oracles.

6.2.4 Tweakable Block Ciphers

A *tweakable block cipher* [108] is a pair of functions (E, D), with

$$\mathsf{E} : (\mathsf{K}, \mathsf{T}, \mathsf{X}) \mapsto \mathsf{X} \tag{6.14}$$

$$\mathsf{D} : (\mathsf{K}, \mathsf{T}, \mathsf{X}) \mapsto \mathsf{X} , \tag{6.15}$$

where K is the key space, T the tweak space, and X the domain, where $\mathsf{X} = \{0,1\}^x$ is a set of strings of a particular length. For all $K \in \mathsf{K}$ and $T \in \mathsf{T}$ it must be the case that E_K^T is a permutation with D_K^T as inverse. We will need to measure the SPRP quality of the tweakable block cipher, which is defined as

$$\mathsf{SPRP}(\mathbf{A}) := \underset{\mathbf{A}}{\Delta}\left(\mathsf{E}_K, \mathsf{D}_K \,;\, \pi, \pi^{-1}\right) , \tag{6.16}$$

where K is chosen uniformly at random from K, and (π, π^{-1}) is a family of independent, uniformly distributed random permutations over X indexed by T.

Although Liskov, Rivest, and Wagner [108] introduced the concept of finite-tweak-length (FTL) block ciphers, for our construction we need tweakable block

ciphers that can process variable tweak lengths (VTL). Starting from an FTL block cipher, one can construct a VTL block cipher by compressing the tweak using a universal hash function, and using the resulting output as the tweak for the FTL block cipher, as explained by Coron *et al.* [55]. Minematsu and Iwata [117] introduce the XTX construction which extends the tweak length while minimizing security loss.

There are a few dedicated constructions of FTL block ciphers: the hash function SKEIN [71] contains an underlying tweakable block cipher, the CAESAR competition candidates Joltik [87] and Deoxys [86] also developed new tweakable block ciphers, and the TWEAKEY framework [85] tackles the problem of designing tweakable block ciphers in general. Besides dedicated constructions, there are also constructions of tweakable block ciphers using regular block ciphers; see for example Rogaway's XE and XEX constructions [138], Minematsu's beyond-birthday-bound construction [116], Landecker, Shrimpton, and Terashima's CLRW2 construction [104], and Mennink's beyond-birthday-bound constructions [113].

6.2.5 (Tweakable) Keyed Compression Functions

Let $F : \{0,1\}^k \times \{0,1\}^{n+m} \to \{0,1\}^n$ be a keyed compression function. Denote by $\mathrm{Func}(\{0,1\}^{n+m}, \{0,1\}^n)$ the set of all compression functions from $n+m$ to n bits. We define the PRF security of F as

$$\mathbf{Adv}^{\mathrm{prf}}_{F}(\mathbf{A}) = \left| \begin{array}{r} \mathbf{Pr}\left(K \xleftarrow{\$} \{0,1\}^k\,,\, \mathbf{A}^{F_K} = 1\right) - \\ \mathbf{Pr}\left(R \xleftarrow{\$} \mathrm{Func}(\{0,1\}^{n+m}, \{0,1\}^n)\,,\, \mathbf{A}^{R} = 1\right) \end{array} \right| .$$

We denote by $\mathbf{Adv}^{\mathrm{prf}}_{F}(q,t)$ the maximum advantage over all adversaries that make at most q queries and that run in time t.

A tweakable keyed compression function $\widetilde{F} : \{0,1\}^k \times \mathcal{T} \times \{0,1\}^{n+m} \to \{0,1\}^n$ takes as additional input a tweak $t \in \mathcal{T}$. Denote by $\widetilde{\mathrm{Func}}(\mathcal{T}, \{0,1\}^{n+m}, \{0,1\}^n)$ the set of all tweakable compression functions from $n+m$ to n bits, where the tweak inputs come from $\mathcal{T}$. Formally, a tweakable keyed compression function is equivalent to a keyed compression function with a larger input, but for our analysis it is more convenient to adopt a dedicated notation. We define the tweakable PRF ($\widetilde{\mathrm{PRF}}$) security of $\widetilde{F}$ as

$$\mathbf{Adv}^{\widetilde{\mathrm{prf}}}_{\widetilde{F}}(\mathbf{A}) = \left| \begin{array}{r} \mathbf{Pr}\left(K \xleftarrow{\$} \{0,1\}^k\,,\, \mathbf{A}^{\widetilde{F}_K} = 1\right) - \\ \mathbf{Pr}\left(\widetilde{R} \xleftarrow{\$} \widetilde{\mathrm{Func}}(\mathcal{T}, \{0,1\}^{n+m}, \{0,1\}^n)\,,\, \mathbf{A}^{\widetilde{R}} = 1\right) \end{array} \right| .$$

We denote by $\mathbf{Adv}^{\widetilde{\text{prf}}}_{F}(q,t)$ the maximum advantage over all adversaries that make at most q queries and that run in time t.

6.3 SPOED

Using the security model from the previous section we now introduce the authenticated encryption scheme SPOED with the motivation of generalizing and simplifying P-OMD. As a bonus, the simplification allows for a better bound and a significantly shorter proof, making the scheme less susceptible to mistakes hiding in one of the lemmata.

6.3.1 Syntax

Let $k, n, \tau \in \mathbb{N}$ such that $\tau \leq n$. Here and throughout, we assume SPOED to process blocks of $m = n$ bits. However, the results easily generalize to arbitrary (but fixed) block sizes. Let $F : \{0,1\}^k \times \{0,1\}^{2n} \to \{0,1\}^n$ be a keyed compression function. SPOED consists of an encryption function E and a decryption function D.

- The encryption function E takes as input a key $K \in \{0,1\}^k$, a nonce $N \in \{0,1\}^n$, an arbitrarily sized associated data $A \in \{0,1\}^*$, and an arbitrarily sized message $M \in \{0,1\}^*$. It returns a ciphertext $C \in \{0,1\}^{|M|}$ and a tag $T \in \{0,1\}^\tau$;
- The decryption function D takes as input a key $K \in \{0,1\}^k$, a nonce $N \in \{0,1\}^n$, an arbitrarily sized associated data $A \in \{0,1\}^*$, an arbitrarily sized ciphertext $C \in \{0,1\}^*$, and a tag $T \in \{0,1\}^\tau$. It returns either a message $M \in \{0,1\}^{|C|}$ such that M satisfies $\mathsf{E}(K,N,A,M) = (C,T)$ or a dedicated failure sign $\perp$.

The encryption and decryption functions are required to satisfy

$$\mathsf{D}(K,N,A,\mathsf{E}(K,N,A,M)) = M$$

for any K, N, A, M.

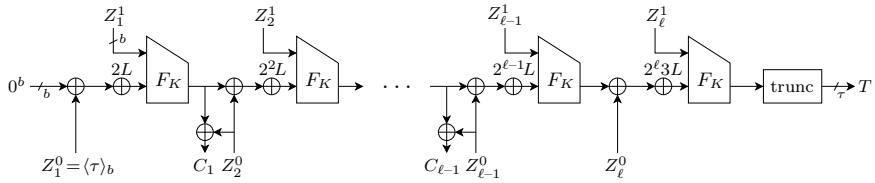


Figure 6.2: SPOED encryption, which outputs $C = \text{left}_{|M|}(C_1 \| \cdots \| C_{\ell-1})$ and T. Here, $L = F_K(N\|0)$.

6.3.2 Data Processing

SPOED is designed with the SHA-256 and SHA-512 compression functions in mind as an underlying primitive. SHA-256 is a compression function

$$\text{SHA-256} : \{0,1\}^{256} \times \{0,1\}^{512} \to \{0,1\}^{256}\,.$$

Similarly, SHA-512 is a compression function

$$\text{SHA-512} : \{0,1\}^{512} \times \{0,1\}^{1024} \to \{0,1\}^{512}.$$

In the sequel, we will define SPOED using SHA-256, or in other words use a keyed compression function

$$F_K(Z) = \text{SHA-256}(K, Z)\,,$$

where K is injected through the chaining value interface, and the block is injected through the message interface. Note that this implicitly means that we take $k = n = 256$. We nevertheless continue to use k and n for clarity. Also note that SPOED can be equivalently designed using the SHA-512 compression function, but a proper change to the sizes of the words should be introduced.

We now informally describe how to use SPOED, and refer the reader to Algorithms 6.1–6.2 for a formal specification. Define $L = F_K(N\|0)$. First, the associated data and message are padded into

$$(Z_1, \ldots, Z_\ell) = \text{GPAD}_{n,\tau}(A, M)\,,$$

where $\text{GPAD}_{n,\tau}$ is defined in Appendix A and each Z_i is a $(2n = 512)$-bit block consisting of two blocks $Z_i = Z_i^0 \,\|\, Z_i^1$ of size 256 bits each. SPOED reads all blocks but the last one sequentially and processes them by

$$t_i = F_K(t_{i-1} \oplus 2^i L \oplus Z_i^0 \,\|\, Z_i^1)\,,$$

where $t_0 = 0^n$. The ciphertext block C_i is generated as $C_i = t_i \oplus M_i$, truncated to the appropriate length if M_i does not contain an integral number of n message bits. The last block Z_ℓ contains the lengths of the message and the associated data and is processed through

$$t_\ell = F_K(t_{\ell-1} \oplus 2^\ell 3L \oplus Z_\ell^0 \parallel Z_\ell^1)\,.$$

The tag T is generated by removing the leftmost 256-τ bits of t_ℓ. SPOED is depicted in Figure 6.2.

Decryption goes fairly the same way: a t_i and a C_i value are used to recover M_i, and the state is set to C_i. This eventually leads to a verification tag T', and the message M is released if $T = T'$.

Algorithm 6.1 SPOED encryption E

Input: (K, N, A, M)
Output: (C, T)
1: $(Z_1, \ldots, Z_\ell) = \mathrm{GPAD}_{n,\tau}(A, M)$
2: $m = \lceil |M|/n \rceil$
3: $L = F_K(N\|0)$
4: $t_0 = 0^n$
5: **for** $i = 1, \ldots, \ell - 1$ **do**
6: $\quad t_i = F_K(t_{i-1} \oplus 2^i L \oplus Z_i^0 \parallel Z_i^1)$
7: $\quad C_i = t_i \oplus Z_{i+1}^0$
8: **end for**
9: $t_\ell = F_K(t_{\ell-1} \oplus 2^\ell 3L \oplus Z_\ell^0 \parallel Z_\ell^1)$
10: $C = \mathrm{left}_{|M|}(C_1\| \cdots \|C_{\ell-1})$
11: $T = \mathrm{left}_\tau(t_\ell)$
12: Return (C, T)

Algorithm 6.2 SPOED decryption D

Input: (K, N, A, C, T)
Output: M or $\perp$
1: $(Z_1, \ldots, Z_\ell) = \mathrm{GPAD}_{n,\tau}(A, C)$
2: $m = \lceil |C|/n \rceil\,,\ \rho = |C| \bmod n$
3: $L = F_K(N\|0)$
4: $t_0 = 0^n\,,\ M_0 = Z_1^0$
5: **for** $i = 1, \ldots, \ell - 1$ **do**
6: $\quad t_i = F_K(t_{i-1} \oplus 2^i L \oplus M_{i-1} \parallel Z_i^1)$
7: $\quad$ **if** $i < m$ **then** $M_i = t_i \oplus Z_{i+1}^0$
8: $\quad$ **if** $i = m$ **then** $M_i = \mathrm{left}_\rho(t_i) \oplus Z_{i+1}^0$
9: $\quad$ **if** $i > m$ **then** $M_i = Z_{i+1}^0$
10: **end for**
11: $t_\ell = F_K(t_{\ell-1} \oplus 2^\ell 3L \oplus Z_\ell^0 \parallel Z_\ell^1)$
12: $M = \mathrm{left}_{|C|}(M_1\| \cdots \|M_{\ell-1})$
13: $T' = \mathrm{left}_\tau(t_\ell)$
14: Return $T = T'\ ?\ M : \perp$

6.3.3 Security of SPOED

SPOED achieves confidentiality and integrity against nonce-respecting adversaries. Note that we do not claim security against nonce-misusing adversaries.

Theorem 6. *We have*

$$\mathbf{Adv}^{\mathrm{conf}}_{\mathrm{SPOED}}(\mathsf{nr}, q, \ell, \sigma, t) \le \frac{1.5\sigma^2}{2^n} + \mathbf{Adv}^{\mathrm{prf}}_{F}(2\sigma, t')\,,$$

$$\mathbf{Adv}^{\mathrm{int}}_{\mathrm{SPOED}}(\mathsf{nr}, q_{\mathsf{E}}, q_{\mathsf{D}}, \ell, \sigma, t) \le \frac{1.5\sigma^2}{2^n} + \frac{\ell q_{\mathsf{D}}}{2^n} + \frac{q_{\mathsf{D}}}{2^\tau} + \mathbf{Adv}^{\mathrm{prf}}_{F}(2\sigma, t')\,,$$

where $t' \approx t$.

These bounds, surprisingly, improve over the ones of P-OMD, but with a much shorter proof. The proof itself is given below. It relies on a preliminary result on a tweakable keyed compression function, which is in fact an abstraction of the XE tweakable block cipher [138] to compression functions, and has also been used for OMD [53] and P-OMD [134, 136], albeit with a worse bound.

Lemma 13. *Let* $F : \{0,1\}^k \times \{0,1\}^{2n} \to \{0,1\}^n$ *be a keyed compression function. Let* $\mathcal{T} = [1, 2^{n/2}] \times [0,1] \times \{0,1\}^n$*, and define* $\widetilde{F} : \{0,1\}^k \times \mathcal{T} \times \{0,1\}^{2n} \to \{0,1\}^n$ *as*

$$\widetilde{F}(K, (\alpha, \beta, N), S) = F(K, (2^\alpha 3^\beta \cdot F_K(N\|0) \parallel 0^n) \oplus S)\,. \tag{6.17}$$

Then, we have

$$\mathbf{Adv}^{\widetilde{\mathrm{prf}}}_{\widetilde{F}}(q, t) \le \frac{1.5q^2}{2^n} + \mathbf{Adv}^{\mathrm{prf}}_{F}(2q, t')\,,$$

where $t' \approx t$.

Proof. The proof is performed using the H-coefficient technique [50, 127]. It closely follows the proof of [76, Thm. 2]; the only significant differences appear in the fact that the underlying primitive is a one-way function instead of a permutation, and hence some bad events do not need to be considered. Specifically, in the terminology of [76, Thm. 2], the events $\mathrm{bad}_{1,2}$ and $\mathrm{bad}_{2,K}$ are inapplicable (as the adversary has no access to the underlying primitive), and for the events $\mathrm{bad}_{1,1}$, $\mathrm{bad}_{1,K}$, and $\mathrm{bad}_{K,K}$, we only have to consider input collisions to the primitives. Checking the corresponding bounds reveals a term $1.5q^2/2^n$.

As a first step, we replace the evaluations of F_K for $K \xleftarrow{\$} \{0,1\}^k$ by a random function $R : \{0,1\}^{2n} \to \{0,1\}^n$. As every evaluation of $\widetilde{F}$ renders at most 2 evaluations of F_K, this step induces a penalty of $\mathbf{Adv}^{\mathrm{prf}}_{F}(2q, t')$, where $t' \approx t$, and allows us to consider

$$\widetilde{F} : ((\alpha, \beta, N), S) \mapsto R((2^\alpha 3^\beta \cdot R(N\|0) \parallel 0^n) \oplus S)\,, \tag{6.18}$$

based on $R \xleftarrow{\$} \mathrm{Func}(\{0,1\}^{2n}, \{0,1\}^n)$. As we have replaced the underlying function F by a secret random primitive, we can focus on adversaries with unbounded computational power, and consider them to be information theoretic. Without loss of generality, any such adversary is deterministic. For the remainder of the analysis, consider any fixed deterministic adversary $\mathbf{A}$. Without loss of generality, we assume that $\mathbf{A}$ does not repeat any queries.

Let $R \xleftarrow{\$} \mathrm{Func}(\{0,1\}^{2n}, \{0,1\}^n)$ and $\widetilde{R} \xleftarrow{\$} \widetilde{\mathrm{Func}}(\mathcal{T}, \{0,1\}^{2n}, \{0,1\}^n)$ and consider any fixed deterministic adversary $\mathbf{A}$. In the real world, $\mathbf{A}$ has access to $\widetilde{F}$ of (6.18), while in the ideal world it has access to $\widetilde{R}$, and its goal is to distinguish both worlds. It makes q queries to the oracle, which are summarized in a view

$$\nu_F = \{(\alpha_1, \beta_1, N_1, S_1, T_1), \ldots, (\alpha_q, \beta_q, N_q, S_q, T_q)\}\,.$$

Note that, as $\mathbf{A}$ is deterministic, this view ν_F properly summarizes the interaction with the oracle. To suit the analysis, we will provide $\mathbf{A}$ with additional information *after* its interaction with its oracle. In more detail, it is given a *subkey transcript* ν_L that includes the computations of $R(N\|0)$ for all $N \in \{N_1, \ldots, N_q\}$. As the latter set may include duplicates, *i.e.*, it may be that $N_i = N_j$ for some $i \neq j$, the formalism of ν_L requires further notation $\{M_1, \ldots, M_r\}$ that is a minimal set that includes $N_1, \ldots, N_q$. Then, after the interaction of $\mathbf{A}$ with its oracle, we reveal

$$\nu_L = \{(M_1, L_1), \ldots, (M_r, L_r)\}\,,$$

In the real world the values $L_1, \ldots, L_r$ are defined as $L_i = R(M_i\|0)$, while in the ideal world, these values are randomly generated dummy subkeys $L_i \xleftarrow{\$} \{0,1\}^n$. Clearly, the disclosure of ν_L is without loss of generality as it only increases the adversary's chances. The complete view is defined as $\nu = (\nu_F, \nu_L)$. It is important to note that, as $\mathbf{A}$ never repeats queries, ν_F does not contain any duplicate elements and neither does ν_L, by minimality of the set $\{M_1, \ldots, M_r\}$.

