On the Existence of Non-Linear Invariants and Algebraic Polynomial Constructive Approach to Backdoors in Block Ciphers

Nicolas T. Courtois

University College London, Gower Street, London, UK

Abstract. In this paper we study cryptanalysis with non-linear polynomials cf. Eurocrypt'95 (adapted to Feistel ciphers at Crypto 2004). Previously researchers had serious difficulties in making such attacks work. Even though this is less general than a general space partitioning attack (FSE'97), a polynomial algebraic approach has enormous advantages. Properties are more intelligible and algebraic computational methods can be applied in order to discover or construct the suitable properties. In this paper we show **how** round invariants can be found for more or less any block cipher, by solving a certain surprisingly simple single algebraic equation (or two). Then if our equation has solutions, which is far from being obvious, it will guarantee that some polynomial invariant will work for an arbitrarily large number of encryption rounds. This paper is a proof of concept showing that it IS possible, for at least one specific quite complex real-life cipher to **construct in a systematic way**, a non-linear component and a variety of non-linear polynomial invariants holding with probability 1 for any number of rounds and any key/IV. Thus we are able to weaken a block cipher in a permanent and pervasive way. An example of a layered attack with two stages is also shown. Moreover we show that sometimes our equation reduces to zero, and this leads to yet stronger invariants, which work for **any** Boolean function including the original historical one used in 1970-1990.

Key Words: block ciphers, Boolean functions, Algebraic Normal Form, unbalanced Feistel ciphers, weak key attacks, backdoors, history of cryptography, T-310, Linear Cryptanalysis, Generalized Linear Cryptanalysis, partitioning cryptanalysis, polynomial invariants, triangular systems of equations, I/O sums, multivariate polynomials, symmetric polynomials, algebraic cryptanalysis.

Acknowledgments. Some of results contained in this paper were found by UCL students doing various projects on cryptanalysis. Students who made valuable contributions are, Marios Georgiou, Matteo Scarlata, Aidan Patrick, Anthony Romm, Kristelle Feghali, Julien Nahum, Vlad Popa, Raymond Tan, Wayne Tsui, Brian Gunawan, Romain Dumon, Charles Albert Desbaux.

1 Introduction

The concept of cryptanalysis with non-linear polynomials sometimes called Generalized Linear Cryptanalysis (GLC) was introduced by Harpes, Kramer, and Massey at Eurocrypt'95, cf [34]. It can also be described in terms of I/O sums or I/O relations (algebraic equations relating Inputs and Output variables), cf. [19]. Constructing such properties is in general a difficult combinatorial problem, and many researchers have in the past failed to find any such properties, cf. for example Knudsen and Robshaw at Eurocrypt'96 cf. [39] and there are extremely very few positive results on this topic except very recently [49]. A later paper Crypto 2004 provides a precious insight into this problem: the idea is that non-linear polynomial I/O sums (or I/O relations) can be eventually be constructed if we consider that the choice of polynomials in such relations will depend strongly on the internal wiring (connections) of the cipher. Concepts of Bi-Linear and Multi-Linear cryptanalysis were subsequently introduced [15, 16, 19] in order to work with Feistel ciphers with two and several branches specifically. In this paper we revisit the question of non-linear cryptanalysis and give it a fresh start. Can we construct a more substantial variety of round invariant properties, with diverse polynomials of degree 2,3,4 etc.., and working in a more natural real-life block cipher setting? This rather than building some very peculiar new ciphers [16, 17, 14, 3]?

1.1 Combinatorial or Algebraic

A major problem in cryptanalysis is discovery of invariant of semi-invariant properties of complex type. Some heuristics assumptions or some simplifications are needed as the space of solutions is very large and systematic exploration is not quite possible. There are two major approaches to our problem: combinatorial and algebraic. A combinatorial approach would be to try to modify the connections of the cipher so that to obtain a desired property, where the S-boxes or the Boolean functions would be given (for example modify the internal permutation inside DES). This leads to combinatorial problems which could be studied with formal coding and SAT solvers. An algebraic approach would be to consider the structure of the cipher fixed, even if very complex, and try to determine the solutions by solving a system of algebraic equations. Also a well known very general combinatorial approach considers arbitrary subsets of linear vectors spaces, and it is called partitioning cryptanalysis, cf. [3, 35, 36]. A more algebraic approach is to consider only specific forms of partitions, mainly those defined by the value (0 or 1) of a single Boolean polynomial. This is of course less general, BUT it leads to a more **effective** approach to the problem, effective in the sense that we expect that properties are described, discovered and studied with the tools of algebra, in particular many new very interesting questions can be asked, and we expect properties to be computed or derived rather than to happen by some incredible coincidence. In other terms we expect the algebraic approach to be more illuminating about what actually happens and also easier to study. More importantly in this paper we put forward an **algebraic approach of a new sort**. We show how the problem can be coded with a surprisingly simple single equation of a limited degree which we will call FE. Solving such equation(s) **guarantees** that we obtain a Boolean function and the polynomial invariant $\mathcal{P}$ which makes a block cipher weak. Moreover when the equation FE is simpler (fewer terms), the invariant is likely become stronger in the sense of holding for a larger space of Boolean functions, cf. Appendix A. Solving FE also avoids exploring the vast double-exponential space of possible Boolean functions (this again would be a combinatorial approach). Specific examples will be constructed based on a historical Cold War block cipher which has a highly complex irregular internal structure. With this cipher we will demonstrate that the set of solutions is sometimes not empty, and therefore our attack actually works.

1.2 Black Box vs. White Box

Research on block ciphers so far was about rather simple attacks (the number of solutions $< 2^n$) and in a **black box model**: approximations for N rounds ignoring what makes that they happen, or may work for a larger number of rounds. We work in **a white-box** model emphasizing that properties are mathematical facts computed explicitly, depending on specific pre-conditions. There is no need to study N rounds, if there is an invariant for 1 or 2 rounds, the game is over.

Our approach continues the research on non-linear cryptanalysis of block ciphers [34, 39, 15, 16] with a specific twist: we allow the attacker to manipulate the Boolean function Z. There is abundant literature about how a highly-nonlinear Boolean function affects the security of block ciphers. Most of these works are now obsolete because the community have ignored an extremely vast class of attacks which do affect the choice of the Boolean function and we are the first to show that numerous[1] truly pathological choices (for Z) may exist.