H-Coefficient Technique. For brevity, denote $\mathbf{A}$'s distinguishing advantage by $\Delta_{\mathbf{A}}(\widetilde{F}; \widetilde{R})$. Denote by $X_{\widetilde{F}}$ the probability distribution of views when $\mathbf{A}$ is interacting with $\widetilde{F}$ and by $X_{\widetilde{R}}$ the probability distribution of views when $\mathbf{A}$ is interacting with $\widetilde{R}$. Let V be the set of all attainable views *i.e.*, those views that can be generated from $\widetilde{R}$ with non-zero probability. Let $\mathsf{V} = \mathcal{V}_{\mathrm{good}} \cup \mathcal{V}_{\mathrm{bad}}$ be a partition of the set of attainable views. The H-coefficient technique states the following. Let $0 \leq \varepsilon \leq 1$ be such that for all $\nu \in \mathcal{V}_{\mathrm{good}}$ we have

$$\frac{\Pr[X_{\widetilde{F}} = \nu]}{\Pr[X_{\widetilde{R}} = \nu]} \geq 1 - \varepsilon\,.$$

Then, the distinguishing advantage of $\mathbf{A}$ satisfies

$$\Delta_{\mathbf{A}}(\widetilde{F};\widetilde{R}) \leq \varepsilon + \Pr[X_{\widetilde{R}} \in \mathcal{V}_{\text{bad}}]\,. \tag{6.19}$$

We refer to [49] for a proof.

Bad Transcripts. Note that every tuple in ν_F uniquely fixes a subkey in ν_L and therewith uniquely fixes one evaluation $R(s) = t$. On the other hand, the evaluations in ν_L represent evaluations of R themselves. Informally, we will consider a transcript as *bad* if there exist two different tuples that have the same input to R. Formally, we say that a view ν is *bad* if it satisfies one of the following conditions:

Bad1. There exist $(\alpha, \beta, N, S, T) \in \nu_F$ and $(N, L), (M^*, L^*) \in \nu_L$ such that:

$$(2^\alpha 3^\beta \cdot L \parallel 0^n) \oplus S = M^* \parallel 0^n\,;$$

Bad2. There exist distinct $(\alpha, \beta, N, S, T), (\alpha^*, \beta^*, N^*, S^*, T^*) \in \nu_F$ and (not necessarily distinct) $(N, L), (N^*, L^*) \in \nu_L$ such that:

$$(2^\alpha 3^\beta \cdot L \parallel 0^n) \oplus S = (2^{\alpha^*} 3^{\beta^*} \cdot L^* \parallel 0^n) \oplus S^*\,.$$

Probability of Bad Transcripts. Consider a view ν in the ideal world $\widetilde{R}$. We will consider both bad events separately.

Bad1. Consider any query $(\alpha, \beta, N, S, T) \in \nu_F$ with corresponding subkey $(N, L) \in \nu_L$, and let $(M^*, L^*) \in \nu_L$ (q^2 choices in total). The queries render a bad view if

$$2^\alpha 3^\beta \cdot L = S^0 \oplus M^*\,.$$

As in the ideal world $L \xleftarrow{\$} \{0,1\}^n$, this equation is satisfied with probability 2^{-n}. Summing over all possible choices of queries, Bad1 is satisfied with probability at most $q^2/2^n$;

Bad2. Consider any distinct $(\alpha, \beta, N, S, T), (\alpha^*, \beta^*, N^*, S^*, T^*) \in \nu_F$ with corresponding $(N, L), (N^*, L^*) \in \nu_L$ ($\binom{q}{2}$ choices in total). The queries render a bad view if

$$2^\alpha 3^\beta \cdot L \oplus S^0 = 2^{\alpha^*} 3^{\beta^*} \cdot L^* \oplus S^{*0} \wedge S^1 = S^{*1}\,.$$

Clearly, if $N \neq N^*$, then $L \xleftarrow{\$} \{0,1\}^n$ is generated independently of the remaining values, and the first part of the condition holds with probability

$1/2^n$. Similar for the case where $N = N^*$ but $2^\alpha 3^\beta \neq 2^{\alpha^*} 3^{\beta^*}$. On the other hand, if $N = N^*$ and $2^\alpha 3^\beta = 2^{\alpha^*} 3^{\beta^*}$, we necessarily have $(N, \alpha, \beta) = (N^*, \alpha^*, \beta^*)$ (due to the non-colliding property of $2^\alpha 3^\beta$). As the two queries in ν_F are distinct, we have $S \neq S^*$, making above condition false. Concluding, Bad2 is satisfied with probability at most $\binom{q}{2}/2^n$.

We thus obtained that $\Pr[X_{\widetilde{R}} \in \mathcal{V}_{\text{bad}}] \leq 1.5q^2/2^n$.

Good Transcripts. Consider a good view ν. Denote by $\Omega_{\widetilde{F}}$ the set of all possible oracles in the real world and by $\text{comp}_{\widetilde{F}}(\nu) \subseteq \Omega_{\widetilde{F}}$ the set of oracles compatible with view ν. Define $\Omega_{\widetilde{R}}$ and $\text{comp}_{\widetilde{R}}(\nu)$ similarly. The probabilities $\Pr[X_{\widetilde{F}} = \nu]$ and $\Pr[X_{\widetilde{R}} = \nu]$ can be computed as follows:

$$\Pr[X_{\widetilde{F}} = \nu] = \frac{|\text{comp}_{\widetilde{F}}(\nu)|}{|\Omega_{\widetilde{F}}|} \text{ and } \Pr[X_{\widetilde{R}} = \nu] = \frac{|\text{comp}_{\widetilde{R}}(\nu)|}{|\Omega_{\widetilde{R}}|}\,.$$

Note that $|\Omega_{\widetilde{F}}| = (2^n)^{2^{2n}}$ and $|\Omega_{\widetilde{R}}| = (2^n)^{|\mathcal{T}|+2^{2n}} \cdot (2^n)^r$ (taking into account that in the ideal world ν contains r dummy subkeys). The computation of the number of compatible oracles is a bit more technical. Starting with $\text{comp}_{\widetilde{F}}(\nu)$, as ν is a good view, every tuple in ν represents *exactly one* evaluation of R, $q+r$ in total, and hence the number of functions R compatible with ν is $|\text{comp}_{\widetilde{F}}(\nu)| = (2^n)^{2^{2n}-(q+r)}$. Next, for $\text{comp}_{\widetilde{R}}(\nu)$, the tuples in ν_F all define *exactly one* evaluation of $\widetilde{R}$, q in total, and ν_L fixes all dummy keys. Therefore, the number of compatible oracles in the ideal world is $|\text{comp}_{\widetilde{R}}(\nu)| = (2^n)^{|\mathcal{T}|+2^{2n}-q}$. We consequently obtain

$$\frac{\Pr[X_{\widetilde{F}} = \nu]}{\Pr[X_{\widetilde{R}} = \nu]} = \frac{|\text{comp}_{\widetilde{F}}(\nu)| \cdot |\Omega_{\widetilde{R}}|}{|\Omega_{\widetilde{F}}| \cdot |\text{comp}_{\widetilde{R}}(\nu)|} = \frac{(2^n)^{2^{2n}-(q+r)} \cdot (2^n)^{|\mathcal{T}|+2^{2n}} \cdot (2^n)^r}{(2^n)^{2^{2n}} \cdot (2^n)^{|\mathcal{T}|+2^{2n}-q}} = 1\,,$$

putting $\varepsilon = 0$.

The proof is concluded via (6.19) and above computations. □

Note that P-OMD uses tweaks of the form 2^α, while we use $2^\alpha 3^\beta$. This is not a problem as long as the offsets are unique [138] (*i.e.*, there is no $(\alpha, \beta) \neq (\alpha', \beta')$ such that $2^\alpha 3^\beta = 2^{\alpha'} 3^{\beta'}$). For the case of $n = 128$, Rogaway [138] proved—via the computation of discrete logarithms—that the tweak domain $[1, 2^{n/2}] \times [0, 1]$ works properly, but this result is inadequate for our purposes as we use a compression function with $n \in \{256, 512\}$. Granger *et al.* [76] recently computed discrete logarithms for $n \leq 1024$, therewith confirming properness of the tweak set domain. Note that the tweak sets computed in [76, 138] commonly exclude the all-zero tweak $(\alpha, \beta) = (0, 0)$ because it is a representative of 1 and hence

problematic for XEX: see also [138, Sect. 6] and [115, Sect. 4]. Because F is a one-way function, its security analysis follows the one of XE, and this issue does not apply.

Also from an efficiency point of view, there is a difference between the masking of $\widetilde{F}$ in P-OMD and in SPOED. In more detail, P-OMD uses the Gray code masking (also used in OCB1 and OCB3) while for SPOED we have opted to describe it with powering-up (used in OCB2 and in various CAESAR candidates). Krovetz and Rogaway demonstrated that Gray codes are more efficient than powering-up [101], but on the downside they require more precomputation. Granger *et al.* [76] revisited the principle of masking of tweakable blockciphers, and presented a masking technique based on word-based linear feedback shift registers that improves over both Gray codes and powering-up in terms of efficiency and simplicity. The new masking technique can be implemented with SPOED with no sacrifice in security (and the result of Lemma 13 still applies).

Proof of Theorem 6

Let $K \in \{0,1\}^k$. Note that all evaluations of F_K are done in a tweakable manner, namely via (6.17). We replace these tweakable evaluations of F_K by a random tweakable compression function $\widetilde{R} \xleftarrow{\$} \widetilde{\mathrm{Func}}([1, 2^{n/2}] \times [0,1] \times \{0,1\}^n, \{0,1\}^{2n}, \{0,1\}^n)$. Note that for both confidentiality and integrity, the underlying $\widetilde{F}_K$ is invoked at most σ times. In other words, this step induces a penalty of (*cf.* Lemma 13)

$$\mathbf{Adv}^{\widetilde{\mathrm{prf}}}_{\widetilde{F}}(\sigma, t) \leq \frac{1.5\sigma^2}{2^n} + \mathbf{Adv}^{\mathrm{prf}}_{F}(2\sigma, t')\,,$$

where $t' \approx t$. This step leads to an idealized version of SPOED, called IDSPOED. IDSPOED is depicted in Figure 6.3. Concretely, we have obtained that

$$\mathbf{Adv}^{\mathrm{conf}}_{\mathrm{SPOED}}(\mathsf{nr}, q, \ell, \sigma, t) \leq \mathbf{Adv}^{\mathrm{conf}}_{\mathrm{IDSPOED}}(\mathsf{nr}, q, \ell, \sigma) + \frac{1.5\sigma^2}{2^n} + \mathbf{Adv}^{\mathrm{prf}}_{F}(2\sigma, t')\,,$$

$$\mathbf{Adv}^{\mathrm{int}}_{\mathrm{SPOED}}(\mathsf{nr}, q_{\mathsf{E}}, q_{\mathsf{D}}, \ell, \sigma, t) \leq \mathbf{Adv}^{\mathrm{int}}_{\mathrm{IDSPOED}}(\mathsf{nr}, q_{\mathsf{E}}, q_{\mathsf{D}}, \ell, \sigma) + \frac{1.5\sigma^2}{2^n} + \mathbf{Adv}^{\mathrm{prf}}_{F}(2\sigma, t')\,,$$

where t dropped out of the advantage function for IDSPOED because it has become irrelevant (formally, we proceed by considering an adversary that is unbounded in time). We prove in Lemma 14 that its confidentiality security

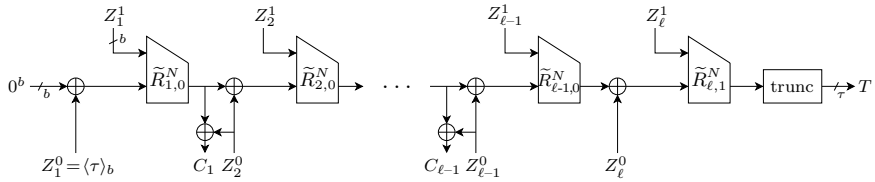


Figure 6.3: IDSPOED encryption, which outputs $C = \text{left}_{|M|}(C_1 \| \cdots \| C_{\ell-1})$ and T.

satisfies $\mathbf{Adv}^{\text{conf}}_{\text{IDSPOED}}(\mathsf{nr}, q, \ell, \sigma) = 0$, and in Lemma 15 that it provides integrity up to bound $\mathbf{Adv}^{\text{int}}_{\text{IDSPOED}}(\mathsf{nr}, q_\mathsf{E}, q_\mathsf{D}, \ell, \sigma) \leq \frac{\ell q_\mathsf{D}}{2^n} + \frac{q_\mathsf{D}}{2^\tau}$.

Lemma 14. *The advantage of any nonce-respecting adversary trying to break the confidentiality of* IDSPOED *is bounded as:*

$$\mathbf{Adv}^{\text{conf}}_{\text{IDSPOED}}(\mathsf{nr}, q, \ell, \sigma) = 0\,.$$

Proof. The functions $\widetilde{R}_{i,j}^N$ for $i = 1, \ldots, \ell - 1$, $j = 0, 1$, and $N \in \{0,1\}^n$ are independently and randomly distributed compression functions. As the adversary is assumed to be nonce-respecting, every nonce is used at most once. Every nonce is used in at most ℓ calls to $\widetilde{R}$, but these calls are by design all for different tweaks $(i, j) \in [1, 2^{n/2}] \times [0, 1]$. Therefore, all responses are randomly generated from $\{0,1\}^n$, and all ciphertext blocks and tag values are perfectly random. □

Lemma 15. *The advantage of any nonce-respecting adversary trying to break the integrity of* IDSPOED *is bounded as:*

$$\mathbf{Adv}^{\text{int}}_{\text{IDSPOED}}(\mathsf{nr}, q_\mathsf{E}, q_\mathsf{D}, \ell, \sigma) \leq \frac{\ell q_\mathsf{D}}{2^n} + \frac{q_\mathsf{D}}{2^\tau}\,.$$

Proof. Assume that $\mathbf{A}$ has made encryption queries (N^j, A^j, M^j) for $j = 1, \ldots, q_\mathsf{E}$, and denote the ciphertexts and tags by (C^j, T^j). Write $(Z_1^j, \ldots, Z_{\ell^j}^j) = \text{GPAD}_{n,\tau}(A^j, M^j)$ and denote the in- and outputs of the random functions by (s_i^j, t_i^j) for $i = 1, \ldots, \ell^j$.

Consider any forgery attempt (N, A, C, T), and denote its length by ℓ. Denote the message computed upon decryption by M. Refer to the state values as (s_i, t_i) for $i = 1, \ldots, \ell$, and write $(Z_1, \ldots, Z_\ell) = \text{GPAD}_{n,\tau}(A, M)$. The forgery is successful if $T = \text{left}_\tau(t_\ell)$.

Denote by col the event that there exists an encryption query j with $N^j = N$, $\ell^j = \ell$, and an index $i \in \{1, \ldots, \ell\}$, such that

$$t^j_{i-1} \oplus Z^{0j}_i \parallel Z^{1j}_i \neq t_{i-1} \oplus Z^0_i \parallel Z^1_i \ \wedge \ t^j_i = t_i \,.$$

Note that, as the adversary is nonce-respecting, there is at most one query j with $N^j = N$. We have, using shorthand notation $[i = \ell]$ for 0 if $i \neq \ell$ and 1 if $i = \ell$,

$$\Pr[\mathsf{col}] \leq \sum_{i=1}^{\ell} \Pr[s^j_i \neq s_i \wedge \widetilde{R}^N_{i,[i=\ell]}(s^j_i) = \widetilde{R}^N_{i,[i=\ell]}(s_i)] \leq \frac{\ell}{2^n} \,. \tag{6.20}$$

We make the following case distinction:

(i) $N \notin \{N^1, \ldots, N^{q_\mathsf{E}}\}$. This particularly means that $\widetilde{R}$ has never been queried for tweak $(\ell, 1, N)$, and thus that $\widetilde{R}^N_{\ell,1}$ responds with $t_\ell \xleftarrow{\$} \{0,1\}^n$. The forgery is successful with probability $2^{-\tau}$;

(ii) $N = N^j$ for some (unique) j. As the different evaluations of IdSpoed for different tweaks are independent, it suffices to focus on these two construction queries (the j-th encryption query and the forgery). We proceed with a further case distinction:

- $\ell \neq \ell^j$. This, again, means that $\widetilde{R}$ has never been queried for tweak $(\ell, 1, N)$. The forgery is successful with probability $2^{-\tau}$;
- $\ell = \ell^j$. We proceed with a further case distinction:
  - $s_\ell \neq s^j_{\ell^j}$. In this case, $\widetilde{R}$ has been queried before for tweak $(\ell, 1, N)$, but only once (as the adversary must be nonce-respecting) and never on input s_ℓ. Consequently, the response t_ℓ is uniformly drawn from $\{0,1\}^n$ and the forgery is successful with probability $2^{-\tau}$;
  - $s_\ell = s^j_{\ell^j}$. As the forgery must be different from the encryption queries, and as $\mathrm{GPAD}_{n,\tau}$ is an injective mapping, this case implies the existence of a non-trivial state collision. Hence, the forgery is successful with probability at most $\Pr[\mathsf{col}]$.

Concluding, the forgery is successful with probability at most $\Pr[\mathsf{col}] + 2^{-\tau}$, where $\Pr[\mathsf{col}]$ is bounded in (6.20). A summation over all q_D forgery attempts (*cf.* [22]) gives our final bound. □

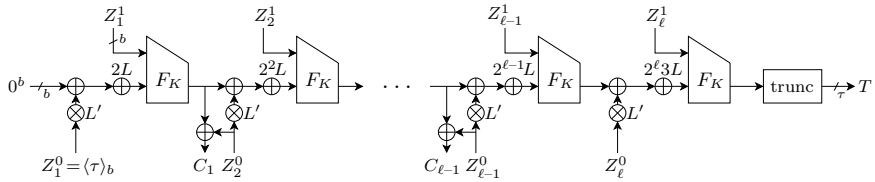


Figure 6.4: SPOEDNIC encryption, which outputs $C = \text{left}_{|M|}(C_1 \| \cdots \| C_{\ell-1})$ and T. Here, $L = F_K(N\|0)$ and $L' = F_K(N\|1)$.

6.4 SPOEDNIC

SPOED is simpler and more efficient than P-OMD, but it also falls victim to nonce-misuse attacks. In this section, we introduce SPOEDNIC, a strengthened version of SPOED that retains some level of security if the nonce is reused. As a matter of fact, SPOEDNIC differs from SPOED in (and only in) the fact that it uses an additional subkey $L' = F_K(N\|1)$ to blind the input values Z_i^0. More formally, SPOEDNIC inherits the syntax and generalized padding from SPOED (see Section 6.3.1 and Appendix A). The data processing is fairly similar to that of SPOED (Section 6.3.2); we only present the depiction in Figure 6.4 and the formal description in Algorithms 6.3–6.4. Both algorithms differ from Algorithms 6.1–6.2 *only in* lines 3 and 6. SPOEDNIC boils down to SPOED (Figure 6.2) when setting $L' = 1$ instead of $L' = F_K(N\|1)$.

Algorithm 6.3 SPOEDNIC encryption E

Input: (K, N, A, M)
Output: (C, T)

$(Z_1, \ldots, Z_\ell) = \mathrm{GPAD}_{n,\tau}(A, M)$
$m = \lceil |M|/n \rceil$
$L = F_K(N\|0)$, $L' = F_K(N\|1)$
$t_0 = 0^n$
for $i = 1, \ldots, \ell - 1$ **do**
  $t_i = F_K(t_{i-1} \oplus 2^i L \oplus (Z_i^0 \cdot L') \,\|\, Z_i^1)$
  $C_i = t_i \oplus Z_{i+1}^0$
end for
$t_\ell = F_K(t_{\ell-1} \oplus 2^\ell 3L \oplus Z_\ell^0 \,\|\, Z_\ell^1)$
$C = \mathrm{left}_{|M|}(C_1\| \cdots \|C_{\ell-1})$
$T = \mathrm{left}_\tau(t_\ell)$
Return (C, T)

Algorithm 6.4 SPOEDNIC decryption D

Input: (K, N, A, C, T)
Output: M or $\perp$

$(Z_1, \ldots, Z_\ell) = \mathrm{GPAD}_{n,\tau}(A, C)$
$m = \lceil |C|/n \rceil$, $\rho = |C| \bmod n$
$L = F_K(N\|0)$, $L' = F_K(N\|1)$
$t_0 = 0^n$, $M_0 = Z_1^0$
for $i = 1, \ldots, \ell - 1$ **do**
  $t_i = F_K(t_{i-1} \oplus 2^i L \oplus (M_{i-1} \cdot L') \,\|\, Z_i^1)$
  if $i < m$ **then** $M_i = t_i \oplus Z_{i+1}^0$
  if $i = m$ **then** $M_i = \mathrm{left}_\rho(t_i) \oplus Z_{i+1}^0$
  if $i > m$ **then** $M_i = Z_{i+1}^0$
end for
$t_\ell = F_K(t_{\ell-1} \oplus 2^\ell 3L \oplus Z_\ell^0 \,\|\, Z_\ell^1)$
$M = \mathrm{left}_{|C|}(M_1\| \cdots \|M_{\ell-1})$
$T' = \mathrm{left}_\tau(t_\ell)$
Return $T = T'$? $M : \perp$

6.4.1 Security of SPOEDNIC

We prove that SPOEDNIC achieves confidentiality against nonce-respecting adversaries and integrity against both nonce-respecting and nonce-misusing adversaries. Note that we do not claim confidentiality against nonce-misusing adversaries.

Theorem 7. *We have*

$$\mathbf{Adv}^{\mathrm{conf}}_{\mathrm{SPOEDNIC}}(\mathsf{nr}, q, \ell, \sigma, t) \le \frac{1.5\sigma^2}{2^n} + \mathbf{Adv}^{\mathrm{prf}}_F(3\sigma, t')\,,$$

$$\mathbf{Adv}^{\mathrm{int}}_{\mathrm{SPOEDNIC}}(\mathsf{nr}, q_{\mathsf{E}}, q_{\mathsf{D}}, \ell, \sigma, t) \le \frac{1.5\sigma^2}{2^n} + \frac{\ell q_{\mathsf{D}}}{2^n} + \frac{q_{\mathsf{D}}}{2^\tau} + \mathbf{Adv}^{\mathrm{prf}}_F(3\sigma, t')\,,$$

$$\mathbf{Adv}^{\mathrm{int}}_{\mathrm{SPOEDNIC}}(\mathsf{nm}, q_{\mathsf{E}}, q_{\mathsf{D}}, \ell, \sigma, t) \le \frac{1.5\sigma^2}{2^n} + \frac{\ell q_{\mathsf{E}}^2/2}{2^n} + \frac{\ell q_{\mathsf{E}} q_{\mathsf{D}}}{2^n} + \frac{q_{\mathsf{D}}}{2^\tau} + \mathbf{Adv}^{\mathrm{prf}}_F(3\sigma, t')\,,$$

where $t' \approx t$.

The proof of Theorem 7 is given below. It again relies on a preliminary result on a tweakable keyed compression function, which we now prove.

We will use a slightly more complex version of the tweakable keyed compression function used to prove the security of SPOED, where the masking using $Z_i^0 \cdot L'$ is included within the function. The proof is a fairly straightforward extension of the one for Lemma 13.

Lemma 16. *Let $F : \{0,1\}^k \times \{0,1\}^{2n} \to \{0,1\}^n$ be a keyed compression function. Let $\mathcal{T} = [1, 2^{n/2}] \times [0,1] \times \{0,1\}^n \times \{0,1\}^n$, and define $\widetilde{F} : \{0,1\}^k \times \mathcal{T} \times \{0,1\}^{2n} \to \{0,1\}^n$ as*

$$\widetilde{F}(K, (\alpha, \beta, A, N), S) = F(K, (2^\alpha 3^\beta \cdot F_K(N\|0) \oplus A \cdot F_K(N\|1) \parallel 0^n) \oplus S)\,. \tag{6.21}$$

Then, we have

$$\mathbf{Adv}^{\widetilde{\text{prf}}}_{\widetilde{F}}(q,t) \leq \frac{1.5q^2}{2^n} + \mathbf{Adv}^{\text{prf}}_F(3q, t')\,,$$

where $t' \approx t$.

Proof. The proof is a slight extension of the one for Lemma 13, where now we have twice as many subkeys. We only sketch the major differences.

Again, the first step is the replacement of F_K for $K \xleftarrow{\$} \{0,1\}^k$ by a random function $R : \{0,1\}^{2n} \to \{0,1\}^n$. As every query to $\widetilde{F}$ renders at most 3 evaluations of F, this step induces a penalty of $\mathbf{Adv}^{\text{prf}}_F(3q, t')$, where $t' \approx t$, and allows to consider

$$\widetilde{F} : ((\alpha, \beta, A, N), S) \mapsto R((2^\alpha 3^\beta \cdot R(N\|0) \oplus A \cdot R(N\|1) \parallel 0^n) \oplus S)\,, \tag{6.22}$$

based on $R \xleftarrow{\$} \text{Func}(\{0,1\}^{2n}, \{0,1\}^n)$. Switching to an information theoretic, deterministic, adversary $\mathbf{A}$ is done in the same way as in Section 6.3.3.

Let $R \xleftarrow{\$} \text{Func}(\{0,1\}^{2n}, \{0,1\}^n)$ and $\widetilde{R} \xleftarrow{\$} \widetilde{\text{Func}}(\mathcal{T}, \{0,1\}^{2n}, \{0,1\}^n)$. Consider any fixed deterministic adversary $\mathbf{A}$. In the real world, it has access to $\widetilde{F}$ of (6.22), while in the ideal world it has access to $\widetilde{R}$, and its goal is to distinguish both worlds. It makes q queries to the oracle, which are summarized in a view

$$\nu_F = \{(\alpha_1, \beta_1, A_1, N_1, S_1, T_1), \ldots, (\alpha_q, \beta_q, A_q, N_q, S_q, T_q)\}\,.$$

As an extension to the proof of Lemma 13, we now reveal to the adversary two subkey transcripts ν_L and $\nu_{L'}$, the former captures the evaluations $R(N\|0)$ and the latter the evaluations $R(N\|1)$ for all $N \in \{N_1, \ldots, N_q\}$. More formally, let $\{M_1, \ldots, M_r\}$ be a minimal set that includes $N_1, \ldots, N_q$. Then, after the interaction of $\mathbf{A}$ with its oracle, we reveal

$$\nu_L = \{(M_1, L_1), \ldots, (M_r, L_r)\}\,,$$

$$\nu_{L'} = \{(M_1, L'_1), \ldots, (M_r, L'_r)\}\,.$$

In the real world, the subkeys are generated as $L_i = R(M_i\|0)$ and $L'_i = R(M_i\|1)$, while in the ideal world they are randomly generated dummy subkeys $L_i, L'_i \xleftarrow{\$} \{0,1\}^n$. The complete view is defined as $\nu = (\nu_F, \nu_L, \nu_{L'})$.

Bad Transcripts. Formally, we say that a view ν is *bad* if it satisfies one of the following conditions:

Bad1. There exist $(\alpha, \beta, A, N, S, T) \in \nu_F$, $(N, L) \in \nu_L$, $(N, L') \in \nu_{L'}$, and $(M^*, L^*) \in \nu_L \cup \nu_{L'}$ such that:

$$(2^\alpha 3^\beta \cdot L \oplus A \cdot L' \parallel 0^n) \oplus S = M^* \parallel 0^n \vee M^* \parallel 1^n\,;$$

Bad2. There exist distinct $(\alpha, \beta, A, N, S, T), (\alpha^*, \beta^*, A^*, N^*, S^*, T^*) \in \nu_F$, $(N, L), (N^*, L^*) \in \nu_L$, and $(N, L'), (N^*, L'^*) \in \nu_{L'}$ such that:

$$(2^\alpha 3^\beta \cdot L \oplus A \cdot L' \parallel 0^n) \oplus S = (2^{\alpha^*} 3^{\beta^*} \cdot L^* \oplus A^* \cdot L'^* \parallel 0^n) \oplus S^*\,.$$

Probability of Bad Transcripts. Consider a view ν in the ideal world $\widetilde{R}$. We will consider both bad events separately.

Bad1. Consider any query $(\alpha, \beta, A, N, S, T) \in \nu_F$ with corresponding subkeys $(N, L) \in \nu_L$ and $(N, L') \in \nu_{L'}$, and let $(M^*, L^*) \in \nu_L \cup \nu_{L'}$. Note that we have q choices for the query from ν_F, and q possible values M^* (even though $\nu_L \cup \nu_{L'}$ may contain up to $2q$ tuples). The queries render a bad view if

$$2^\alpha 3^\beta \cdot L \oplus A \cdot L' = S^0 \oplus M^*\,.$$

As in the ideal world $L \xleftarrow{\$} \{0,1\}^n$, this equation is satisfied with probability $1/2^n$. Summing over all possible choices of queries, Bad1 is satisfied with probability at most $q^2/2^n$;

Bad2. Consider any distinct $(\alpha, \beta, A, N, S, T), (\alpha^*, \beta^*, A^*, N^*, S^*, T^*) \in \nu_F$, $(N, L), (N^*, L^*) \in \nu_L$, and $(N, L'), (N^*, L'^*) \in \nu_{L'}$ ($\binom{q}{2}$ choices in total). The queries render a bad view if

$$2^\alpha 3^\beta \cdot L \oplus A \cdot L' \oplus S^0 = 2^{\alpha^*} 3^{\beta^*} \cdot L^* \oplus A \cdot L'^* \oplus S^{*0} \wedge S^1 = S^{*1}\,.$$

The case $N \neq N^*$ and the case $N = N^*$ but $2^\alpha 3^\beta \neq 2^{\alpha^*} 3^{\beta^*}$ are as in Lemma 13. Similarly, if $N = N^*$ but $A \neq A^*$, we can rely on the randomness of L' to argue that the condition is satisfied with probability $1/2^n$. On the other hand, if $(\alpha, \beta, A, N) = (\alpha^*, \beta^*, A^*, N^*)$, this necessarily implies that $S \neq S^*$, making above condition false. Concluding, Bad2 is satisfied with probability at most $\binom{q}{2}/2^n$.

We thus obtained that $\Pr[X_{\widetilde{R}} \in \mathcal{V}_{\text{bad}}] \leq 1.5q^2/2^n$.

Good Transcripts. The analysis is fairly identical to the one of Lemma 13, and hence omitted.

The proof is concluded via (6.19) and above computations. □

Proof of Theorem 7

Let $K \in \{0,1\}^k$. Note that all evaluations of F_K are done in a tweakable manner, namely via (6.21). We replace these tweakable evaluations of F_K by a random tweakable compression function $\widetilde{R} \xleftarrow{\$} \widetilde{\text{Func}}([1, 2^{n/2}] \times [0,1] \times \{0,1\}^n \times \{0,1\}^n, \{0,1\}^{2n}, \{0,1\}^n)$. Note that for both confidentiality and integrity, the underlying $\widetilde{F}_K$ is invoked at most σ times. In other words, this step induces a penalty of (*cf.* Lemma 16)

$$\mathbf{Adv}^{\widetilde{\text{prf}}}_{\widetilde{F}}(\sigma, t) \leq \frac{1.5\sigma^2}{2^n} + \mathbf{Adv}^{\text{prf}}_{F}(3\sigma, t')\,,$$

where $t' \approx t$. This step leads to an idealized version of SPOEDNIC, called IDSPOEDNIC. IDSPOEDNIC is depicted in Figure 6.5. Concretely, we have obtained that

$$\mathbf{Adv}^{\text{conf}}_{\text{SPOEDNIC}}(\mathsf{nr}, q, \ell, \sigma, t) \leq \mathbf{Adv}^{\text{conf}}_{\text{IDSPOEDNIC}}(\mathsf{nr}, q, \ell, \sigma) + \frac{1.5\sigma^2}{2^n} + \mathbf{Adv}^{\text{prf}}_{F}(3\sigma, t')\,,$$

$$\mathbf{Adv}^{\text{int}}_{\text{SPOEDNIC}}(\mathsf{n}, q_{\mathsf{E}}, q_{\mathsf{D}}, \ell, \sigma, t) \leq \mathbf{Adv}^{\text{int}}_{\text{IDSPOEDNIC}}(\mathsf{n}, q_{\mathsf{E}}, q_{\mathsf{D}}, \ell, \sigma) + \frac{1.5\sigma^2}{2^n} + \mathbf{Adv}^{\text{prf}}_{F}(3\sigma, t')\,,$$

where $\mathsf{n} \in \{\mathsf{nr}, \mathsf{nm}\}$, and where t dropped out of the advantage function for IDSPOEDNIC because it has become irrelevant. The remainder of the proof centers around this scheme. For the nonce-respecting setting, the bounds of Lemma 14 and Lemma 15 carry over almost verbatim, with the same security bound. We consider integrity in the nonce-misuse setting in Lemma 17 and prove that $\mathbf{Adv}^{\text{int}}_{\text{IDSPOEDNIC}}(\mathsf{nm}, q_{\mathsf{E}}, q_{\mathsf{D}}, \ell, \sigma) \leq \frac{\ell q_{\mathsf{E}}^2/2}{2^n} + \frac{\ell q_{\mathsf{E}} q_{\mathsf{D}}}{2^n} + \frac{q_{\mathsf{D}}}{2^\tau}$.

Lemma 17. *The advantage of any nonce-misusing adversary trying to break the integrity of* IDSPOEDNIC *is bounded as:*

$$\mathbf{Adv}^{\text{int}}_{\text{IDSPOED}}(\mathsf{nm}, q_{\mathsf{E}}, q_{\mathsf{D}}, \ell, \sigma) \leq \frac{\ell q_{\mathsf{E}}^2/2}{2^n} + \frac{\ell q_{\mathsf{E}} q_{\mathsf{D}}}{2^n} + \frac{q_{\mathsf{D}}}{2^\tau}\,.$$

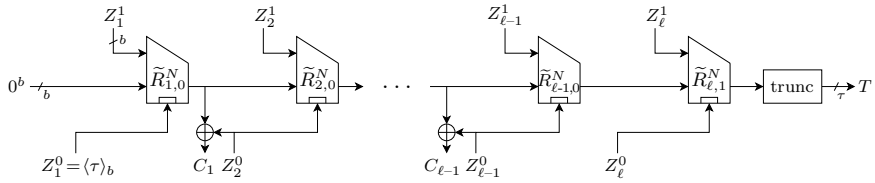


Figure 6.5: IdSpoednic encryption, which outputs $C = \text{left}_{|M|}(C_1 \| \cdots \| C_{\ell-1})$ and T. The boxes in $\widetilde{R}$ indicate that Z_i^0 also functions as a tweak.

Proof. At a high level, the proof follows the one of Lemma 15, with the difference that now, potentially, nonces may be the same leading to a slightly different bound.

Assume that $\mathbf{A}$ has made encryption queries (N^j, A^j, M^j) for $j = 1, \ldots, q_{\mathsf{E}}$, and denote the ciphertexts and tags by (C^j, T^j). Write $(Z_1^j, \ldots, Z_{\ell^j}^j) = \text{GPAD}_{n,\tau}(A^j, M^j)$ and denote the in- and outputs of the random functions by (s_i^j, t_i^j) for $i = 1, \ldots, \ell^j$.

Denote by $\mathsf{col}\mathcal{E}$ the event that there exist two distinct encryption queries j, j' with $N^j = N^{j'}$, and an index $i \in \{1, \ldots, \ell^j\}$, such that

$$t_{i-1}^j \| Z_i^{1j} \neq t_{i-1}^{j'} \| Z_i^{1j'} \ \wedge \ t_i^j = t_i^{j'} .$$

We have, using shorthand notation $[i = \ell]$ for 0 if $i \neq \ell$ and 1 if $i = \ell$,

$$\Pr[\mathsf{col}\mathcal{E}] \leq$$

$$\sum_{\substack{j,j'=1 \\ j \neq j'}}^{q_{\mathsf{E}}} \sum_{i=1}^{\min\{\ell^j, \ell^{j'}\}} \Pr[s_i^j \neq s_i^{j'} \wedge \widetilde{R}_{i,[i=\ell^j]}^N(Z_i^{0j}, s_i^j) = \widetilde{R}_{i,[i=\ell^{j'}]}^N(Z_i^{0j'}, s_i^{j'})] \leq \frac{\ell \binom{q_{\mathsf{E}}}{2}}{2^n} . \tag{6.23}$$

The remainder of the analysis is under the assumption that $\neg\mathsf{col}\mathcal{E}$ applies, and we add the term of (6.23) at the end.