1.3 Related Work: Non-Linear vs. Partitioning Cryptanalysis

There have been numerous works on non-linear cryptanalysis and Generalized Linear Cryptanalysis (GLC) [34, 39, 15, 16, 38, 49] and then Bi-Linear and Multi-Linear cryptanalysis [15, 16, 19] for Feistel ciphers. There have also been many constructions of weak ciphers in cryptographic literature, cf. for example work related to the AES S-box [16, 17], very recent work by Filiol *et al.* [3, 9], work on choice of constants in modern hash functions [1, 41]. Almost all research in this area revolves around the fundamental notion of **partitioning**[2] **cryptanalysis** cf. [3, 35, 36] and Harpes will explain that partitioning cryptanalysis (PC) generalizes both LC and GLC [33, 35–37]. All these works are closely related and also to the study of the groups generated by various cipher transformations and the question of primitive/imprimitive groups [45, 46, 43, 47, 2, 16]. A **serious theory** is nowadays being developed around what is possible or not to achieve in partitioning attacks, with new important notions such as strongly proper transformations, anti-invariant properties, hidden sums etc, cf. [14, 9, 4].

[1] In fact a more recent paper shows that probably no Boolean function whatsoever can defend against polynomial invariant attack at a certain degree, see [26].

[2] In fact what we do in this paper is ALSO partitioning cryptanalysis, except that our partitions are characterized by a [single] multivariate polynomial.

In this paper we point out that the partitioning approach is simultaneously **too general and too obscure**. It is good for establishing numerous impossibility results cf. [9, 4], but potentially it will obscure any **possibility results**. We can discover some invariant, properties but do we understand their nature? Can we manipulate the properties efficiently and compress them (represent them in a compact way)? Can we discover properties with some effective computational methods and see how various constraints will imply their existence or not? Can we show that some ciphers are going to be secure against such attacks? Can we add and remove some complex invariants in block ciphers in a modular on-demand approach (keeping only more complex ones)? Can we construct weak keys secure w.r.t. simpler invariants and all previously known attacks? Yes we can, and our polynomial algebraic approach is what makes all these possible.

1.4 Low Degree Polynomials and the Cipher Structure

The idea to privilege a low degree multi-linear approach and more importantly that one needs to adapt to the high-level structure (wiring) of the block cipher was explicitly suggested in [15]. However none of the previous work has focused on explicitly constructing weak non-linear components in order to obtain invariants holding with probability 1. Also none of the previous works we are aware of, constructs invariant properties by solving a system of algebraic equations. Also in no cipher weakening research we are aware of, we can obtain a whole range of non-trivial solutions with increasing complexity, apparently without a limit.

1.5 On Mathematical Theory of Invariants.

There exists an extensive theory of multivariate polynomial algebraic invariants w.r.t linear transformations going back to 1845 [10–12]. This classical 19-th century invariant theory however deals with invariants in the situations where (simultaneously):

1) invariants are polynomials of small degree,
2) they have only 2 sometimes up to 5 variables,
3) polynomials are over large fields and rings, frequently algebraically closed or infinite (or both), or in fields with large characteristic,
4) invariants should not change when we operate a **LINEAR** input variable transformation L, a very important limitation,

3+4 makes that there is a scaling scalar or factor σ in most invariants known in classical mathematics: a determinant of the linear transformation L,

5) these invariants are in general multivariate polynomials.

In modern invariant theory, however, there are of course more possibilities [12], and here is what the first author of this book has put in a preface of his another (unrelated) book, cf. slide 28 in [28]:

> [...] Everybody in mathematics knows that going from one to several variables is an important jump that is accompanied by great difficulties and calls for completely new methods [...]

In general however we are quite far from the traditional preoccupations of mathematicians. The common points are:

1) we study polynomial invariants $\mathcal{P}$ of limited degree and
5) our invariants are multivariate polynomials over some fields,

However there are **very substantial** differences:

2') we work with many more variables for $\mathcal{P}$, typically between 8 and 36 at a time.
3') we work on $GF(2)$ mainly,
4') and finally we are looking for invariants which remain the same after applying an extremely complex **NON-LINEAR** transformation called ϕ, or any power of it ϕ^k, which are no longer linear cf. point 4) above, very peculiar, and not of the sort the mathematicians would consider worth studying,
3+4 Here the scaling factor δ could only be equal to 1 and should be omitted.

1.6 Round Invariants for Block Ciphers

A major risk in mathematics is that mathematical theories operate mainly at a syntaxic level and they could be dealing essentially with an empty set. This unless the objective is to prove the security by proving that the set is empty, see [9, 4] and later Section 5.6. Current research in application of polynomial invariants in symmetric cryptography has lacked substance or material to work in the form of real-life positive examples we can work on. Main results are about cipher components rather than full ciphers. For example for the AES-like S-box, it is possible to us the so called cross-ratio, which is already an invariant in the more general non-linear ϕ case which is more rarely studied in mathematics. However this type of invariant is however still quite simple or we work with only one variable. In our research we study a substantially wider variety of multivariate invariants with increasing size and complexity.

In cryptographic invariants the main object to study are **round invariants**. They can be defined as polynomials $\mathcal{P}$ the value of which does not change after we apply a transformation called "a round" we call ϕ. This round function ϕ is typically a bijection and is like one round of encryption. In addition, typically it is NOT one fixed permutation but it has a parameter, a secret key and potentially additional parameters. The more parameters, the harder it becomes to find invariants. For example the T-310 cipher can be viewed as each round is applying one of the 8 possible permutations $\phi_0 : \{0,1\}^{36} \to \{0,1\}^{36}$ up to $\phi_7 : \{0,1\}^{36} \to \{0,1\}^{36}$ and the choice which ϕ is actually used depends on 2 bits of the secret key and 1 bit of the IV (which is public and known to the attacker), all using the original notations of [45]. Technically speaking, finding such invariants is easy and they exist in vast numbers and many are in some sense trivial or degenerated, cf. Appendix A.2. A key problem and basically the main problem in this paper is to find **simultaneous invariants** to hold in all the eight cases, i.e. for all of $\phi_0 \dots \phi_7$ simultaneously, cf. later Sections 5.5 and 7, and such invariants should occur without being a consequence of simpler linear invariants, cf. Appendix A.2.