Consider any forgery attempt (N, A, C, T), and denote its length by ℓ. Denote the message computed upon decryption by M. Refer to the state values as (s_i, t_i) for $i = 1, \ldots, \ell$, and write $(Z_1, \ldots, Z_\ell) = \text{GPAD}_{n,\tau}(A, M)$. The forgery is successful if $T = \text{left}_\tau(t_\ell)$.

Denote by $\mathsf{col}\mathcal{D}$ the event that there exists an encryption query j with $N^j = N$, $\ell^j = \ell$, and an index $i \in \{1, \ldots, \ell\}$, such that

$$t_{i-1}^j \| Z_i^{1j} \neq t_{i-1} \| Z_i^1 \ \wedge \ t_i^j = t_i .$$

We have

$$\Pr[\mathsf{col}\mathcal{D} \mid \neg\mathsf{col}\mathcal{E}] \leq \sum_{j=1}^{q_{\mathsf{E}}} \sum_{i=1}^{\ell} \Pr[s_i^j \neq s_i \wedge \widetilde{R}^N_{i,[i=\ell]}(s_i^j) = \widetilde{R}^N_{i,[i=\ell]}(s_i)] \leq \frac{\ell q_{\mathsf{E}}}{2^n}\,. \tag{6.24}$$

We make the following case distinction:

(i) $N \notin \{N^1, \ldots, N^{q_{\mathsf{E}}}\}$. This particularly means that $\widetilde{R}$ has never been queried for tweak $(\ell, 1, Z_\ell^0, N)$, and thus that $\widetilde{R}^N_{\ell,1}(Z_\ell^0, \cdot)$ responds with $t_\ell \xleftarrow{\$} \{0,1\}^n$. The forgery is successful with probability $1/2^\tau$;

(ii) $N = N^j$ for $j \in \{1, \ldots, q'_{\mathsf{E}}\}$ for some $1 \leq q'_{\mathsf{E}} \leq q_{\mathsf{E}}$. Note that we have implicitly reordered the encryption queries such that the ones for nonce N are the first q'_{E}. This is without loss of generality, as the different evaluations of IdSpoednic for different tweaks are independent. We proceed with a further case distinction:

- $\ell \notin \{\ell^1, \ldots, \ell^{q'_{\mathsf{E}}}\}$. This, again, means that $\widetilde{R}$ has never been queried for tweak $(\ell, 1, Z_\ell^0, N)$. The forgery is successful with probability $1/2^\tau$;
- $\ell = \ell^j$ for $j \in \{1, \ldots, q''_{\mathsf{E}}\}$ for some $1 \leq q''_{\mathsf{E}} \leq q'_{\mathsf{E}}$. Note that we have implicitly reordered the encryption queries such that the ones for nonce N and length ℓ are the first q''_{E}. This is, again, without loss of generality. We proceed with a further case distinction:
  - $s_\ell \notin \{s^1_{\ell^1}, \ldots, s^{q''_{\mathsf{E}}}_{\ell^{q''_{\mathsf{E}}}}\}$. In this case, $\widetilde{R}$ has been queried before for tweak $(\ell, 1, *, N)$, where $*$ denotes any tweak input Z_ℓ^{0j} which is left irrelevant, but never on input s_ℓ. Consequently, the response t_ℓ is uniformly drawn from $\{0,1\}^n$ and the forgery is successful with probability $1/2^\tau$;
  - $s_\ell = s^j_{\ell^j}$ for some $j \in \{1, \ldots, q''_{\mathsf{E}}\}$. As the forgery must be different from the encryption queries, as $\mathrm{GPAD}_{n,\tau}$ is an injective mapping, and moreover as $\neg\mathsf{col}\mathcal{E}$, this case implies the existence of a non-trivial state collision. Hence, the forgery is successful with probability at most $\Pr[\mathsf{col}\mathcal{D} \mid \neg\mathsf{col}\mathcal{E}]$.

Concluding, the forgery is successful with probability at most $\Pr[\mathsf{col} \mid \neg\mathsf{col}\mathcal{E}] + 1/2^\tau$, where $\Pr[\mathsf{col} \mid \neg\mathsf{col}\mathcal{E}]$ is bounded in (6.24). A summation over all q_{D} forgery attempts (*cf.* [22]) gives the final bound. □

6.5 Adding RUP Security to Encryption Schemes

In this section we introduce our generic method of adding RUP security to a class of encryption schemes. Following Shrimpton and Terashima [151], we take a modular approach in designing our construction. We start by describing a generic construction, and discuss informally why it enhances the security of the underlying encryption scheme. The generic construction achieves RUPAE, meaning it provides both authenticity and confidentiality even if unverified plaintext is released. A formal security argument for the construction is given in Section 6.5.2. In Section 6.5.3 we show how GCM's components can be used to instantiate the generic construction. Finally, in Section 6.5.4 we discuss how this construction can be used to prevent the crypto-tagging attack on Tor.

6.5.1 Generic Construction

Let $(\mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme with key space K, nonce space N, message space M, and ciphertext space C. Let (E, D) be a tweakable block cipher with $\mathsf{T} = \mathsf{N} \times \mathsf{C}$, $\mathsf{X} = \mathsf{N}$, and key space K. Let $\alpha \in \{0,1\}^\tau$ be some pre-defined constant. Then, define the separated AE scheme $(\mathsf{SEnc}, \mathsf{SDec}, \mathsf{SVer})$ as follows. The key space is K^2, with keys denoted by (K, L), the nonce space is N, the message space is M, and the ciphertext space is $\mathsf{N} \times \mathsf{C}$:

$$\mathsf{SEnc}^N_{K,L}(M) := \left(\mathsf{E}^C_L(N), C\right) \tag{6.25}$$

$$\text{with } C := \mathsf{Enc}^N_K(\alpha \parallel M) \tag{6.26}$$

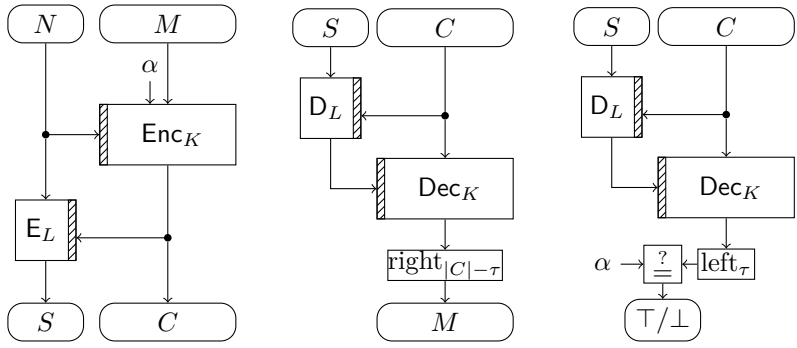
$$\mathsf{SDec}_{K,L}(S, C) := \text{right}_{|C|-\tau}\left(\mathsf{Dec}^{N'}_K(C)\right) \tag{6.27}$$

$$\text{with } N' := \mathsf{D}^C_L(S) \tag{6.28}$$

$$\mathsf{SVer}_{K,L}(S, C) := \left(\text{left}_\tau\left(\mathsf{Dec}^{N'}_K(C)\right) \stackrel{?}{=} \alpha\right). \tag{6.29}$$

The construction is depicted in Figure 6.6.

The construction adds robustness to the encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ by compressing the ciphertext via the tweak of the tweakable block cipher, and using that information to encrypt the nonce. As a result, during decryption, if any bit of the ciphertext is modified, then the ciphertext will result in a different tweak, and the tweakable block cipher will decrypt the nonce into some random value, which is used as the nonce for Dec. By assumption, Dec will output



(a) Encryption (b) Decryption (c) Verification

Figure 6.6: Adding RUP security to an existing encryption scheme

garbage, or more precisely, plaintext which is unrelated to any other plaintext queried.

Similarly, if the ciphertext is kept the same, and the encrypted nonce, S, is modified, then the tweakable block cipher will be queried on an input for which it has not been queried before with the given tweak computed from the ciphertext. As a result, the decryption of S will be random, and Dec's output will look random as well.

With respect to authenticity, our construction follows the encode-then-encipher paradigm [26], which uses redundancy in the plaintext in order to guarantee authenticity. The level of authenticity is determined by the length of the constant α (namely τ): if verification can be removed, then α's length is set to zero. However, the only requirement from α is to be known to both parties, and users may use any predictable bits already present in the plaintext, or the nonce if it is anyway synchronized.

6.5.2 Formal Security Argument for the Generic Construction

We start by defining two reductions which use an adversary $\mathbf{A}$ playing the RUPAE game against the construction $\mathbb{S} = (\mathsf{SEnc}, \mathsf{SDec}, \mathsf{SVer})$. Let $\mathbb{B} = (\mathsf{E}, \mathsf{D})$ denote the tweakable block cipher and $\mathbb{E} = (\mathsf{Enc}, \mathsf{Dec})$ the encryption scheme. Furthermore, let $\$_{\mathbb{S}} = (\$_{\mathsf{SEnc}}, \$_{\mathsf{SDec}}, \perp)$, $\$_{\mathbb{B}} := (\pi, \pi^{-1})$, where (π, π^{-1}) is from

the definition of SPRP security in Eq. (6.16), and $\$_{\mathbb{E}} := (\$_{\mathsf{Enc}}, \$_{\mathsf{Dec}})$. Then we define the following two reductions:

1. A reduction $\mathbf{B}\langle\mathbf{A}\rangle$ to the SPRP quality of the tweakable block cipher $\mathbb{B}$, meaning $\mathbf{B}\langle\mathbf{A}\rangle$ will attempt to distinguish $\mathbb{B}$ from $\$_{\mathbb{B}}$, using $\mathbf{A}$, an algorithm which is expecting either $\mathbb{S}$ or $\$_{\mathbb{S}}$. The reduction $\mathbf{B}$ generates a key K independently, and uses K to simulate the encryption scheme $\mathbb{E}$. Then, $\mathbf{B}$ runs $\mathbf{A}$, and responds to $\mathbf{A}$'s queries by reconstructing $\mathbb{S}$ using its own oracles, either $\mathbb{B}$ or $\$_{\mathbb{B}}$, and the simulated $\mathbb{E}$.
2. A reduction $\mathbf{C}\langle\mathbf{A}\rangle$ to the SRND quality of the encryption scheme $\mathbb{E}$. In contrast with $\mathbf{B}$, the reduction $\mathbf{C}$ simulates $\$_{\mathbb{B}}$ instead of $\mathbb{B}$. Then using its own oracles, either $\mathbb{E}$ or $\$_{\mathbb{E}}$, and $\$_{\mathbb{B}}$, $\mathbf{C}$ reconstructs $\mathbb{S}$.

Theorem 8. *The advantage of any nonce-respecting RUPAE adversary* $\mathbf{A}$ *attempting to distinguish* $\mathbb{S}$ *from* $\$_{\mathbb{S}}$*, making at most* q SEnc *queries, and at most* v SDec *and* SVer *queries, is upper bounded by*

$$2\frac{v(q+v)}{|\mathsf{N}|-q-v} + \frac{v}{2^{\tau}} + \mathsf{SPRP}_{\mathbb{B}}(\mathbf{B}\langle\mathbf{A}\rangle) + \mathsf{SRND}_{\mathbb{E}}(\mathbf{C}\langle\mathbf{A}\rangle)\,. \tag{6.30}$$

Proof. Let $\mathbb{S}^{\Pi,\Sigma}$ denote $\mathbb{S}$ using Π as tweakable block cipher and Σ as encryption scheme. By definition, we seek to bound

$$\mathsf{RUPAE}(\mathbf{A}) = \underset{\mathbf{A}}{\Delta}\left(\mathbb{S}^{\mathbb{B},\mathbb{E}}\,;\,\$_{\mathbb{S}}\right)\,. \tag{6.31}$$

Applying the triangle inequality, we get

$$\underset{\mathbf{A}}{\Delta}\left(\mathbb{S}^{\mathbb{B},\mathbb{E}}\,;\,\$_{\mathbb{S}}\right) \le \underset{\mathbf{A}}{\Delta}\left(\mathbb{S}^{\mathbb{B},\mathbb{E}}\,;\,\mathbb{S}^{\$_{\mathbb{B}},\mathbb{E}}\right) + \underset{\mathbf{A}}{\Delta}\left(\mathbb{S}^{\$_{\mathbb{B}},\mathbb{E}}\,;\,\$_{\mathbb{S}}\right) \tag{6.32}$$

Using reduction $\mathbf{B}\langle\mathbf{A}\rangle$, we know that

$$\underset{\mathbf{A}}{\Delta}\left(\mathbb{S}^{\mathbb{B},\mathbb{E}}\,;\,\mathbb{S}^{\$_{\mathbb{B}},\mathbb{E}}\right) \le \underset{\mathbf{B}\langle\mathbf{A}\rangle}{\Delta}\left(\mathbb{B}\,;\,\$_{\mathbb{B}}\right)\,. \tag{6.33}$$

Therefore we can focus on

$$\underset{\mathbf{A}}{\Delta}\left(\mathbb{S}^{\$_{\mathbb{B}},\mathbb{E}}\,;\,\$_{\mathbb{S}}\right) \tag{6.34}$$

which in turn is bounded from above by

$$\underset{\mathbf{A}}{\Delta}\left(\mathbb{S}^{\$_{\mathbb{B}},\mathbb{E}}\,;\,\mathbb{S}^{\$_{\mathbb{B}},\$_{\mathbb{E}}}\right) + \underset{\mathbf{A}}{\Delta}\left(\mathbb{S}^{\$_{\mathbb{B}},\$_{\mathbb{E}}}\,;\,\$_{\mathbb{S}}\right)\,. \tag{6.35}$$

The analysis of these two remaining terms relies on computing the probability that $\mathbf{A}$ makes a query which results in a nonce collision during a decryption

query, thereby violating the SRND game's requirement. In the analysis below, we find that the probability of such an event is at most

$$\varepsilon := \frac{v(q+v)}{|\mathsf{N}| - q - v}\,. \tag{6.36}$$

Therefore, using the reduction $\mathbf{C}\langle\mathbf{A}\rangle$ we know that the first term of (6.35) is bounded by

$$\varepsilon + \underset{\mathbf{C}\langle\mathbf{A}\rangle}{\Delta}(\mathbb{E}\,;\,\$_{\mathbb{E}})\,, \tag{6.37}$$

and the bound for

$$\underset{\mathbf{A}}{\Delta}\left(\mathbb{S}^{\$_{\mathbb{B}},\$_{\mathbb{E}}}\,;\,\$_{\mathbb{S}}\right) \tag{6.38}$$

is given below.

Say that $\mathbf{A}$ generates SEnc inputs (N_1, M_1), (N_2, M_2), ..., (N_q, M_q), and SDec and SVer inputs (S_1, C_1), (S_2, C_2), ..., (S_v, C_v), where (S_i, C_i) could be the input to either an SDec or SVer query. Let N_i^* denote the nonce input to $\$_{\mathsf{Dec}}$ resulting from the query (S_i, C_i), that is

$$N_i^* = \pi^{-1,C_i}(S_i)\,. \tag{6.39}$$

Similarly, define M_i^* and α_i^* such that

$$\alpha_i^* \| M_i^* = \$_{\mathsf{Dec}}^{N_i^*}(C_i)\,. \tag{6.40}$$

We call N_i^*, M_i^*, and α_i^* the "decrypted" nonces, plaintexts, and constants, respectively.

If the nonces N_i and N_j^* are distinct from each other then the SRND game's requirement is respected, hence $\$_{\mathbb{E}}$ will always give uniformly distributed and independent output.

Let event denote the event that either $N_i = N_j$ for $1 \le i < j \le q$, or $N_i^* = N_j^*$ for $1 \le i < j \le v$, or $N_i = N_j^*$ for $1 \le i \le q$ and $1 \le j \le v$. Then, by the fundamental lemma of game playing [27], (6.38) can be bounded by

$$\Pr[\mathsf{event}] + \Pr[\exists\, i \text{ s.t. } \alpha_i^* = \alpha \mid *\overline{\mathsf{event}}]\,, \tag{6.41}$$

where $\overline{\mathsf{event}}$ is the negation of event. Given $\overline{\mathsf{event}}$, the nonce input to $\$_{\mathsf{Dec}}$ will always be distinct, hence the α_i^* are independent and uniformly distributed, which means the quantity on the right is bounded above by $v/2^\tau$.

Therefore we focus on the probability of event, *i.e.*, that there is a collision in the N_i and N_j^*. By hypothesis, $\mathbf{A}$ is nonce-respecting, hence we know that $N_i \ne N_j$ for $1 \le i < j \le q$. Therefore we focus on the case that a decrypted nonce collides with some N_i, or another decrypted nonce.

Consider the query (S_i, C_i) associated to the i-th decrypted nonce N_i^*, and say that event has not yet been triggered. Let (N_j, M_j) be a previous SEnc query, and (S_j, C_j) its corresponding output. By hypothesis, $(S_j, C_j) \neq (S_i, C_i)$. If $C_j \neq C_i$, then the tweak input to (π, π^{-1}) will be different for the SEnc and SDec or SVer queries, hence the probability that N_i^* collides with N_j is at most $1/|\mathsf{N}|$. If $C_j = C_i$, then $S_j \neq S_i$, which means that (π, π^{-1}) is queried under the same tweak for both the SEnc and SDec or SVer queries. However, the probability that

$$N_j = \pi^{-1,C_j}(S_j) = \pi^{-1,C_i}(S_i) = N_i^* \tag{6.42}$$

is at most $1/(|\mathsf{N}| - q - v)$.

Now consider the probability that an SEnc query (N_i, M_i) is such that N_i equals N_j^* for some previous SDec or SVer query. Since the adversary's view is independent of N_j^*, it can guess N_j^* with probability at most $1/(|\mathsf{N}| - q - v)$. Therefore, the probability that a decrypted nonce collides with some nonce N_j is at most

$$\frac{qv}{|\mathsf{N}| - q - v}\,. \tag{6.43}$$

Given that no decrypted nonces collide with any nonce N_j, we are left with the event that two decrypted nonces collide with each other. However, similar reasoning as above shows that this probability is bounded above by

$$\frac{v^2}{|\mathsf{N}| - q - v}\,, \tag{6.44}$$

Putting the above computations together, if $\mathbf{A}$ makes q SEnc queries, and v SDec and SVer queries, then (6.38) is bounded above by

$$\frac{v(q+v)}{|\mathsf{N}| - q - v} + \frac{v}{2^\tau}\,. \tag{6.45}$$

□

6.5.3 GCM-RUP

We illustrate an instantiation of the construction using familiar primitives, namely those used to construct GCM [111,112]. The resulting instantiation uses three independent keys, but only makes three minor modifications to AES-GCM in order to achieve RUP security:

1. the plaintext is prepended by a string of zero bits of length τ;

2. the nonce N instead of $\mathsf{GHASH}(\varepsilon, N)$ is used to generate the mask for the polynomial hash; and

3. the output of GHASH is XORed with the nonce before it is encrypted.