1.7 Group Theory vs. Invariants for Block Ciphers

We recall that for the finite field "inverse" S-box, and NOT for the actual AES S-box, cf. [16], it is possible to use the so called cross-ratio. Again this is already an invariant in our more general non-linear ϕ case. We refer to Section 4 in [16] for more details and further references. This cross ratio invariant is NOT always correctly applied. In fact there are singularities in polynomial invariants, and the cross-ratio invariant is simply not always true, sometimes it "breaks", or there is a discontinuity cf. [16], this correspond to invariants being correct probability of type $1 - \varepsilon$ with some small ε. This fact plays an important role inside the so called "Whitening Paradox" [16, 17] which is about security (and insecurity) of block ciphers and it is highly relevant here. The essence of why we have a paradox here can be briefly summarized as follows. The "Whitening Paradox" paradox is a proof of concept that a group-theoretic claims in cryptography [45–47, 2] can be highly misleading and can lead to a ciphers where the group of transformations generated by the cipher is proven mathematically to be extremely large, and which are nevertheless insecure and can simultaneously broken for an exponentially large number of rounds. This even though it is clear that when the number of rounds grows further, they are ultimately secure, cf. [16, 17]. In other words we get a cipher which is simultaneously provably secure and practically insecure.

There exists numerous modern works on the group of transformations generated by a block cipher [46, 43, 47, 2, 16] and primitive groups, [46, 43, 2, 14]. This research topic was very clearly was invented during the Cold War and was already studied very carefully in the 1970s with very specific security claims which are contained in [45], however these claims are not formulated as precise mathematical theorems in [45] and are therefore subject to interpretation.

1.8 On the Chicken and Egg Problem and Entry Barriers

On the surface this paper is about very specific attacks in very specific cases on a very specific cipher. In reality it makes an important and essential contribution to cryptanalysis of symmetric ciphers at large. For some four decades the block cipher cryptanalysis has suffered from a bootstrapping problem which can be summarized as follows:

1. Researchers have hardly found any invariant attacks ever, except very simple ones such as linear cryptanalysis, where the set of all possible attacks or characteristics is not too large (like small HW only, 64 bits).
2. Un incredibly large and rich space of attacks have been ignored due to the double exponential growth in the number of possible attacks. In general, the number of polynomial invariants is double exponential in the block size, 2^{2^n}.
3. Probabilistic invariants for many round were studied, stronger invariants true with probability 1 for just one round were neglected or ignored.
4. A black box round approximation approach on full cipher rounds was privileged. In this paper we adopt a more powerful white box approach where properties come to life, depend on very specific algebraic conditions, can be transposed from one place to another, sometimes added or removed at will.

5. Academic ciphers with a small number of rounds were studied, and few researchers dared to attack any cipher with a substantially larger number of rounds, fearing that this would be impossible.
6. As a consequence, researchers were stuck in a vicious circle. Researchers could not find new attacks on block ciphers and they failed to see or imagine how new or more complex attacks could even look like.
7. When they can imagine new attacks, there is a certain "gap" or entry barrier to overcome, for each new attack, initially the researchers will find nothing and will get discouraged.
8. This "gap" was exacerbated by the fact that some symmetric crypto research was classified during the Cold War. Therefore the researchers failed to see what was already known while many ciphers have been already enhanced to be secure against simpler attacks. This basically increased the gap for finding new attacks on some ciphers such as 3DES which have been designed carefully and behind closed doors, and further discouraged any relevant crypto research.

Positive examples of working attacks (cf. this paper) are essential because they can be imitated and they reduce the number of possibilities for attacking other ciphers. The gap between what is already known and new attacks shrinks. They also teach us that most of what we know or believe about security of block ciphers are specifically fallacies: things which do not imply or guarantee security, and we can demonstrate this by numerous examples.

The approach in this paper to show that one invariant property suggests or implies the existence of another more complex property for another cipher. We learn from examples. However we should also think that MANY MORE countless attacks directly exist in block ciphers, more complex, more sporadic, harder to find.

2 Notation and Methodology

In this paper we are going to work with one specific block cipher, in order to show what is possible or not. We are not however going to provide a full description of an encryption system and how it is initialized and used. We just concentrate on how one block cipher round operates and how it translates into relatively simple Boolean polynomials and eventually such a cipher could be strong or weak w.r.t. our attack. We will construct non-linear invariant properties which very substantially reduce the space of permutations which can be obtained by iterating our cipher. We will limit the number of variables used here to for example 20 out of 36. Our work is therefore very closely related to the so called partitioning attacks [3, 35, 9]. However we obtain and show the existence of specific polynomial invariants, rather than some obscure partition of a subspace the nature of which could be hard to apprehend.

Quite importantly, we are going to consider, which is very rarely done in symmetric cryptanalysis research, that the Boolean function is an unknown, yet to be determined. We will denote this function by a special variable Z. We will then postulate that Z may satisfy a certain algebraic equation [with additional variables] and then this equation will be solved for Z.

In order to have notations, which are as compact as possible, in this paper the sign + will denote addition modulo 2, frequently we will omit the sign * in products and will frequently use short or single letter variable names. In general in this paper we will use small letters $a - z$, and x_1, x_{36} or e_1 for various binary variables $\in \{0, 1\}$. Capital letters such as $S1, S2, L, F, Z$ will be used to represent some very "special" sorts of variables which are placeholders for something more complex. In particular the capital letter Z is a placeholder for substitution of the following kind

$$Z(e_1, e_2, e_3, e_4, e_5, e_6)$$

where $e_1 \ldots e_6$ will be some 6 of the other variables. In practice, the e_i will represent a specific subset of variables of type a-z, or other such as L, therefore at the end, our substitution will actually look like:

$$Z \leftarrow Z00 + Z01 * L + Z02 * j + Z03 * Lj + \ldots + Z62 * jhfpd + Z63 * Ljhfpd$$

Other capital letters will be used to signify some bits which are also unknown which will be bits of the secret key used in a given round and such bits are in our work called by letters $S1, S2$, where S2 will be sometimes renamed L. We also use capital letters to represent some bits which depend on the IV, or a round-dependent variable constant. This sort of variable which is typically known to the attacker will be denoted by the letter F in this paper. In general we are going the omit to specify in which round of encryption these bits are taken, as most of our work is about constructing one round invariants (which however do extend to an arbitrarily large numbers of rounds). We consider in general that each round of encryption will be identical except that they can differ only in some "public" bits called F (and known to the attacker) and some "secret" bits called $S1$ or L and unknown to the attacker. This framework covers most block ciphers ever made and ever studied in human history.