See Figure 6.7 for an illustration.

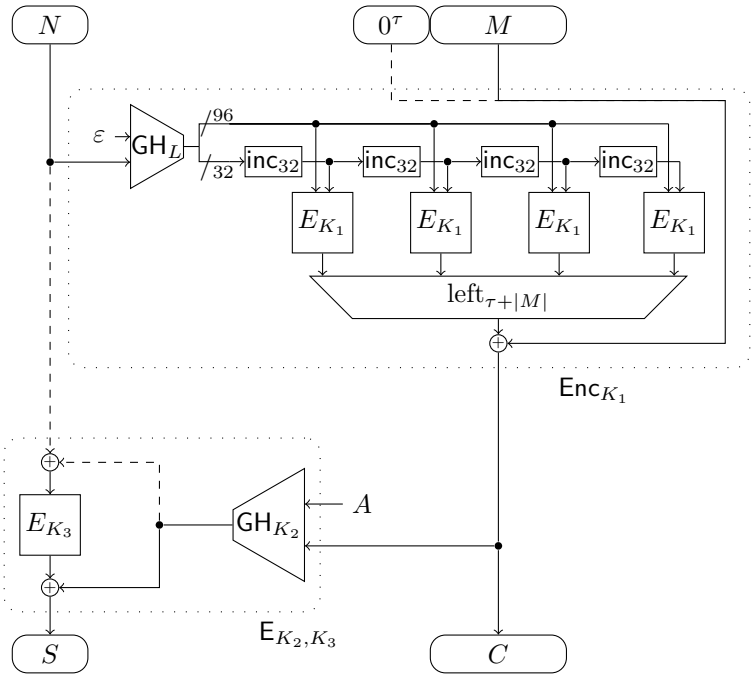


Figure 6.7: Instantiation of our construction using GCM's components. Changes from GCM are indicated using a dashed pattern, and the dotted boxes point out the underlying encryption scheme and tweakable block cipher. Filled circles indicate duplication of the values. GH is GHASH, and $/_m$ indicates the number of bits on a wire. The value L is $E_{K_1}(0^n)$, A represents associated data, and ϵ is the empty string.

Section 1.2.5 contains a description of the GCM components that we use to describe the instantiation, including the function GHASH, defined in Algorithm 1.1, and CTR mode, defined in Algorithm 1.2. Note that our formalization above did not include associated data, whereas GCM-RUP does, however it is straightforward to extend the definitions and generic construction to include it.

Since the generic construction's security relies on generating random nonce input during decryption, in order to maintain the same security level as GCM, the nonce size in the instantiation is fixed to be the same as the block size. The encryption scheme underlying the instantiation, $(\mathsf{Enc}, \mathsf{Dec})$, is the same as GCM without authentication, or in other words CTR mode, therefore Enc and Dec are identical, and so the SRND quality of $(\mathsf{Enc}, \mathsf{Dec})$ can be measured by looking only at Enc-queries. This allows us to use the GCM confidentiality result of Niwa *et al.* [121, 122], which gives $(\mathsf{Enc}, \mathsf{Dec})$ an SRND-bound of

$$\frac{0.5(\sigma + q + d + 1)^2}{2^n} + \frac{64 \cdot q(\sigma + q + d)}{2^n}\,, \tag{6.46}$$

where σ is the total number of blocks queried, q the number of Enc queries, d the number of Dec queries, and the nonce length is n bits, which is the block size as well.

Security of the underlying tweakable block cipher follows from the XTX construction of Minematsu and Iwata [117], where we extend the tweak space of a block cipher to arbitrary tweak size by XORing GHASH to both the input and output of the block cipher. Hence the SPRP-quality of the underlying tweakable block cipher is

$$\frac{q^2(\ell + 1)}{2^n}\,, \tag{6.47}$$

where q is the total number of queries made to the tweakable block cipher, and ℓ is the maximal tweak length, or in other words, the maximal ciphertext and associated data length in blocks.

Putting together the results along with the result of Section 6.5.2, we get the following bound for the instantiation.

Theorem 9. *Let* $\mathbf{A}$ *be a RUPAE-adversary against the instantiation making at most* q SEnc *queries, and* v SDec *and* SVer *queries. Say that at most* σ *blocks are queried, with* ℓ *the maximum ciphertext and associated data block length of any query, then* $\mathbf{A}$*'s advantage is at most*

$$\frac{0.5(\sigma + q + v + 1)^2}{2^n} + \frac{64 \cdot q(\sigma + q + v)}{2^n} + \frac{(q+v)^2(\ell+1)}{2^n} + 2\frac{v(q+v+1)}{2^n - q - v}\,. \tag{6.48}$$

If $q + v \leq 2^{n-1}$, then since $q + v \leq \sigma$, the bound can be simplified to

$$\frac{3 \cdot 64 \cdot \sigma^2}{2^n} + \frac{\sigma^2(\ell + 1)}{2^n}\,, \tag{6.49}$$

which is similar to GCM's security bounds [84, 121].

6.5.4 Tor

Tor [66] is a circuit-based low-latency anonymous communication service. The core idea underlying Tor is *onion routing*, a distributed overlay network designed to anonymize TCP-based applications, presented by Syverson, Reed and Goldschlag in [153].

Generally speaking, Tor communication is encrypted and relayed by nodes in the Tor-network via *circuits*. When building circuits, clients exchange keys with several nodes, usually 3, where each node only knows its predecessor and successor.

Clients prepare messages using multiple layers of encryption. First, the message is encrypted using the key and nonce shared with the circuit's last node. The resulting ciphertext is then encrypted again with the keys and nonce of the one-before-last node. This process is repeated for each node, until the first node's key is used.

The output of the multi-layered encryption is then sent from the client to the first node, which decrypts one layer, and forwards the result to the next node. In every step, another layer of encryption is removed, and the message is passed forward, until it reaches the last node. The last node authenticates and forwards the message to the intended recipient outside of the Tor network.

The crypto-tagging attack

By design, the Tor protocol offers an end-to-end integrity check, which the exit node handles by computing a SHA-1 digest of the decrypted message. Such a check prevents *e.g.*, attacks by rogue nodes which "tag" the encrypted message, and then search outbound communication for the corresponding corrupted traffic.

In 2012, an anonymous email was sent to the Tor developers mailing list describing the *crypto-tagging attack* [154]. In this attack, two nodes, the first and last, collude by tagging and untagging messages upon entry and exit to the network, respectively, thereby making the changes transparent to all other parties.[3] Due to the mode of operation used for encryption, namely CTR mode, the corrupted bits do not spread across the word and are maintained through all decryptions. They can then be removed by the exit node by just knowing their location. Furthermore, since the integrity check is only performed by the exit

[3]Tagging can be done in several ways. We mention here only one: the entry node XORs an identifying string to the message they are passing. Untagging is done by XORing the same identifier by the exit node.

node, the corruption cannot be detected by intermediate nodes in the circuit. Moreover, the attack is amplified by the fact that if only one of the nodes (*i.e.*, either the entry node or the exit node) is malicious, the tagged message cannot be verified, and the circuit is destroyed. Any new circuit where only one of the nodes is malicious will also be destroyed, thus biasing the set of possible circuits towards compromised ones.

An obvious solution to this problem is to add an authentication tag to each layer of the encryption, allowing intermediate nodes to verify the data passed through them and act according to some policy. However, in the discussion following the attack, such a solution was ruled out due to two main problems: (i) by adding authentication tags, the available bandwidth for sending messages is reduced, and (ii) the circuit's length could be revealed, an undesirable property in such systems.

Avoiding the attack

We propose a different approach allowing intermediate nodes to release unverified plaintext, using the generic construction proposed in Section 6.5.1. The only change from the above procedure for preparing the message is how the nonces are chosen.

As before, Tor-clients start by encrypting the plaintext with the key and nonce of the last node using CTR mode. Then, the ciphertext is compressed and used as a tweak for the encryption of the nonce as per Figure 6.6a. Afterwards, the encrypted nonce, S, is used as the nonce for the next layer of encryption, *i.e.*, with the keys of the one-before last node. This is repeated for each node of the circuit all the way to the first one. The result is a multi-layered application of our construction where the first layer receives the nonce and the plaintext as input, and each subsequent layer receives the previous layer's output. The new RUP secure layered encryption mode of operation is presented in Figure 6.8, where each layer can be realized using *e.g.*, the robust version of GCM presented in Section 6.5.3 with $|\alpha| = 0$.

When the message is ready, the client sends the ciphertext, along with the 3-times encrypted nonce to the first node. The first node uses the decryption algorithm as per Figure 6.6b to remove the outermost encryption, and forwards the result, as well as the now 2-times encrypted nonce, to the next node. After the last layer of encryption is removed by the last node, it authenticates the message and sends it to the intended recipient.

The security against an adversary trying to mount the crypto-tagging attack comes from the fact that any change to the ciphertext will affect the entire

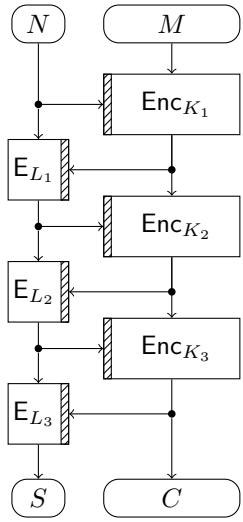


Figure 6.8: A RUP-secure 3-node layered encryption. The three layers are distinguished by their keys: $(K_1, L_1), (K_2, L_2)$, and (K_3, L_3).

message, effectively decrypting it to garbage. In other words, once decrypted by a non-colluding node, the crypto-tag corrupts the nonce, which will then be used to decrypt the message into garbage. Using the Tor terminology, by the time the message reaches the exit node, the crypto-tag can no longer be removed and the message is unrecognizable and should thus be dropped and the circuit is torn down.

For example, consider a circuit with three nodes, and say that (S_1, C_1), (S_2, C_2), and (S_3, C_3) are the outputs of the first, second, and third layers of encryption, respectively. In particular, the client uses (N, P) to produce (S_1, C_1), then (S_1, C_1) to produce (S_2, C_2), and (S_2, C_2) to produce (S_3, C_3). Finally, (S_3, C_3) is sent to the first node. Say that the first node is malicious. It decrypts (S_3, C_3) and obtains (S_2, C_2), then proceeds to tag (S_2, C_2) and passes (S'_2, C'_2) instead of (S_2, C_2) as it should do. Then, assuming the second node is honest, it will follow the protocol and decrypt (S'_2, C'_2). However, by the properties of our construction, we know that the decryption will be random since $(S_2, C_2) \neq (S'_2, C'_2)$, and in particular, the first node will not be able to predict anything about (S'_1, C'_1), *i.e.*, the decryption of (S'_2, C'_2). As a result, the second node will pass (S'_1, C'_1) to the third node, and the third node will not be able to conclude anything, regardless of whether it shares information with the first

node or not. More specifically, it would not be able to conclude the source and the destination of the message.

The disadvantage to our approach is that 16 extra bytes must be expropriated in order to send the encrypted nonce S. However, unlike adding per-hop authentication tags, the reduction in available bandwidth to send messages is fixed with respect to the circuit length. Furthermore, the solution can be built efficiently using familiar components, and is simple enough to allow for fast deployment. Finally, since the construction can use the plaintext's inherent redundancy for authentication, 8 bytes can be saved by dropping Tor's SHA-1 end-to-end authentication and using the algorithm described in Figure 6.6c instead.

6.6 Summary and Future Work

In this chapter we apply the lessons we learned about cryptanalysis in the previous chapters to build secure systems. The first contribution is a security analysis of the authentication service offered by the GALILEO project. This analysis was performed as part of a larger project to design this service. After choosing the TESLA protocol and adapting it the project's needs, the analysis provided in this Thesis was used to estimate the effort required by an adversary who wants to attack the system, and to derive the security paramaters (*e.g.*, key sizes) that make such effort infeasible.

The second contribution (spanning over two sections in this chapter) was to fix and improve P-OMD. The resulting schemes SPOED and SPOEDNIC are simplified, more efficient versions of P-OMD. SPOEDNIC also restores the lost nonce-misuse resistance of P-OMD.

The third contribution is a generic RUP-secure construction. After presenting the construction we show how RUP security can be easily added to GCM by a few small modifications. We discussed the Tor crypto-tagging attack and showed how using a RUP-secure mode of operation would prevent it.

Future work

What is common to all contributions in this chapter is the use of mathematical tools to argue their security. Such arguments are more common in developing modes of operation (where it is called *provable security*) than in developing primitives. Future research should try to lend some provable security notions for the design of primitives (some of this is already being done). As a long term

vision, symmetric-key designers should take an example from their asymmetric-key counterparts and try to build primitives that are reducable to computationaly hard problems. Then, when complexity theorists will finally do their part and prove the existence of hard problems, symmetric-key cryptography would have a chain of reductions (from a mode of operation to a primitive; from a primitive to a hard problem) with quantifable security.

While trying to realize this long term vision, a more short term goal is to fix existing problems. A patch is being prepared for Tor to replace its *ad hoc* AES-CTR encryption and SHA-1 end-to-end authentication with our GCM-RUP design.

Chapter 7

Concluding Words

> *"It is not incumbent upon you to finish the task, but neither are you free to absolve yourself from it"*
>
> - Ethics of the Fathers (Pirkei Avot), *2:16*

It is not easy to conclude four years of work in a single chapter, especially when there is still so much work to be done. In a way, this Thesis is that conclusion. We shall nevertheless try to write a concluding chapter.[1]

In this Thesis I explored symmetric-key cryptography. Each of the previous chapters tries to offer a glimpse into a different area within it. Since each chapter already included a summary of its own content, as well as ideas for future research specific to that chapter, I will try to focus this section on explaining how each chapter fits within the larger picture, and how this Thesis fits within the field of cryptography as a whole.

Deciding on the proper order of the content in this Thesis was not an easy task. Nor was the decision about what to include and what to leave out. It is easy to make an arbitrary decision on what should be the title of each chapter, but a much harder job is to decide where each individual contribution belongs. Overall, I think I did a good job in making these decisions. Although I expect some readers would think a certain section more suitable in a different chapter, I will try to explain the flow I chose.

[1] Luckily, some of this Author's work was concluded in his master thesis, saving him the trouble of concluding 7 years of work.

As is always the case in academic writing, the first chapter provides an introduction. In writing this section I had to make a decision: should I start from the basics, define what a block cipher is from a mathematical point of view, explain that a hash functions is a symmetric-key primitive although it involves no key, or discuss the pros and cons of public-key encryption vis-à-vis symmetric-key encryption.

I have decided otherwise for several reasons. First, I view such introduction as being past-oriented which I find in contrast with my research which is meant to be future-oriented. Early in the writing I realized that in order to keep this Thesis to a reasonable size I would have to keep some of my work out. I had therefore preferred in this case the future over the past, and tried to maximize the inclusion of novel ideas and included in the introduction only those parts that I felt that are necessary to understand this Thesis. I find this approach to be beneficial to both experts in cryptography (who already know what a redactable signature is) and for non-experts (who do not need to understand the difference between a Hellman-table and a Rainbow-table).

Secondly, I felt that such structure better supports what a Ph.D dissertation is supposed to be—a way for the candidate to prove that they are capable of independent research. I have often heard that the only readers of my Thesis would be the jury. In this Thesis I offer the best of me and it is now left for their judgment. If, after reading this Thesis, they deem it insufficient and me incapable of independent research at this time, no amount of introduction I had to offer can change that. I hope that through the research presented in this Thesis I implicitly also proved that I know the difference between a block cipher and a mode of operation.

Thirdly, despite being widely accepted, there is yet another group of people who is likely to read this Thesis—future students I may have. Let this Thesis be their passage into working with me, how I view things and what kind of research I am (at the time of submission) interested in. Therefore, Chapter 1 is an introduction to this Thesis and not an introduction to cryptography. After a very brief explanation about the context of this Thesis it goes straight to business and provides what information is useful in reading the rest: the structure of this Thesis, mathematical background, and the structure of the analyzed cryptosystems.

My contribution starts in Chapter 2. In this chapter I present 4 contributions all pertaining to existing theory of cryptanalysis (a summary of these contributions can be found in Section 2.6). Existing theory is the giant whose shoulders are used to see further. Developing new theories on top of quicksand hinders scientific progress and I hope these contributions would help others avoid it. Interestingly, all contributions of Chapter 2 are in the area of linear cryptanalysis.

This is not to suggest that other areas are perfect and have nothing to correct, but it does seem that, given our reliance on it, linear cryptanalysis is still not nearly as well understood as it should be. I hope to spend much of my future research time on linear cryptanalysis.

In Chapter 3 I try my luck in climbing the giant's shoulders and see further. Two new cryptanalytic methods are presented: RX-cryptanalysis and linear cryptanalysis using low-bias approximations. In order not to repeat the ideas for further research already mentioned in Section 3.3, I will limit myself to saying that most cryptanalytic methods have not yet been discovered.

I have recently heard a fellow cryptographer occupying a middle level position in another university saying that there is no future in cryptanalysis as we have already achieved sufficient proficiency in designing symmetric-key primitives and there is nothing more to discover. I utterly reject this sentiment. If at all, we have passed a tipping point and are now designing insecure "lightweight" algorithms, unidentified as such because we have not yet discovered the methods for breaking them. It is very dangerous (and very natural) to believe that the current state of affairs is complete, and it is as natural for this to be proven wrong. Examples for this include mechanics (Newtonian vs. Quantum), crystallography (Bravais lattices vs. Quasicrystals), economics (Classical vs. Behavioral), and even cryptography (Symmetric vs. A-symmetric)! Admittedly, none of my new methods will revolutionize cryptanalysis, but I hope that both can offer insights as to where the revolution might lie.