2.1 Our Specific Cipher

Too many block ciphers which appear in cryptographic literature are never used in any real-life situation. In this paper we are going however to work primarily on a particularly complex Feistel cipher on 4 branches known as T-310, which is how our proof of concept is going to be constructed. A nice web page which explains how this block cipher works can be found at:
`http://blog.bettercrypto.com/?page_id=3864`, see also Fig. 1 page 12 and Fig. 2 page 13. From our perspective all the matters, and this works for any block cipher, is that each round can be described by a bunch of polynomials, see later Section 3.5.

The choice of T-310 implies that in our proof of concept, we are going to search for Boolean functions on 6 bits which lead to specific invariant attacks. There are $2^{2^6} = 2^{64}$ Boolean functions on 6 bits. The block size is be 36 bits. There are potentially approximately $2^{2^{36}}$ possible non-linear polynomial invariants for 1 round of our cipher with 36-bit blocks.

2.2 The Challenge

One of the biggest open problems in symmetric cryptography, is to find new classes of invariant properties which hold for specific types of block ciphers. For example, could that be possible that for a complex block cipher with no weakness w.r.t. simpler attacks such as Linear Cryptanalysis, we have that the following irreducible polynomial with 144 products the value of which is preserved after an arbitrary number of encryption rounds?

```
abehij+abcehij+abdehij+abcdehij+abefhij+abcefhij+abdefhij+abcdefhij+abeghij+abceghij+abdeghij+abcdeghij+bcefjk+abcefjk+bcdefjk+
abcdefjk+bcefgjk+abcefgjk+bcdefgjk+abcdefgjk+bcefhjk+abcefhjk+bcdefhjk+abcdefhjk+bcefijk+abcefijk+bcdefijk+abcdefijk+bcefgijk+
abcefgijk+bcdefgijk+abcdefgijk+abehijk+abcehijk+abdehijk+abcdehijk+abefhijk+bcefhijk+abdefhijk+bcdefhijk+abeghijk+abceghijk+
abdeghijk+abcdeghijk+adghil+abdghil+acdghil+abcdghil+adeghil+abdeghil+acdeghil+abcdeghil+adfghil+abdfghil+acdfghil+abcdfghil+
abehijl+abcehijl+abdehijl+abcdehijl+abefhijl+abcefhijl+abdefhijl+abcdefhijl+adghijl+abdghijl+acdghijl+abcdghijl+abeghijl+abceghijl+
adeghijl+acdeghijl+adfghijl+abdfghijl+acdfghijl+abcdfghijl+cdefgkl+acdefgkl+bcdefgkl+abcdefgkl+cdfghkl+acdfghkl+bcdfghkl+abcdfghkl+
cdefgikl+acdefgikl+bcdefgikl+abcdefgikl+adghikl+abdghikl+acdghikl+abcdghikl+adeghikl+abdeghikl+acdeghikl+abcdeghikl+adfghikl+
abdfghikl+cdfghikl+bcdfghikl+bcefjkl+abcefjkl+bcdefjkl+abcdefjkl+bcefgjkl+abcefgjkl+cdefgjkl+acdefgjkl+bcefhjkl+abcefhjkl+bcdefhjkl+abc
```

In this paper we make this happen, on demand, given an arbitrary specification of the internal wiring of the cipher. Our method is constructive-algebraic and works potentially for all known block ciphers and for Boolean functions and S-boxes of arbitrary size.

2.3 Polynomial Invariants

In general our methodology applies to any block cipher and we are looking for arbitrary polynomial invariants coded as formal polynomials. In particular however, the structure of the polynomial is provided or guessed by the attacker depending on "hints" or "heuristics" which are based on the high-level structure of the cipher. This is what is suggested from known examples of polynomial cryptanalysis [15, 16]. In particular for Feistel ciphers we will see a lot specific[3]

[3] Which can also be seen as unions of cycles, cf. for example Fig. 7.

types of symmetric[4] polynomials on subsets of variables. For example we may consider all polynomials of the form:

$$\mathcal{P}(a, b, \ldots) = P00 + P51 * (e + f + g + h) + P42 * (abc + abd + acd + bcd) + \ldots$$

this rather than completely general polynomials which would be too costly to consider and process efficiently. The goal is of course to avoid excessive generality and reduce the number of variables such as P51 which are determined at a later "algebraic solving" stage of the attack.

2.4 Invariants for More Than One Round

Almost all invariants in this paper are invariants for 1 round of encryption. Invariants for more rounds which as such do not work for 1 round (but are likely to imply the existence of other invariants for 1 round) are very common in Linear Cryptanalysis, see for example Section 21.16 in [21]. This also happens here, for example in Section 10.6 and in Appendix B.2.

[4] It is however important to see that some of our polynomials will be very far from being symmetric, for example in Section 5.1 we have $\mathcal{P}$ such that $\mathcal{P} - fg$ is a symmetric homogenous polynomial, and in Section 9.1 there we see no symmetries whatsoever and non-linear part is $eg + fh + fl$ and in Sections 9.2 and 9.4 we have only 1 or 2 terms of degree 2 which is hardly a symmetric situation.

3 General Approach vs. One Specific Cipher

Most cryptanalytic attacks work, well, only in some cases. In this paper we try to strike a balance between a completely general constructive approach applicable to any block cipher, such as 3DES, AES, GOST, etc, and constructing simple examples dictated by the high-level structure of one specific Feistel cipher.

3.1 Polynomial Invariants vs. General Partitioning Attacks

In theory, our approach is totally general as every function could be written as a polynomial over a finite field, therefore in a very broad (and naive) sense Partitioning Cryptanalysis (PC) and Generalized Linear Cryptanalysis are equivalent. However our approach would not be practical if needed to study completely general polynomials with 36 variables In reality however, we will be working with sets characterized by one polynomial at a time, our ability to solve the equations is limited, and our approach will principally work when $\mathcal{P}$ is not too complex or when it is sparse and/or of reduced degree, and with specific very strong symmetries, cf. Section 2.3.

3.2 Constructive Approach Given the Cipher Wiring

We decided to execute our task on a cipher which offers great **flexibility** in the choice of the internal wiring, so that we can possibly make such adjustments if we do not find a property we are looking for. In theory most ciphers such as DES or AES do offer this level of flexibility (choice of P-boxes, arbitrary invertible matrices inside the S-box and inside the mixing layers, etc). Here we work the T-310 cipher from 1980s where such changes are officially allowed and are officially specified by the designers of the cipher. Here if we find a weak setup, it can be directly implemented with original historical hardware. Our work is at the antipodes compared to [16, 17] where the ciphers are really very special and have very strong hidden high-level structure (and then non-linear invariants propagating for 1 million rounds can be constructed). Our approach is really the opposite: we start from any given cipher spec in forms of ANFs for one round (possibly with some "chose permutation of wires" flexibility for connecting non-linear components) and we generate invariant properties on demand.

In general the attacker is allowed to manipulate the Boolean function, which variable has a large entropy (64 bits typically) and to a lesser extent also to somewhat but less[5] manipulate the cipher wiring, and also he is able to freely select an invariant polynomial $\mathcal{P}$ which choice however is restricted in several ways[6]. The approach we present is applicable to more or less[7] arbitrary block

[5] Only in the sense of 1 out of 1000: study of weaker and stronger cipher wirings.

[6] It will be restricted by 1) which kind of polynomial invariants maybe exit for this cipher, for example Feistel ciphers tend towards combinations of symmetric polynomials with subsets of variables, and 2) the computing power of the attacker and his ability to find invariants by solving increasingly complex systems of equations.

[7] This is if there is a sufficient amount of entropy inside these non-linear components, possibly not to Simon or TEA where S-boxes are too small or rather inexistent for us to manipulate.

ciphers which contain non-linear components. It is also applicable to hash functions and stream ciphers based on a core block cipher (a large permutation with few round-dependent bits which can be key bits, IV bits or message bits).

3.3 Representing One Encryption Round

Let us imagine that we are presented with an arbitrary complex block cipher designed by some very paranoid designers with large complexity and a sophisticated internal structure. For example the picture below shows the internal structure of T-310, one of the most important block ciphers of the Cold War, which was used at a massive scale in Eastern Europe to encrypt all sorts of state communications, cf. [48]. The cipher operates on 36-bit blocks and the state bits are numbered 1-36. Here is a glimpse of the internal structure:

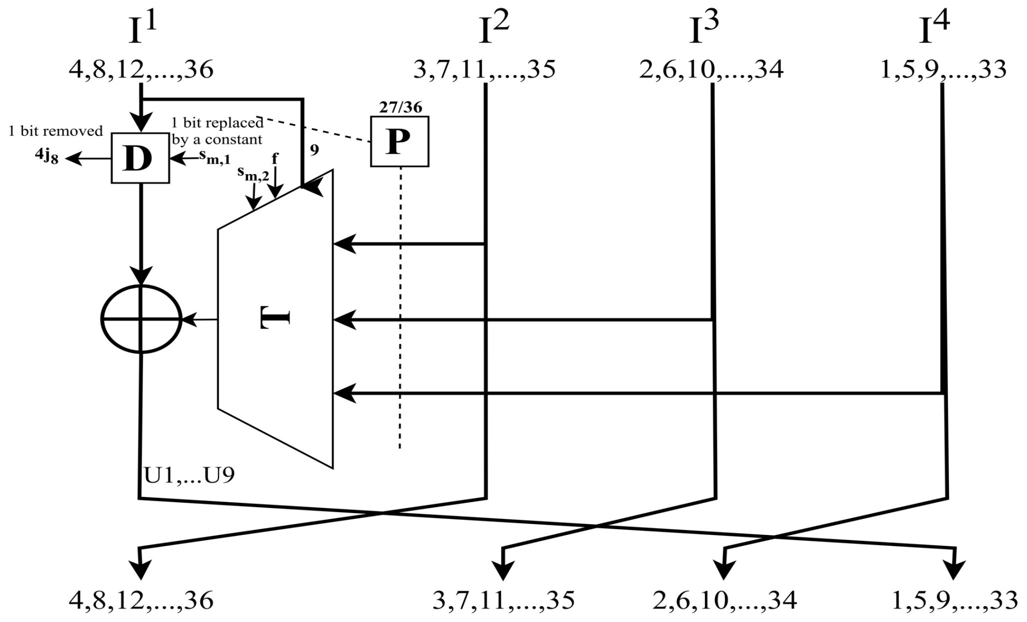

Fig. 1. T-310: a peculiar sort of a Compressing Unbalanced Feistel scheme cf. [42, 21].

T-310 happens to be one of the most "paranoid" cipher designs we have ever seen. The per-encrypted-bit hardware complexity of T-310 is absolutely staggering. It is hundreds of times more costly, even with modern software and hardware, than AES or triple DES, cf. [24, 25]. Also typically and almost always the bits actually used in encryption come from the right end, see [30, 21], which is more complex than the other end. However one round is **intelligible** and by no means secure. Therefore the designers have mandated that incredibly large number of 1651 such rounds must be executed in order encrypt just one character on 5 bits. Does it make this cipher very secure as intended for a serious government security cipher in the middle of the Cold War? Not quite, if we are able to show that algebraic invariants can be constructed which work **no matter**

how many rounds we apply, and not quite if such invariants exist which work without using all the state bits, and without using all the key bits either.

3.4 Internals of One Round

A lot remains unspecified on our picture: which bits and in which order are connected to D1-D9 and v1-v27. In T-310 this specification is called an LZS or *Langzeitschlüssel* which means a long-term key. It could be compared to the knowledge of rotors in Enigma or S-boxes in GOST.

We simply need to specify two functions $D : \{1 \dots 9\} \rightarrow \{0 \dots 36\}$, $P : \{1 \dots 27\} \rightarrow \{1 \dots 36\}$. We will assume both functions to be injective in order to avoid numerous degenerate cases. For example $D(5) = 36$ will mean that input bit 36 is connected to the wire D5 on our picture, and $P(1) = 25$ will mean that input 25 is connected as v1 or the 1st input of $Z1$.