Chapters 4–5 deal with those things that "have to be done". Developing theories and never using them is not called cryptography but mathematics. To justify receiving tax payers money, a scientist must contribute to society. The way I chose to do so is by evaluating the cryptographic security of the systems people are using. Specifically, I attempted to do so by searching for undesirable properties, either using automated tools, or manually. I tried to limit myself to high-impact algorithms that are either being used or likely to be used by the general public. Simon, Speck, and Gost2 are all algorithms designed by governments which, as we are seeing again and again, can force their adoption within products. An exception is p-omd which impressed me with its ingenuity in processing the associated data part. While studying the trick they used I realized that it was insecure. I outline ideas for further research on automated tools in Section 4.3 and propose a systematic review of existing algorithms in Section 5.4.

Finally, in Chapter 6 I put what I learned into use by helping in the design of secure systems. As the title of this Thesis suggests, I am not a designer nor do I not wish to become one. As a cryptanalyst, I can and did offer unique insights on what is possible for an adversary to do and how to avoid certain

problems. I find the result to be quite satisfying. Not many can say that they affected the whole humanity by sending something to space as I did in being a cog in the Galileo project. Not many can claim to offer help to so many individuals unable to communicate freely as I am doing with the patch fixing the Tor crypto-tagging attack. In between, while I do not expect Spoed and Spoednic to attain wide use, I am proud in helping to salvage the clever trick used by the p-omd designers, and hope that the idea of blinding a value before absorbing it into the state will gain popularity. Other, long- and middle term ideas for future research are discussed in Section 6.6.

There is yet so much for cryptography to achieve. Reflecting on the amount of effort I spent on doing so little I realize that the cliché about science being a team effort is true. I hope that my contributions, past, present, and future, would help advance cryptanalysis, cryptography, science, and humankind.

The Author would like to thank whomever read this Thesis in full, reaching this point. You are awesome.

—Concluded here on Dec. 22nd in Leuven—

Appendix A

Generalized Padding

We describe here the general padding scheme used by SPOED and SPOEDNIC. This algorithm for a general padding scheme was published in [15] together with SPOED and SPOEDNIC. It was not included as an integral part of this Thesis since this Author was not a main contributor to this part of the work. It is included here since it is necessary for the understanding of Sections 6.3–6.4.

We define the generalized padding function

$$\mathrm{GPAD}_{n,\tau} : \{0,1\}^* \times \{0,1\}^* \to \left(\{0,1\}^{2n}\right)^+ .$$

It is indexed by state sizes n, τ, and maps the associated data and message to generalized message blocks. Formally, it is defined as follows: First, A (associated data) and X (message or ciphertext) are padded into n-bit message blocks $A_1 \| \cdots \| A_a = A \| 0^{n-(|A| \bmod n)}$ and $X_1 \| \cdots \| X_m = X \| 0^{n-(|X| \bmod n)}$, respectively. Denote $\ell = \max\left\{m, \lceil \frac{a+m}{2} \rceil\right\} + 1$, and define $\mathrm{len}(A, X) = \langle |A| \rangle_{n/2} \| \langle |X| \rangle_{n/2}$.[1] The function $\mathrm{GPAD}_{n,\tau}(A, X)$ outputs $Z_1, \ldots, Z_\ell$ as follows:

[1] As we show in Section 6.3.3 and Section 6.4.1, both ciphers claim only birthday-bound security, and the limitation of the length of X and A to $2^{n/2} - 1$ does not pose any issues.

if $a \le m$:		
Z_1	$=$	$\langle\tau\rangle_n \parallel A_1$
Z_2	$=$	$X_1 \parallel A_2$
$\cdots$		
Z_a	$=$	$X_{a-1} \parallel A_a$
Z_{a+1}	$=$	$X_a \parallel 0^n$
$\cdots$		
$Z_{\ell-1}$	$=$	$X_{m-1} \parallel 0^n$
Z_ℓ	$=$	$X_m \parallel \mathrm{len}(A, X)$

if $a > m$, $a + m$ even:		
Z_1	$=$	$\langle\tau\rangle_n \parallel A_1$
Z_2	$=$	$X_1 \parallel A_2$
$\cdots$		
Z_{m+1}	$=$	$X_m \parallel A_{m+1}$
Z_{m+2}	$=$	$A_{m+2} \parallel A_{m+3}$
$\cdots$		
$Z_{\ell-1}$	$=$	$A_{a-2} \parallel A_{a-1}$
Z_ℓ	$=$	$A_a \parallel \mathrm{len}(A, X)$

if $a > m$, $a + m$ odd:		
Z_1	$=$	$\langle\tau\rangle_n \parallel A_1$
Z_2	$=$	$X_1 \parallel A_2$
$\cdots$		
Z_{m+1}	$=$	$X_m \parallel A_{m+1}$
Z_{m+2}	$=$	$A_{m+2} \parallel A_{m+3}$
$\cdots$		
$Z_{\ell-1}$	$=$	$A_{a-1} \parallel A_a$
Z_ℓ	$=$	$0^n \parallel \mathrm{len}(A, X)$

The encoding of the message length is included in order to circumvent the need for a case distinction in the description of of the algorithms. Note that, in fact, almost any injective padding rule would do the job; however, for our purposes the described $\mathrm{GPAD}_{n,\tau}$ is the most suitable. We generically write $Z_i = Z_i^0 \parallel Z_i^1$, and denote $Z^\beta = Z_1^\beta \| \cdots \| Z_\ell^\beta$ for $\beta \in \{0, 1\}$.

Bibliography

[1] Abdelraheem, M. A., Ågren, M., Beelen, P., and Leander, G. On the distribution of linear biases: Three instructive examples. In Safavi-Naini and Canetti [141], pp. 50–67.

[2] Abed, F., List, E., Lucks, S., and Wenzel, J. Differential cryptanalysis of round-reduced Simon and Speck. In *Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers* (2014), C. Cid and C. Rechberger, Eds., vol. 8540 of *Lecture Notes in Computer Science*, Springer, pp. 525–545.

[3] Alizadeh, J., AlKhzaimi, H., Aref, M. R., Bagheri, N., Gauravaram, P., Kumar, A., Lauridsen, M. M., and Sanadhya, S. K. Cryptanalysis of SIMON variants with connections. In *Radio Frequency Identification: Security and Privacy Issues - 10th International Workshop, RFIDSec 2014, Oxford, UK, July 21-23, 2014, Revised Selected Papers* (2014), N. Saxena and A. Sadeghi, Eds., vol. 8651 of *Lecture Notes in Computer Science*, Springer, pp. 90–107.

[4] Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., and Yasuda, K. How to securely release unverified plaintext in authenticated encryption. In Sarkar and Iwata [144], pp. 105–125.

[5] Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Tischhauser, E., and Yasuda, K. Parallelizable and authenticated online ciphers. In Sako and Sarkar [142], pp. 424–443.

[6] Ashur, T., Bar-On, A., and Dunkelman, O. Cryptanalysis of GOST2. *IACR Trans. Symmetric Cryptol. 2017*, 1 (2017), 203–214.

[7] Ashur, T., Beyne, T., and Rijmen, V. Revisiting the wrong-key-randomization hypothesis. *IACR Cryptology ePrint Archive 2016* (2016), 990.

[8] Ashur, T., and Bodden, D. Linear cryptanalysis of reduced-round SPECK. In *Proceedings of the 37th Symposium on Information Theory in the Benelux* (2016).

[9] Ashur, T., Bodden, D., and Dunkelman, O. Linear cryptanalysis using low-bias linear approximations. *IACR Cryptology ePrint Archive 2017* (2017), 204.

[10] Ashur, T., De Witte, G., and Liu, Y. An automated tool for rotational cryptanalysis in the presence of constants of ARX-based primitives. In *Proceedings of the 38th Symposium on Information Theory in the Benelux, Werkgemeenschap voor Informatie- en Communicatietheorie* (2017), R. Heusdens and J. H. Weber, Eds., IEEE, pp. 59–66.

[11] Ashur, T., and Dunkelman, O. Linear analysis of reduced-round CubeHash. In *Applied Cryptography and Network Security - 9th International Conference, ACNS 2011, Nerja, Spain, June 7-10, 2011. Proceedings* (2011), J. Lopez and G. Tsudik, Eds., vol. 6715 of *Lecture Notes in Computer Science*, pp. 462–478.

[12] Ashur, T., Dunkelman, O., and Luykx, A. Boosting authenticated encryption robustness with minimal modifications. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III* (2017), J. Katz and H. Shacham, Eds., vol. 10403 of *Lecture Notes in Computer Science*, Springer, pp. 3–33.

[13] Ashur, T., and Liu, Y. Rotational cryptanalysis in the presence of constants. *IACR Trans. Symmetric Cryptol. 2016*, 1 (2016), 57–70.

[14] Ashur, T., and Mennink, B. Trivial nonce-misusing attack on pure OMD. *IACR Cryptology ePrint Archive 2015* (2015), 175.

[15] Ashur, T., and Mennink, B. Damaging, simplifying, and salvaging p-OMD. In *Information Security - 19th International Conference, ISC 2016, Honolulu, HI, USA, September 3-6, 2016, Proceedings* (2016), M. Bishop and A. C. A. Nascimento, Eds., vol. 9866 of *Lecture Notes in Computer Science*, Springer, pp. 73–92.

[16] Ashur, T., and Rijmen, V. On linear hulls and trails. In *Progress in Cryptology - INDOCRYPT 2016 - 17th International Conference on Cryptology in India, Kolkata, India, December 11-14, 2016, Proceedings* (2016), O. Dunkelman and S. K. Sanadhya, Eds., vol. 10095 of *Lecture Notes in Computer Science*, pp. 269–286.

[17] Ashur, T., and Rijmen, V. R&D support on Galileo open service navigation message authentication. Tech. rep., KU Leuven, 02 2017.

[18] Barrett, C., Stump, A., and Tinelli, C. The SMT-LIB standard: Version 2.0. Tech. rep., Department of Computer Science, The University of Iowa, 2010. www.SMT-LIB.org.

[19] Barwell, G., Page, D., and Stam, M. Rogue decryption failures: Reconciling AE robustness notions. In Groth [77], pp. 94–111.

[20] Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., and Wingers, L. The SIMON and SPECK families of lightweight block ciphers. *IACR Cryptology ePrint Archive 2013* (2013), 404.

[21] Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., and Wingers, L. Notes on the design and analysis of SIMON and SPECK. *IACR Cryptology ePrint Archive 2017* (2017), 560.

[22] Bellare, M., Goldreich, O., and Mityagin, A. The power of verification queries in message authentication and authenticated encryption. *IACR Cryptology ePrint Archive 2004* (2004), 309.

[23] Bellare, M., and Namprempre, C. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In Okamoto [126], pp. 531–545.

[24] Bellare, M., and Namprempre, C. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *IACR Cryptology ePrint Archive 2000* (2000), 25.

[25] Bellare, M., and Namprempre, C. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *J. Cryptology 21*, 4 (2008), 469–491.

[26] Bellare, M., and Rogaway, P. Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography. In Okamoto [126], pp. 317–330.

[27] Bellare, M., and Rogaway, P. The security of triple encryption and a framework for code-based game-playing proofs. In *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings* (2006), S. Vaudenay, Ed., vol. 4004 of *Lecture Notes in Computer Science*, Springer, pp. 409–426.

[28] Biham, E. On Matsui's linear cryptanalysis. In De Santis [63], pp. 341–355.

[29] Biham, E., Biryukov, A., and Shamir, A. Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. In *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding* (1999), J. Stern, Ed., vol. 1592 of *Lecture Notes in Computer Science*, Springer, pp. 12–23.

[30] Biham, E., and Shamir, A. Differential cryptanalysis of DES-like cryptosystems. *J. Cryptology 4*, 1 (1991), 3–72.

[31] Biryukov, A., De Cannière, C., and Quisquater, M. On multiple linear approximations. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings* (2004), M. K. Franklin, Ed., vol. 3152 of *Lecture Notes in Computer Science*, Springer, pp. 1–22.

[32] Biryukov, A., Velichkov, V., and Corre, Y. L. Automatic search for the best trails in ARX: application to block cipher Speck. In Peyrin [129], pp. 289–310.

[33] Bitmain. `https://www.minerslab.com/product/smart-miner-2-se-10ths-rack\-mount-bitcoin-miner/`.

[34] Blondeau, C., Gérard, B., and Tillich, J. Accurate estimates of the data complexity and success probability for various cryptanalyses. *Des. Codes Cryptography 59*, 1-3 (2011), 3–34.

[35] Blondeau, C., and Nyberg, K. Joint data and key distribution of the linear cryptanalysis test statistic and its impact to data complexity estimates of multiple/multidimensional linear and truncated differential attacks. *IACR Cryptology ePrint Archive 2015* (2015), 935.

[36] Blondeau, C., and Nyberg, K. Improved parameter estimates for correlation and capacity deviates in linear cryptanalysis. *IACR Trans. Symmetric Cryptol. 2016*, 2 (2016), 162–191.

[37] Blondeau, C., and Nyberg, K. Joint data and key distribution of simple, multiple, and multidimensional linear cryptanalysis test statistic and its impact to data complexity. *Des. Codes Cryptography 82*, 1-2 (2017), 319–349.

[38] Bodden, D. Linear cryptanalysis of reduced-round Speck, 2016. Master thesis, KU Leuven, T. Ashur, and V. Rijmen (promotors).

[39] Bogdanov, A., Kavun, E. B., Tischhauser, E., and Yalçin, T. Large-scale high-resolution computational validation of novel complexity

models in linear cryptanalysis. *J. Computational Applied Mathematics 259* (2014), 592–598.

[40] Bogdanov, A., and Rijmen, V. Zero-correlation linear cryptanalysis of block ciphers. *IACR Cryptology ePrint Archive 2011* (2011), 123.

[41] Bogdanov, A., and Rijmen, V. Linear hulls with correlation zero and linear cryptanalysis of block ciphers. *Des. Codes Cryptography 70*, 3 (2014), 369–383.

[42] Bogdanov, A., and Tischhauser, E. On the wrong key randomisation and key equivalence hypotheses in Matsui's Algorithm 2. In *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers* (2013), S. Moriai, Ed., vol. 8424 of *Lecture Notes in Computer Science*, Springer, pp. 19–38.

[43] Bogdanov, A., Tischhauser, E., and Vejre, P. S. Multivariate linear cryptanalysis: The past and future of PRESENT. *IACR Cryptology ePrint Archive 2016* (2016), 667.

[44] Brummayer, R., and Biere, A. Boolector: An efficient SMT solver for bit-vectors and arrays. In *Tools and Algorithms for the Construction and Analysis of Systems, 15th International Conference, TACAS 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings* (2009), S. Kowalewski and A. Philippou, Eds., vol. 5505 of *Lecture Notes in Computer Science*, Springer, pp. 174–177.

[45] Burgess, M. When a tanker vanishes, all the evidence points to Russia, September 2017. [Online; posted 21-September-2017].

[46] CAESAR: competition for authenticated encryption: Security, applicability, and robustness, May 2014. `http://competitions.cr.yp.to/caesar.html`.

[47] Canteaut, A., Ed. *Fast Software Encryption - 19th International Workshop, FSE 2012, Washington, DC, USA, March 19-21, 2012. Revised Selected Papers* (2012), vol. 7549 of *Lecture Notes in Computer Science*, Springer.

[48] Chabaud, F., and Vaudenay, S. Links between differential and linear cryptanalysis. In De Santis [63], pp. 356–365.

[49] Chen, S., Lampe, R., Lee, J., Seurin, Y., and Steinberger, J. P. Minimizing the two-round Even-Mansour cipher. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference,*

Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I (2014), J. A. Garay and R. Gennaro, Eds., vol. 8616 of *Lecture Notes in Computer Science*, Springer, pp. 39–56.

[50] Chen, S., and Steinberger, J. P. Tight security bounds for key-alternating ciphers. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings* (2014), P. Q. Nguyen and E. Oswald, Eds., vol. 8441 of *Lecture Notes in Computer Science*, Springer, pp. 327–350.

[51] Cho, J. Y. Linear cryptanalysis of reduced-round PRESENT. In *Topics in Cryptology - CT-RSA 2010, The Cryptographers' Track at the RSA Conference 2010, San Francisco, CA, USA, March 1-5, 2010. Proceedings* (2010), J. Pieprzyk, Ed., vol. 5985 of *Lecture Notes in Computer Science*, Springer, pp. 302–317.

[52] Cho, J. Y., and Pieprzyk, J. Multiple modular additions and crossword puzzle attack on NLSv2. In *Information Security, 10th International Conference, ISC 2007, Valparaíso, Chile, October 9-12, 2007, Proceedings* (2007), J. A. Garay, A. K. Lenstra, M. Mambo, and R. Peralta, Eds., vol. 4779 of *Lecture Notes in Computer Science*, Springer, pp. 230–248.

[53] Cogliani, S., Maimut, D., Naccache, D., do Canto, R. P., Reyhanitabar, R., Vaudenay, S., and Vizár, D. OMD: A compression function mode of operation for authenticated encryption. In Joux and Youssef [89], pp. 112–128.

[54] Cook, S. A. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA* (1971), M. A. Harrison, R. B. Banerji, and J. D. Ullman, Eds., ACM, pp. 151–158.

[55] Coron, J., Dodis, Y., Mandal, A., and Seurin, Y. A domain extender for the ideal cipher. In *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings* (2010), D. Micciancio, Ed., vol. 5978 of *Lecture Notes in Computer Science*, Springer, pp. 273–289.

[56] Courtois, N. Security evaluation of GOST 28147-89 in view of international standardisation. *IACR Cryptology ePrint Archive 2011* (2011), 211.

[57] Courtois, N. T., and Misztal, M. Differential cryptanalysis of GOST. *IACR Cryptology ePrint Archive 2011* (2011), 312.

[58] Daemen, J., Govaerts, R., and Vandewalle, J. Correlation matrices. In Preneel [131], pp. 275–285.

[59] Daemen, J., and Rijmen, V. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

[60] Daemen, J., and Rijmen, V. Probability distributions of correlation and differentials in block ciphers. *J. Mathematical Cryptology 1*, 3 (2007), 221–242.