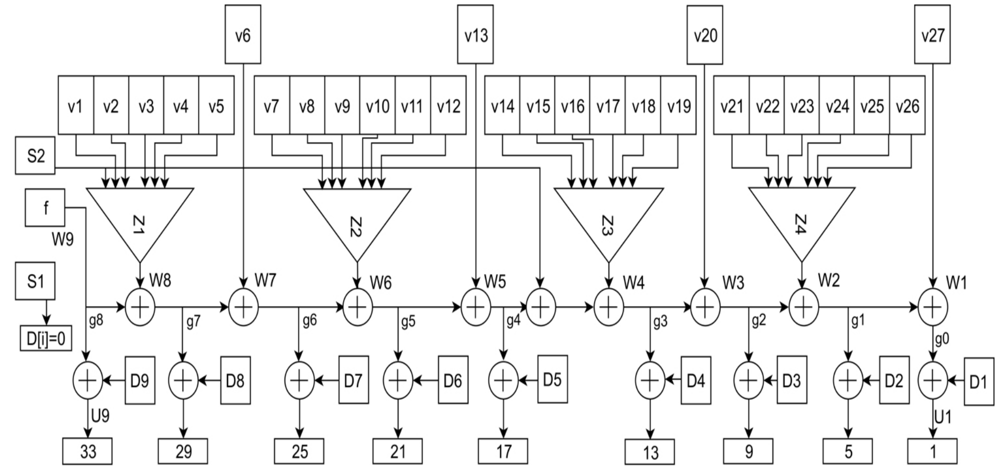

Fig. 2. Internal wiring of one round of T-310.

It remains to specify that each round, bits $1, 5, 9 \dots 33$ are those freshly created by this round, while ALL the input bits the numbers of which are NOT multiples of 4 are shifted by 1 position, i.e. bit 1 becomes 2 in the next round, and bit 35 becomes 36. This completes the specification of the internal connections of one round[8]. We will also note that F is a public bit derived from an IV transmitted in the cleartext, and S1 and S2 are bits of the secret key which has 240 bits. Finally the internal wiring LZS has a special convention where the bit

[8] In fact, it is possible to rewrite this cipher to make it look as a non-orthodox variant of an "Unbalanced Compressing Feistel" cipher with 4 branches, cf. [42, 21].

S1 is used as one of Di by specifying [9] that $D(i) = 0$. Most of the time being[10] we are going to assume that all the four Boolean functions Z1-Z4 are identical.

3.5 ANF Coding of One Full Round

Overall the cipher can be described as 36 Boolean polynomials out of which only 9 are non-trivial. Let $x_1, \ldots, x_{36}$ be the inputs and let $y_1, \ldots, y_{36}$ be the outputs.

$$
\begin{aligned}
y_{33} &= F + x_{D(9)} \\
&\quad Z1 \overset{def}{=} Z(S2, x_{P(1)}, \ldots, x_{P(5)})) \\
y_{29} &= F + Z1 + x_{D(8)} \\
y_{25} &= F + Z1 + x_{P(6)} + x_{D(7)} \\
&\quad Z2 \overset{def}{=} Z(x_{P(7)}, \ldots, x_{P(12)}) \\
y_{21} &= F + Z1 + x_{P(6)} + Z2 + \quad x_{D(6)} \\
y_{17} &= F + Z1 + x_{P(6)} + Z2 + \quad x_{P(13)} + x_{D(5)} \\
&\quad Z3 \overset{def}{=} Z(x_{P(14)}, \ldots, x_{P(19)}) \\
y_{13} &= F + Z1 + x_{P(6)} + Z2 + \quad x_{P(13)} + S2 + Z3 + x_{D(4)} \\
y_9 &= F + Z1 + x_{P(6)} + Z2 + \quad x_{P(13)} + S2 + Z3 + x_{P(20)} + x_{D(3)} \\
&\quad Z4 \overset{def}{=} Z(x_{P(21)}, \ldots, x_{P(26)}) \\
y_5 &= F + Z1 + x_{P(6)} + Z2 + \quad x_{P(13)} + S2 + Z3 + x_{P(20)} + Z4 + x_{D(2)} \\
y_1 &= F + Z1 + x_{P(6)} + Z2 + \quad x_{P(13)} + S2 + Z3 + x_{P(20)} + Z4 + x_{P(27)} + x_{D(1)} \\
&\quad x_0 \overset{def}{=} S1 \\
y_{i+1} &= x_i \text{ for all other } i \neq 4k \qquad (\text{ with } 1 \leq i \leq 36)
\end{aligned}
$$

We observe that T-310 has a remarkable "triangular" structure with increasing complexity and we aim to benefit from this with invariants using a subset of all cipher state bits only.

For better readability we reproduce our two figures on one page:

[9] Needless to say D and $\mathcal{P}$ are injective and have been specified and studied with serious amount of maths and analysis since the early 1970s [45]. Historical documents [45] show advanced combinatorial results about strength of the cipher after a few rounds, and contain explicit group-theoretic claims about the size of the group generated by the permutations of this cipher. A recent paper analyses the historical classes KT1 and KT2 of long-term keys from [45] and shows that they can be proven to be secure against a certain class of ciphertext-only attacks, see [22].

[10] This condition will probably be relaxed in our future work and offers yet much larger freedom for the attacker, with 256 bit of entropy to manipulate.

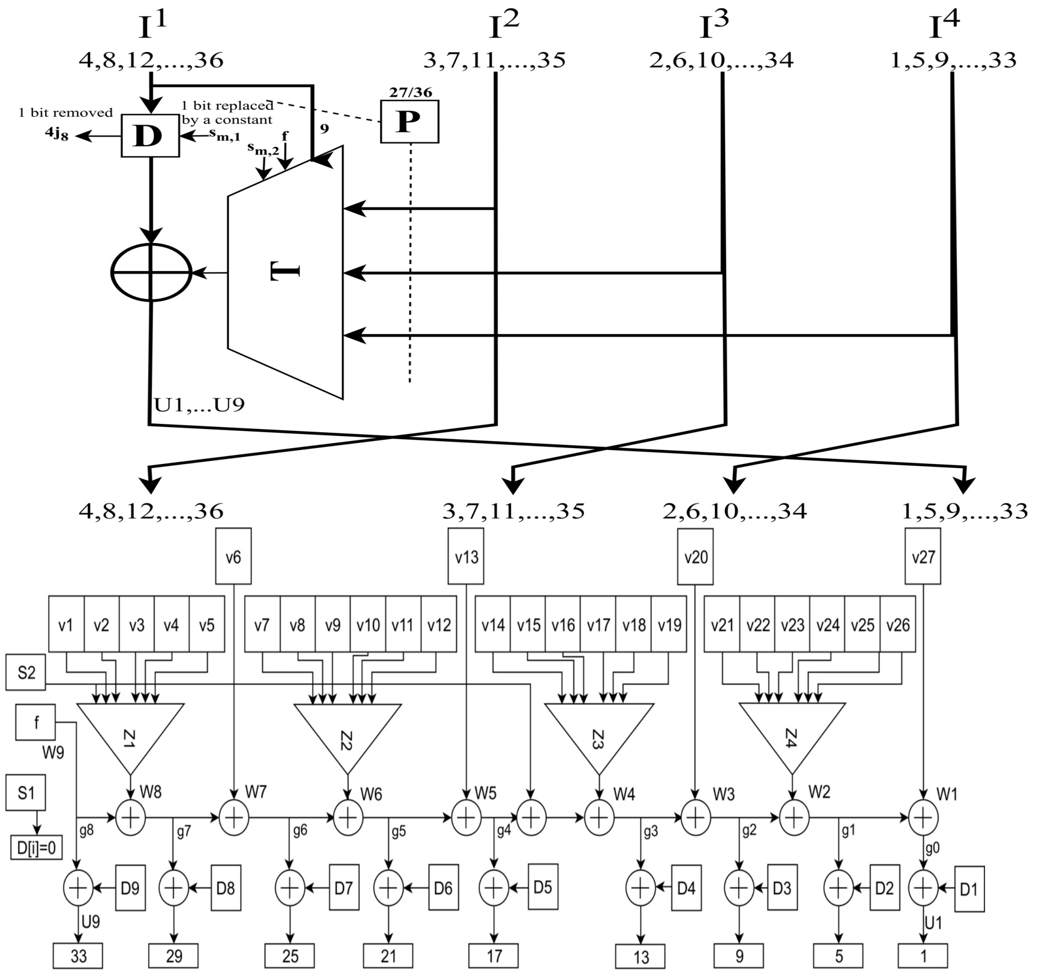

Fig. 3. One round of T-310 in a typical case. We have a round function ϕ which takes 36 bits x_i, $i = 1, \ldots, 36$ as an input, which go to one of the four branches depending on i mod 4, produces an output on 36 bits y_i and also has 3 extra bits as a parameter which are called $F, S1, S2$. The selection of 27 out of 36 bits which are used inside the round function box $T()$ in each round is specified by two functions $P(1) \ldots P(27)$ which defines the connection of $v1$ to $v27$ to block cipher state wires $1-36$ and $D(1) \ldots D(9)$ which typically in the standard KT1 type of wiring uses only 8 bits from the wires $1-36$ and if in one case we have $D(i) = 0$ this special value indicates that we will XOR with a key-dependent constant $S1$ equal to $s_{m,1}$ in round $m = 1, 2, \ldots$.

3.6 Variable Renaming

We recall that input of one round are denoted by $x_1, \ldots, x_{36}$. Now we are going to migrate towards more convenient and shorter notations. We will emphasise the fact that the variables x_i and y_i are treated "alike" and we will call them by lowercase letters a-z backwards starting from x_{36} till x_{11} and y_{36} till y_{11}. Thus for example $a = x_{36}$ and $t = x_{17}$. This will allow our higher degree polynomial expressions to be take less space. Then we are going to use letters M to V for further variables x_{10} till x_1, with $x_1 = V$ and $x_{10} = M$. This convention avoids a number of capital such as F, K, L, W, X, Y, Z which are used elsewhere. Due to the peculiar triangular structure of this cipher, variable $x_1 = V$ has the most complex dependency on the highest possible number of input bits.

Numbers	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
Letters	V	U	T	S	R	Q	P	O	N	M	z	y	x	w	v	u	t	s	r	q	p	o	n	m	l	k	j	i	h	g	f	e	d	c	b	a

Fig. 4. Variable naming conventions

The reader may be surprised to hear that in our cryptanalysis efforts we will be able to map x_i to y_i (they are denoted by the same letter) and this for ALL the 36 variables, and this after exactly 1 round. This even though for almost all variables $x_i = y_{i-1}$ after one round. This is **not impossible**, not even in Linear Cryptanalysis (LC). For example if $a \overset{def}{=} x_{36}$ in input expressions, and also used to denote y_{36} for output expressions $b \overset{def}{=} y_{35}$ and also used to denote y_{35}, it is unthinkable that say a is an invariant property because actually b becomes a in the next round. However in fact a polynomial such as $a+b+c+d$ absolutely CAN be an invariant expression for one round (!). This is, only IF our cipher had ANY invariant linear property for 1 round true with probability w.r.t the usual Linear Cryptanalysis (LC). This is possible but not so common. It does happen for this cipher, and a recent paper shows that linear invariants will occur for about 3% of so called LC-weak long term keys inside the so called KT1 specification of officially approved or "well-formed" long-term keys, cf. [24].

In our paper we will be of course looking at what happens for the remaining **stronger** 97 % of LZS keys and at more general **non-linear** polynomial expressions such as say $h + g + f + e + hgf + hge + hfe + gfe$. All examples we give in this paper are real-life examples which actually have been encountered in our research on cryptanalysis of this cipher. The polynomials we obtain are however not random, but rather they combinations of specific symmetric polynomials on subsets of variables, with additional terms which are not symmetric, and they do depend very strongly on the wiring of the cipher. For example this irreducible polynomial invariant with 43 monomials has been seen to work:

```
eg+fh+gi+hi+ej+hj+ij+ek+fk+jk+fl+il+jl+hm+km+lm+en+in+fo+io+jo+mo+
gp+jp+kp+np+hq+kq+lq+oq+er+ir+pr+fs+is+js+ms+qs+gt+jt+kt+nt+rt
```

3.7 The Result After Renaming

Overall, depending on the exact values of $D(i)$ and $P(j)$, we can rewrite the beginning of our equation system as follows, skipping the last few equations (our invariants will NOT use them!). The variables on the left hand side will be output variables after 1 round, and on the right hand side, we have ANF or polynomials in the input variables.