[61] Daum, M. *Cryptanalysis of Hash Functions of the MD4-Family*. PhD thesis, Ruhr-Universität Bochum, 2005.

[62] De Leeuw, K. *Cryptology and statecraft in the Dutch Republic*. PhD thesis, University of Amsterdam, 2000.

[63] De Santis, A., Ed. *Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings* (1995), vol. 950 of *Lecture Notes in Computer Science*, Springer.

[64] De Witte, G. Automatic SAT-solver based search tools for cryptanalysis, 2017. Master thesis, KU Leuven, T. Ashur, Y. Liu, and V. Rijmen (promotors).

[65] Desmedt, Y., Ed. *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings* (1994), vol. 839 of *Lecture Notes in Computer Science*, Springer.

[66] Dingledine, R., Mathewson, N., and Syverson, P. F. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA* (2004), M. Blaze, Ed., USENIX, pp. 303–320.

[67] Dinur, I. Improved differential cryptanalysis of round-reduced Speck. In Joux and Youssef [89], pp. 147–164.

[68] Dinur, I., Dunkelman, O., and Shamir, A. Improved attacks on full GOST. In Canteaut [47], pp. 9–28.

[69] Dmukh, A., Dygin, D., and Marshalko, G. A lightweight-friendly modifcation of GOST block cipher. *IACR Cryptology ePrint Archive 2015* (2015), 65.

[70] Feller, W. *An Introduction to Probability Theory and Its Applications.* John Wiley & Sons, 1967. Exercice 10.

[71] Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., and Walker, J. The Skein hash function family. `http://skein-hash.info/`.

[72] Flajolet, P., and Odlyzko, A. M. Random mapping statistics. In *Advances in Cryptology - EUROCRYPT '89, Workshop on the Theory and Application of of Cryptographic Techniques, Houthalen, Belgium, April 10-13, 1989, Proceedings* (1989), J. Quisquater and J. Vandewalle, Eds., vol. 434 of *Lecture Notes in Computer Science*, Springer, pp. 329–354.

[73] Fleischmann, E., Forler, C., and Lucks, S. Mcoe: A family of almost foolproof on-line authenticated encryption schemes. In Canteaut [47], pp. 196–215.

[74] Fu, K., Wang, M., Guo, Y., Sun, S., and Hu, L. MILP-based automatic search algorithms for differential and linear trails for Speck. In Peyrin [129], pp. 268–288.

[75] Ganesh, V., and Dill, D. L. A decision procedure for bit-vectors and arrays. In *Computer Aided Verification, 19th International Conference, CAV 2007, Berlin, Germany, July 3-7, 2007, Proceedings* (2007), W. Damm and H. Hermanns, Eds., vol. 4590 of *Lecture Notes in Computer Science*, Springer, pp. 519–531.

[76] Granger, R., Jovanovic, P., Mennink, B., and Neves, S. Improved masking for tweakable blockciphers with applications to authenticated encryption. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I* (2016), M. Fischlin and J. Coron, Eds., vol. 9665 of *Lecture Notes in Computer Science*, Springer, pp. 263–293.

[77] Groth, J., Ed. *Cryptography and Coding - 15th IMA International Conference, IMACC 2015, Oxford, UK, December 15-17, 2015. Proceedings* (2015), vol. 9496 of *Lecture Notes in Computer Science*, Springer.

[78] Halevi, S., and Rogaway, P. A parallelizable enciphering mode. In *Topics in Cryptology - CT-RSA 2004, The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings* (2004), T. Okamoto, Ed., vol. 2964 of *Lecture Notes in Computer Science*, Springer, pp. 292–304.

[79] Harpes, C., Kramer, G. G., and Massey, J. L. A generalization of linear cryptanalysis and the applicability of Matsui's Piling-Up Lemma. In *Advances in Cryptology - EUROCRYPT '95, International Conference on the Theory and Application of Cryptographic Techniques, Saint-Malo, France, May 21-25, 1995, Proceeding* (1995), L. C. Guillou and J. Quisquater, Eds., vol. 921 of *Lecture Notes in Computer Science*, Springer, pp. 24–38.

[80] Hermelin, M., Cho, J. Y., and Nyberg, K. Multidimensional linear cryptanalysis of reduced round Serpent. In *Information Security and Privacy, 13th Australasian Conference, ACISP 2008, Wollongong, Australia, July 7-9, 2008, Proceedings* (2008), Y. Mu, W. Susilo, and J. Seberry, Eds., vol. 5107 of *Lecture Notes in Computer Science*, Springer, pp. 203–215.

[81] Huang, J., Vaudenay, S., Lai, X., and Nyberg, K. Capacity and data complexity in multidimensional linear attack. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I* (2015), R. Gennaro and M. Robshaw, Eds., vol. 9215 of *Lecture Notes in Computer Science*, Springer, pp. 141–160.

[82] Information technology — security techniques — encryption algorithms — part 3: Block ciphers. International Organization for Standardization, Geneva, Switzerland.

[83] Isobe, T. A single-key attack on the full GOST block cipher. In Joux [88], pp. 290–305.

[84] Iwata, T., Ohashi, K., and Minematsu, K. Breaking and repairing GCM security proofs. In Safavi-Naini and Canetti [141], pp. 31–49.

[85] Jean, J., Nikolic, I., and Peyrin, T. Tweaks and keys for block ciphers: The TWEAKEY framework. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II* (2014), P. Sarkar and T. Iwata, Eds., vol. 8874 of *Lecture Notes in Computer Science*, Springer, pp. 274–288.

[86] Jean, J., Nikolić, I., and Peyrin, T. Deoxys v1.3. CAESAR submissions, 2015. `http://competitions.cr.yp.to/round2/deoxysv13.pdf`.

[87] Jean, J., Nikolić, I., and Peyrin, T. Joltik v1.3. CAESAR submissions, 2015. `http://competitions.cr.yp.to/round2/joltikv13.pdf`.

[88] Joux, A., Ed. *Fast Software Encryption - 18th International Workshop, FSE 2011, Lyngby, Denmark, February 13-16, 2011, Revised Selected Papers* (2011), vol. 6733 of *Lecture Notes in Computer Science*, Springer.

[89] Joux, A., and Youssef, A. M., Eds. *Selected Areas in Cryptography - SAC 2014 - 21st International Conference, Montreal, QC, Canada, August 14-15, 2014, Revised Selected Papers* (2014), vol. 8781 of *Lecture Notes in Computer Science*, Springer.

[90] Jovanovic, P., Luykx, A., and Mennink, B. Beyond $2^{c/2}$ security in sponge-based authenticated encryption modes. In Sarkar and Iwata [144], pp. 85–104.

[91] Jovanovic, P., Luykx, A., and Mennink, B. Beyond $2^{c/2}$ security in sponge-based authenticated encryption modes. *IACR Cryptology ePrint Archive 2014* (2014), 373.

[92] Jr., B. S. K., and Robshaw, M. J. B. Linear cryptanalysis using multiple approximations. In Desmedt [65], pp. 26–39.

[93] Kahn, D. *The Codebreakers — The Story of Secret Writing.* Scribner, 1996.

[94] Kara, O. Reflection cryptanalysis of some ciphers. In *Progress in Cryptology - INDOCRYPT 2008, 9th International Conference on Cryptology in India, Kharagpur, India, December 14-17, 2008. Proceedings* (2008), D. R. Chowdhury, V. Rijmen, and A. Das, Eds., vol. 5365 of *Lecture Notes in Computer Science*, Springer, pp. 294–307.

[95] Keliher, L., Meijer, H., and Tavares, S. E. New method for upper bounding the maximum average linear hull probability for SPNs. In *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding* (2001), B. Pfitzmann, Ed., vol. 2045 of *Lecture Notes in Computer Science*, Springer, pp. 420–436.

[96] Khovratovich, D., and Nikolic, I. Rotational cryptanalysis of ARX. In *Fast Software Encryption, 17th International Workshop, FSE 2010, Seoul, Korea, February 7-10, 2010, Revised Selected Papers* (2010), S. Hong and T. Iwata, Eds., vol. 6147 of *Lecture Notes in Computer Science*, Springer, pp. 333–346.

[97] Khovratovich, D., Nikolic, I., Pieprzyk, J., Sokolowski, P., and Steinfeld, R. Rotational cryptanalysis of ARX revisited. In Leander [106], pp. 519–536.

[98] Knudsen, L. R. Truncated and higher order differentials. In Preneel [131], pp. 196–211.

[99] Knudsen, L. R., and Robshaw, M. *The Block Cipher Companion.* Information Security and Cryptography. Springer, 2011.

[100] Ko, Y., Hong, S., Lee, W., Lee, S., and Kang, J. Related key differential attacks on 27 rounds of XTEA and full-round GOST. In *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers* (2004), B. K. Roy and W. Meier, Eds., vol. 3017 of *Lecture Notes in Computer Science*, Springer, pp. 299–316.

[101] Krovetz, T., and Rogaway, P. The software performance of authenticated-encryption modes. In Joux [88], pp. 306–327.

[102] Krovetz, T., and Rogaway, P. The OCB authenticated-encryption algorithm. `http://datatracker.ietf.org/doc/draft-irtf-cfrg-ocb`, June 2013.

[103] Kölbl, S. CryptoSMT: An easy to use tool for cryptanalysis of symmetric primitives. `https://github.com/kste/cryptosmt`.

[104] Landecker, W., Shrimpton, T., and Terashima, R. S. Tweakable blockciphers with beyond birthday-bound security. In Safavi-Naini and Canetti [141], pp. 14–30.

[105] Langford, S. K., and Hellman, M. E. Differential-linear cryptanalysis. In Desmedt [65], pp. 17–25.

[106] Leander, G., Ed. *Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers* (2015), vol. 9054 of *Lecture Notes in Computer Science*, Springer.

[107] Leurent, G. ARXtools: a toolkit for arx analysis, 2012. `https://who.rocq.inria.fr/Gaetan.Leurent/arxtools.html`.

[108] Liskov, M., Rivest, R. L., and Wagner, D. Tweakable block ciphers. *J. Cryptology 24*, 3 (2011), 588–613.

[109] Liu, Y., Witte, G. D., Ranea, A., and Ashur, T. Rotational-XOR cryptanalysis of reduced-round SPECK. *IACR Trans. Symmetric Cryptol. 2017*, 3 (2017), 24–36.

[110] Matsui, M. Linear cryptanalysis method for DES cipher. In *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993,*

Proceedings (1993), T. Helleseth, Ed., vol. 765 of *Lecture Notes in Computer Science*, Springer, pp. 386–397.

[111] McGrew, D. A., and Viega, J. The security and performance of the Galois/counter mode (GCM) of operation. In *Progress in Cryptology - INDOCRYPT 2004, 5th International Conference on Cryptology in India, Chennai, India, December 20-22, 2004, Proceedings* (2004), A. Canteaut and K. Viswanathan, Eds., vol. 3348 of *Lecture Notes in Computer Science*, Springer, pp. 343–355.

[112] McGrew, D. A., and Viega, J. The security and performance of the Galois/counter mode of operation (full version). *IACR Cryptology ePrint Archive 2004* (2004), 193.

[113] Mennink, B. Optimally secure tweakable blockciphers. In Leander [106], pp. 428–448.

[114] Meurer, A., Smith, C. P., Paprocki, M., Certík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller, R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., Curry, M. J., Terrel, A. R., Roucka, S., Saboo, A., Fernando, I., Kulal, S., Cimrman, R., and Scopatz, A. M. SymPy: symbolic computing in Python. *PeerJ Computer Science 3* (2017), e103.

[115] Minematsu, K. Improved security analysis of XEX and LRW modes. In *Selected Areas in Cryptography, 13th International Workshop, SAC 2006, Montreal, Canada, August 17-18, 2006 Revised Selected Papers* (2006), E. Biham and A. M. Youssef, Eds., vol. 4356 of *Lecture Notes in Computer Science*, Springer, pp. 96–113.

[116] Minematsu, K. Beyond-birthday-bound security based on tweakable block cipher. In *Fast Software Encryption, 16th International Workshop, FSE 2009, Leuven, Belgium, February 22-25, 2009, Revised Selected Papers* (2009), O. Dunkelman, Ed., vol. 5665 of *Lecture Notes in Computer Science*, Springer, pp. 308–326.

[117] Minematsu, K., and Iwata, T. Tweak-length extension for tweakable blockciphers. In Groth [77], pp. 77–93.

[118] Mouha, N., and Preneel, B. A proof that the ARX cipher Salsa20 is secure against differential cryptanalysis. *IACR Cryptology ePrint Archive 2013* (2013), 328.

[119] Murphy, S. The effectiveness of the linear hull effect. *J. Mathematical Cryptology 6*, 2 (2012), 137–147.

[120] National Institute of Standards and Technology. DES modes of operation. FIPS 81, December 1980.

[121] Niwa, Y., Ohashi, K., Minematsu, K., and Iwata, T. GCM security bounds reconsidered. In Leander [106], pp. 385–407.

[122] Niwa, Y., Ohashi, K., Minematsu, K., and Iwata, T. GCM security bounds reconsidered. *IACR Cryptology ePrint Archive 2015* (2015), 214.

[123] Nyberg, K. Linear approximation of block ciphers. In De Santis [63], pp. 439–444.

[124] Nyberg, K. Statistical and linear independence of binary random variables. *IACR Cryptology ePrint Archive 2017* (2017), 432.

[125] Nyberg, K., and Wallén, J. Improved linear distinguishers for SNOW 2.0. In *Fast Software Encryption, 13th International Workshop, FSE 2006, Graz, Austria, March 15-17, 2006, Revised Selected Papers* (2006), M. J. B. Robshaw, Ed., vol. 4047 of *Lecture Notes in Computer Science*, Springer, pp. 144–162.

[126] Okamoto, T., Ed. *Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings* (2000), vol. 1976 of *Lecture Notes in Computer Science*, Springer.

[127] Patarin, J. The "coefficients h" technique. In *Selected Areas in Cryptography, 15th International Workshop, SAC 2008, Sackville, New Brunswick, Canada, August 14-15, Revised Selected Papers* (2008), R. M. Avanzi, L. Keliher, and F. Sica, Eds., vol. 5381 of *Lecture Notes in Computer Science*, Springer, pp. 328–345.

[128] Perrig, A., Canetti, R., Tygar, J. D., and Song, D. X. Efficient authentication and signing of multicast streams over lossy channels. In *2000 IEEE Symposium on Security and Privacy, Berkeley, California, USA, May 14-17, 2000* (2000), IEEE Computer Society, pp. 56–73.

[129] Peyrin, T., Ed. *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers* (2016), vol. 9783 of *Lecture Notes in Computer Science*, Springer.

[130] Pinsky, M. A. The normal approximation to the hypergeometric distribution. Unpublished manuscript. `https://www.dartmouth.edu/~chance/teaching_aids/books_articles/probability_book/pinsky-hypergeometric.pdf`.

[131] Preneel, B., Ed. *Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994, Proceedings* (1995), vol. 1008 of *Lecture Notes in Computer Science*, Springer.

[132] Ranea, A. An easy to use tool for rotational-XOR cryptanalysis of ARX block ciphers, 2017. Master thesis, KU Leuven, T. Ashur, Y. Liu, and V. Rijmen (promotors).

[133] Ranea, A., Ashur, T., and Liu, Y. An easy-to-use tool for rotational-XOR cryptanalysis of ARX block ciphers. In *Proceedings of the Romanian Academy, Series A* (2017), F. Negru, Ed., vol. 18, pp. 307–316.

[134] Reyhanitabar, R., Vaudenay, S., and Vizár, D. Boosting OMD for almost free authentication of associated data. In Leander [106], pp. 411–427.

[135] Reyhanitabar, R., Vaudenay, S., and Vizár, D. Boosting OMD for almost free authentication of associated data. FSE 2015 preprint version, 2015.

[136] Reyhanitabar, R., Vaudenay, S., and Vizár, D. Boosting OMD for almost free authentication of associated data. *IACR Cryptology ePrint Archive 2015* (2015), 302.

[137] Röck, A., and Nyberg, K. Generalization of Matsui's Algorithm 1 to linear hull for key-alternating block ciphers. *Des. Codes Cryptography 66*, 1-3 (2013), 175–193.

[138] Rogaway, P. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In *ASIACRYPT* (2004), P. J. Lee, Ed., vol. 3329 of *Lecture Notes in Computer Science*, Springer, pp. 16–31.

[139] Rogaway, P., Bellare, M., and Black, J. OCB: A block-cipher mode of operation for efficient authenticated encryption. *ACM Trans. Inf. Syst. Secur. 6*, 3 (2003), 365–403.

[140] Russian National Bureau of Standards. Federal information processing standard-cryptographic protection - cryptographic algorithm. GOST 28147-89, 1989.

[141] Safavi-Naini, R., and Canetti, R., Eds. *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings* (2012), vol. 7417 of *Lecture Notes in Computer Science*, Springer.

[142] Sako, K., and Sarkar, P., Eds. *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part I* (2013), vol. 8269 of *Lecture Notes in Computer Science*, Springer.

[143] Samajder, S., and Sarkar, P. Another look at normal approximations in cryptanalysis. *J. Mathematical Cryptology 10*, 2 (2016), 69–99.

[144] Sarkar, P., and Iwata, T., Eds. *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I* (2014), vol. 8873 of *Lecture Notes in Computer Science*, Springer.

[145] Schulte-Geers, E. On CCZ-equivalence of addition mod 2^n. *Des. Codes Cryptography 66*, 1-3 (2013), 111–127.

[146] Seki, H., and Kaneko, T. Differential cryptanalysis of reduced rounds of GOST. In *Selected Areas in Cryptography, 7th Annual International Workshop, SAC 2000, Waterloo, Ontario, Canada, August 14-15, 2000, Proceedings* (2000), D. R. Stinson and S. E. Tavares, Eds., vol. 2012 of *Lecture Notes in Computer Science*, Springer, pp. 315–323.

[147] Selçuk, A. A. On probability of success in linear and differential cryptanalysis. *J. Cryptology 21*, 1 (2008), 131–147.

[148] Shi, D., Hu, L., Sun, S., and Song, L. Linear (hull) cryptanalysis of round-reduced versions of KATAN. *IACR Cryptology ePrint Archive 2015* (2015), 964.

[149] Shi, D., Hu, L., Sun, S., Song, L., Qiao, K., and Ma, X. Improved linear (hull) cryptanalysis of round-reduced versions of SIMON. *IACR Cryptology ePrint Archive 2014* (2014), 973.

[150] Shi, D., Hu, L., Sun, S., Song, L., Qiao, K., and Ma, X. Improved linear (hull) cryptanalysis of round-reduced versions of SIMON. *SCIENCE CHINA Information Sciences 60*, 3 (2017), 39101:1–39101:3.

[151] Shrimpton, T., and Terashima, R. S. A modular framework for building variable-input-length tweakable ciphers. In Sako and Sarkar [142], pp. 405–423.

[152] Sun, S., Hu, L., Wang, M., Wang, P., Qiao, K., Ma, X., Shi, D., Song, L., and Fu, K. Towards finding the best characteristics of some bit-oriented block ciphers and automatic enumeration of (related-key)

differential and linear characteristics with predefined properties. *IACR Cryptology ePrint Archive 2014* (2014), 747.

[153] Syverson, P. F., Goldschlag, D. M., and Reed, M. G. Anonymous connections and onion routing. In *1997 IEEE Symposium on Security and Privacy, May 4-7, 1997, Oakland, CA, USA* (1997), IEEE Computer Society, pp. 44–54.

[154] The 23 Raccoons. Analysis of the relative severity of tagging attacks, Mar 2012. Email to the Tor developers mailing list `https://lists.torproject.org/pipermail/tor-dev/2012-March/003347.html`.

[155] US National Institute of Standards and Technology (NIST). Federal information processing standards publication 46-3, 1999.

[156] Vaudenay, S. An experiment on DES statistical cryptanalysis. In *CCS '96, Proceedings of the 3rd ACM Conference on Computer and Communications Security, New Delhi, India, March 14-16, 1996.* (1996), L. Gong and J. Stearn, Eds., ACM, pp. 139–147.

[157] Velichkov, V. Y. YAARX: yet another toolkit for the analysis of ARX cryptographic algorithms, 2016. `https://github.com/vesselinux/yaarx`.

[158] von Clausewitz, C. *Vom Kriege.* Reclam, 1832.

[159] Wagner, D. A. The boomerang attack. In *Fast Software Encryption, 6th International Workshop, FSE '99, Rome, Italy, March 24-26, 1999, Proceedings* (1999), L. R. Knudsen, Ed., vol. 1636 of *Lecture Notes in Computer Science*, Springer, pp. 156–170.

[160] Wang, G., Keller, N., and Dunkelman, O. The delicate issues of addition with respect to XOR differences. In *Selected Areas in Cryptography, 14th International Workshop, SAC 2007, Ottawa, Canada, August 16-17, 2007, Revised Selected Papers* (2007), C. M. Adams, A. Miri, and M. J. Wiener, Eds., vol. 4876 of *Lecture Notes in Computer Science*, Springer, pp. 212–231.

[161] Yao, Y., Zhang, B., and Wu, W. Automatic search for linear trails of the SPECK family. In *Information Security - 18th International Conference, ISC 2015, Trondheim, Norway, September 9-11, 2015, Proceedings* (2015), J. Lopez and C. J. Mitchell, Eds., vol. 9290 of *Lecture Notes in Computer Science*, Springer, pp. 158–176.

[162] Zhang, L., and Malik, S. The quest for efficient boolean satisfiability solvers. In *Automated Deduction - CADE-18, 18th International*

Conference on Automated Deduction, Copenhagen, Denmark, July 27-30, 2002, Proceedings (2002), A. Voronkov, Ed., vol. 2392 of *Lecture Notes in Computer Science*, Springer, pp. 295–313.

[163] Zheng, L., and Zhang, S. FFT-based multidimensional linear attack on PRESENT using the 2-bit-fixed characteristic. *Security and Communication Networks 8*, 18 (2015), 3535–3545.

Curriculum

Education:

Academic

- 2013 – 2017 — Ph.D student, KU Leuven.
- 2010 – 2013 — University of Haifa, Israel. M.Sc in Computer Science. Graduated *magna cum laude*.
- 2009 – 2008 — Studies of Computer Science and Applied Mathematics at the Weizmann Institute as a free listener (mostly cryptography related classes).
- 2004 – 2010 — The Open University, Israel. B.A in Computer Science and Management.
- 1998 – 2001 — The Herzliya Hebrew Gymnasium. Honor students' program, combining studies towards full matriculation with academic studies.

Awards and Scholarships

1. 2016, Best student paper award, The 37th Symposium on Information Theory in the Benelux.
2. 2014, Google Europe Scholarship for Students with Disabilities.
3. 2013, Excellence Scholarship for M.Sc. Studies, University of Haifa, Israel.
4. 2013, Outstanding Teaching Assistant Award, Department of Computer Science, University of Haifa, Israel.
5. 2012, National Cyber Scholarship, Israel's Ministry of Science.

Thesis Supervision

1. A. Ranea Robles, **An Easy to Use Tool for Rotational-XOR Cryptanalysis of ARX Block Ciphers**, Master thesis, KU Leuven, T. Ashur, Y. Liu, and V. Rijmen (promotors), 81 pages, 2017.
2. G. De Witte, **Automatic SAT-solver Based Search Tools for Cryptanalysis**, Master thesis, KU Leuven, T. Ashur, Y. Liu, and V. Rijmen (promotors), 61 pages, 2017.
3. D. Bodden, **Linear Cryptanalysis of Reduced-Round Speck**, Master thesis, KU Leuven, T. Ashur, and V. Rijmen (promotors), 11 pages, 2016.

Invited Panelist

1. "Government Works in the Public Domain — All your Tax-paid Content are Belong to us" in The 3rd Free Culture Research Conference — held in The Free University of Berlin, Germany, 09/10/2010.

Invited Presentations

1. "Breaching the Privacy of Israel's Paper Ballot Voting System", University of Kent, Canterbury, UK. 9/11/2016.
2. "Simon: NSA-designed cipher in the post Snowden world", Cryptoday 2015, Technion, Israel, 28/12/2015
3. "Provable Bounds for the Linear Properties of Simon", Technical University of Denmark, Lyngby, Denmark, 7/7/2014.
4. "How to Change the World while Still in your Slippers — Wikipedia, Wikimedia and the free Content Movement", 9th Jerusalem Conference on the Digitization of Cultural Heritage, the Van Leer Jerusalem Institute, Israel, 16/11/2011.

Teaching & Work Experience:

Teaching Assistant

- Probleemoplossen en ontwerpen, deel 5, KU Leuven — Winter 2016.
- Computers and Network Security, University of Haifa — Spring 2015.
- Probleemoplossen en ontwerpen, deel 3, KU Leuven — Winter 2013.
- Privacy Technologies, KU Leuven — Winter 2015.
- Database Systems, University of Haifa — Winter 2013.
- Software Engineering, University of Haifa — Spring 2012.

– Database Systems, University of Haifa — Winter 2012.

Exercise Checker

– Introduction to Cryptography at the University of Haifa — Winter 2013 (lecturer in charge: Dr. Dunkelman).
– Introduction to Artificial Intelligence at the University of Haifa — Spring 2012 (lecturer in charge: Prof. Manevitz).
– Computers and Network Security at the University of Haifa — Winter 2012 (lecturer in charge: Dr. Dunkelman).

Industry Employment Experience

– 2011 – 2013 — Chairman at Wikimedia Israel (voluntarily).
– 2010 – 2011 — Board member and treasurer at Wikimedia Israel (voluntarily).
– 2009 – 2010 — Chief Information Officer at Mediton Health-care services.
– 2007 – 2009 — Services & Support Manager, Safend inc.
– 2002 – 2007 — Army service as an officer in the Israeli Medical Corps. Separated as First Lieutenant (NATO OF-1). Current rank: Captain (NATO OF-2).

Media Coverage

– **English:**

1. Distrustful U.S. allies force spy agency to back down in encryption fight, https://www.reuters.com/article/us-cyber-standards-insight/distrustful-u-s-allies-force-spy-agency-to-back-down-in-encryption-fight-idUSKCN1BW0GV.
2. ISO decides not to approve two NSA encryption algorithms, citing trust issues, https://www.scmagazine.com/iso-refuses-to-approve-nsa-encryption-algorithms-as-industry-standards/article/690470/.
3. Distrustful U.S. allies force spy agency to back down in encryption fight, https://ca.news.yahoo.com/distrustful-u-allies-force-spy-agency-back-down-050318878.html.
4. Mistrust among US allies leads international experts to force NSA to step back from cryptography fight, http://www.firstpost.com/tech/news-analysis/mistrust-among-us-allies-leads-international-experts-to-force-nsa-to-step-back-from-cryptography-fight-4069507.html.
5. Distrustful US allies force spy agency to back down in encryption row, https://www.cnbc.com/2017/09/21/distrustful-us-allies-force-nsa-to-back-down-in-encryption-row.html.
6. 'Subversive' NSA forced to back down over cyber encryption techniques, https://www.rt.com/usa/404249-allies-nsa-encryption-techniques/.
7. Cyber Week in Review: September 22, 2017, https://www.cfr.org/blog/cyber-week-review-september-22-2017.
8. NSA Tried to Push Global Encryption Standards "Because It Knew How to Break Them", http://wccftech.com/us-allies-accuse-nsa-manipulating-encryption-backdoors/.

9. KU Leuven: Galileo signals will become more difficult to falsify, http://gpsworld.com/ku-leuven-galileo-signals-will-become-more-difficult-to-falsify/.
10. Technique to prevent the falsification of Galileo navigational signals, https://phys.org/news/2017-02-technique-falsification-galileo.html.
11. Falsifying Galileo satellite signals will become more difficult, https://www.sciencedaily.com/releases/2017/02/170210084541.htm.
12. Leuven duo ensure safe ride for Europe's navigation system, http://www.flanderstoday.eu/innovation/leuven-duo-ensure-safe-ride-europes-navigation-system.
13. Falsifying Galileo Satellite Signals Will Become More Difficult, https://www.ecnmag.com/news/2017/02/falsifying-galileo-satellite-signals-will-become-more-difficult.

– **Dutch:**

1. Door NSA ontwikkelde encryptie-algoritmen onder vuur, https://www.security.nl/posting/532069/Door+NSA+ontwikkelde+encryptie-algoritmen+onder+vuur.
2. NSA stopt standaardisering iot-encryptie na wantrouwen deskundigen, https://tweakers.net/nieuws/129901/nsa-stopt-standaardisering-iot-encryptie-na-wantrouwen-deskundigen.html.
3. Leuvense onderzoekers zorgen voor beveiliging satellietnavigatiesysteem Galileo, http://www.knack.be/nieuws/belgie/leuvense-onderzoekers-zorgen-voor-beveiliging-satellietnavigatiesysteem-galileo/article-belga-814487.html.

– **Hebrew:**

1. Tesla, https://i-hls.com/he/archives/74824
2. Op-ed in Hebrew about Israel national biometric database, https://www.haaretz.co.il/captain/room404/.premium-1.2083716
3. Israel voting system in Hebrew, http://blogs.haaretz.co.il/talschneider/502/

– **Other langauges:**

1. Žvalgybos naujienose – nepasitikėjimas amerikiečių standartais ir nevykęs vokiečių šnipas, http://www.alfa.lt/straipsnis/50218700/zvalgybos-naujienose-nepasitikejimas-amerikieciu-standartais-ir-nevykes-vokieciu-snipas.
2. Ameryka ma problem z zaufaniem wśród sojuszników. Dwa nowe standardy kryptograficzne proponowane przez amerykańskie NSA

budzą spore wątpliwości wśród ekspertów z całego świata. Chociaż nikt nie doszukał się w nich żadnych błędów, https://www.spidersweb.pl/2017/09/nsa-simon-speck-szyfrowanie-iso.html.

3. L'Organisation internationale de normalisation (ISO) retarde l'adoption de « Simon and Speck », https://www.developpez.com/actu/161684/L-Organisation-internationale-de-normalisation-ISO-retarde-l-adoption-de-Simon-and-Speck-deux-familles-d-algorithmes-de-chiffrement-de-la-NSA/.
4. Spojenci USA obviňují NSA z manipulování šifrovacích standardů, http://www.otechnice.cz/spojenci-usa-obvinuji-nsa-z-manipulovani-sifrovacich-standardu/.
5. US-Allierte werfen NSA vor, Krypto-Standards manipuliert zu haben, http://de.engadget.com/2017/09/22/us-allierte-werfen-nsa-vor-krypto-standards-manipuliert-zu-habe/.
6. Einst begehrte NSA-Kryptoexperten sind international quasi geächtet, http://winfuture.de/news,99738.html.
7. Anche Galileo avrà la firma elettronica, http://www.media.inaf.it/2017/02/13/galileo-cifratura-tesla/.

List of Publications

– Journals:

1. Y. Liu, G. De Witte, A. Ranea, and T. Ashur, **Rotational-XOR Cryptanalysis of Reduced-round SPECK**, IACR Transactions on Symmetric Cryptology 2017(3), pp. 24–36, 2017.
2. T. Ashur, A. Bar On, and O. Dunkelman, **Cryptanalysis of GOST2**, IACR Transactions on Symmetric Cryptology 2017(1), pp. 203–214, 2017.
3. T. Ashur, and Y. Liu, **Rotational Cryptanalysis in the Presence of Constants**, IACR Transactions on Symmetric Cryptology 2016(1), pp. 57–70, 2016.

– International Conferences:

4. T. Ashur, J. Delvaux, S. Lee, P. Maene, E. Marin, S. Nikova, O. Reparaz, V. Rozic, D. Singelée, B. Yang, and B. Preneel, **A Privacy-Preserving Device Tracking System Using a Low-Power Wide-Area Network (LPWAN)**, In Cryptology and Network Security, 16th International Conference, CANS 2017, Lecture Notes in Computer Science, Springer-Verlag, 22 pages, 2017. (not included in this Thesis)
5. T. Ashur, O. Dunkelman, and A. Luykx, **Boosting Authenticated Encryption Robustness with Minimal Modifications**, In Advances in Cryptology — CRYPTO 2017 (III), Lecture Notes in Computer Science, J. Katz, and H. Shacham (eds.), Springer-Verlag, pp. 3–33, 2017.
6. T. Ashur, and V. Rijmen, **On Linear Hulls and Trails**, In Progress in Cryptology — INDOCRYPT 2016, Lecture Notes in Computer Science, Springer-Verlag, pp. 269–286, 2016.
7. T. Ashur, O. Dunkelman, and N. Talmon, **Breaching the Privacy of Israel's Paper Ballot Voting System**, In International Joint Conference on Electronic Voting 2016, Lecture Notes in Computer

Science, R. Krimmer, and M. Volkamer (eds.), Springer-Verlag, pp. 128–124, 2016. (not included in this Thesis)

8. T. Ashur, and B. Mennink, **Damaging, Simplifying, and Salvaging p-OMD**, In Information Security — 19th International Conference, ISC 2016, Lecture Notes in Computer Science 9866, M. Bishop, and A. C. Nascimento (eds.), Springer-Verlag, pp. 73–92, 2016.
9. Tomer Ashur, Orr Dunkelman, **On the Anonymity of Israel's General Elections** (Poster), 2013 ACM SIGSAC Conference on Computer and Communications Security, ACM CCS 2013, A. R. Sadeghi, V. D. Gligor, and M. Yung (eds.), ACM, pp. 1399–1402, 2013. (not included in this Thesis)
10. T. Ashur, and O. Dunkelman, **A Practical Related-Key Boomerang Attack for the Full MMB Block Cipher**, In Cryptology and Network Security, 12th International Conference, CANS 2013, Lecture Notes in Computer Science LNCS 8257, M. Abdalla, R. Dahab, and C. Nita-Rotaru (eds.), Springer-Verlag, pp. 281–290, 2013. (not included in this Thesis)
11. T. Ashur, and O. Dunkelman, **Linear Analysis of Reduced-Round CubeHash**, In Applied Cryptography and Network Security — 9th International Conference, ACNS 2011, Lecture Notes in Computer Science LNCS 6715, J. Lopez, and G. Tsudik (eds.), Springer-Verlag pp. 462–478, 2011. (not included in this Thesis)

– National Conferences:

12. A. Ranea, Y. Liu, and T. Ashur, **An Easy-to-Use Tool for Rotational-XOR Cryptanalysis of ARX Block Ciphers**, In Proceedings of the Romanian Academy, Series A 18(3), pp. 307-316, 2017.
13. T. Ashur, G. De Witte, and Y. Liu, **An Automated Tool for Rotational-XOR Cryptanalysis of ARX-based Primitives**, In Proceedings of the 38th Symposium on Information Theory in the Benelux, Werkgemeenschap voor Informatie- en Communicatietheorie, pp. 59–66, 2017.
14. T. Ashur, and D. Bodden, **Linear Cryptanalysis of Reduced-Round Speck**, In Proceedings of the 37th Symposium on Information Theory in the Benelux, Werkgemeenschap voor Informatie- en Communicatietheorie, SiTB 2016, F. Glineur, and J. Louveaux (eds.), pp. 183–190, 2016

– Technical Reports

15. I. Fernández, V. Rijmen, T. Ashur, P. Walker, G. Seco, J. Simón, C. Sarto, D. Burkey and O. Pozzobon, **Galileo Navigation Message Authentication Specification for Signal-In-Space Testing - v1.0**, European Commission, 2016. (not included in this thesis)
16. T. Ashur, and V. Rijmen, **R&D Support on Galileo Open Service Navigation Message Authentication**, Technical report, KU Leuven, 02/2017, 35 pages.

– Unpublished Manuscripts and Preprints

17. T. Ashur, D. Bodden, O. Dunkelman, **Linear Cryptanalysis Using Low-bias Linear Approximations**, IACR Cryptology ePrint Archive 2017: 2017/204.
18. T. Ashur, **Improved Linear Trails for the Block Cipher Simon**, IACR Cryptology ePrint Archive 2015: 2015/285. (not included in this Thesis)
19. T. Ashur, **Higher-order Linear Cryptanalysis with Implications for Simon**, Unpublished Manuscript. (not included in this Thesis)

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF ELECTRICAL ENGINEERING
ESAT
Kasteelpark Arenberg 10 – bus 2452
B-3001 Leuven
Tomer.Ashur@esat.kuleuven.com
https://www.esat.kuleuven.be/cosic/?page_id=750