$$\begin{aligned}
a &\leftarrow b\\
b &\leftarrow c\\
c &\leftarrow d\\
d &\leftarrow F + i\\
e &\leftarrow f\\
f &\leftarrow g\\
g &\leftarrow h\\
h &\leftarrow F + Z1 + e\\
Z1 &\leftarrow Z(L, j, h, f, p, d))\\
i &\leftarrow j\\
j &\leftarrow k\\
k &\leftarrow l\\
l &\leftarrow F + Z1 + r + g\\
m &\leftarrow n\\
n &\leftarrow o\\
o &\leftarrow p\\
p &\leftarrow F + Z1 + r + Z2 + c\\
Z2 &\leftarrow Z(k, l, o, e, n, t)\\
q &\leftarrow r\\
r &\leftarrow s\\
s &\leftarrow t\\
t &\leftarrow F + Z1 + r + Z2 + m + s\\
u &\leftarrow v\\
v &\leftarrow w\\
w &\leftarrow x\\
x &\leftarrow F + Z1 + r + Z2 + m + L + Z3 + b\\
Z3 &\leftarrow Z(u, s, N, R, M, b)\\
y &\leftarrow z\\
&\ldots \text{[few lines missing]}
\end{aligned}$$

These expressions should be viewed as a sequence of substitutions where a variable is replaced by a polynomial algebraic expression.

3.8 Further Remarks and Observations

We will then aim at not using at all these variables in our invariants. Moreover in order to have shorter expressions to manipulate later on, we have replaced S2 by a single letter L. The other key bits $S1$ currently do not appear at all (the place where they would appear depends on $D(i)$). In general in our attacks we would represent key bits $S1$ by short[11] notation K. In practice will **avoid** using S1, as in real-life encryption it is almost always used at place D1, cf. [30, 21] and would appear in the very last equation $y_1 = F + \ldots + x_{D(1)}$ in Section 3.5 which we have omitted now. We also sometimes replace a single letter which comes from $P(6)$ by a variable G. The lower side is the side we will be simply avoiding (for the time being) in our invariant attacks. This not only due to the shortage of lowercase letters, but also because equations tend to be more complex than at the upper end(!). Using only a subset of variables and avoiding K, or eliminating K is what we need, and it is a great idea form the cryptanalysis point of view. In this way we will in fact obtain I/O properties with fewer monomials and such that half of the secret key of type S1 (up to 120 bits) are **eliminated** from the start(!).

3.9 On ANF Degree

One important point is that the algebraic degree of all our ANF expressions is constant. The ANF expressions have degree at most 6 and it does not increase for the lower parts (parts with K and bits 1 to 11 we may want to avoid). In contrast the number of monomials actually used increases (linearly).

3.10 Search for Invariants

Now the only thing which remains to be done is to find a polynomial expression $\mathcal{P}$ say

$$\mathcal{P}(a, b, c, d, e, f, g, h, \ldots) = abcdijkl + efg + efh + egh + fgh$$

using any number between 1 and 36 variables such that if we substitute in $\mathcal{P}$ all the variables by the substitutions defined above in Section 3.7, we would get exactly the same polynomial expression $\mathcal{P}$.

For simplicity and in practice in our proof of concept examples in this paper we will be using only 20 variables a-t. and we will avoid Z3 and Z4, constructing only invariants using Z1 and Z2 with round-dependent constants L and F only, cf. Section 4.12. More generally what we do, applies also to Z3 and Z4 and to any block cipher, however there will be more than 2 round-dependent parameters [key bits or round-dependent constants] and the success rate will be lower.

[11] This is used when we are looking at 1-round invariant properties and there is no ambiguity about in which round this variable K is used.

4 The Fundamental Equation

We want to **find** a polynomial expression $\mathcal{P}$ using any number between 1 and 36 variables such that it is an invariant after the substitutions of Section 3.7. In this paper the idea is that $\mathcal{P}$ could potentially be any non-linear polynomial of a certain degree with a specific well-chosen set of say 17 variables (specifically avoiding using too many variables from the lower parts of the cipher) and with specific internal symmetries due to the structure of the cipher. One simple method is to select a specific short symmetric polynomial which have already been seen to work for this cipher. More generally the polynomial is not always symmetric (cf. Section 5.1 or Section 9.1 and will also have variables which are not yet known. However not every polynomial works and combinations of symmetric polynomials are also frequently seen cf. Section 2.3.

Once the polynomial $\mathcal{P}$ is fixed, the attacker will write ONE SINGLE algebraic equation which he is going to solve to determine the unknown Boolean function Z, if it exists.

Definition 4.1 (Compact Uni/Quadri-variate FE). Our "Fundamental Equation (FE)" to solve is simply a substitution like:

$$\mathcal{P}(Inputs) = \mathcal{P}(Outputs)$$

or more precisely

$$\mathcal{P}(a, b, c, d, e, f, g, h, \ldots) = \mathcal{P}(b, c, d, F + i, f, g, h, F + Z1 + e, \ldots)$$

At this stage expressions of type Z1 or Z3 are placeholders for degree 6 polynomials yet to be specified fully.

The main unknown in FE is a Boolean function Z and in simple cases the FE can be of type $fZ = g$ where f and g are two polynomials[12]. More generally Z is represented by an Algebraic Normal Form (ANF) and 64 binary variables which are the coefficients of the ANF of Z, and there will be several equations, and several **instances** Z1-Z4 of the same Z:

Definition 4.2 (A Multivariate FE). Furthermore we will rewrite FE as follows. We will replace Z1 by:

$$Z1 \leftarrow Z00 + Z01 * L + Z02 * j + Z03 * Lj + \ldots + Z62 * jhfpd + Z63 * Ljhfpd$$

Likewise we will also replace:

$$Z2 \leftarrow Z00 + Z01 * k + Z02 * l + Z03 * kl + \ldots + Z62 * loent + Z63 * kloent$$

and the coefficients $Z00 \ldots Z63$ will be the same inside $Z2$, $Z3$ and $Z4$, however the sets of 6 variables chosen out of 36 will be different in Z1-Z4.

Note. Our compact notations omit the stars for products of small variables.

[12] Such equations have numerous non-trivial solutions, cf. [20].

4.3 On Choice of $\mathcal{P}$ in our FE

Initially, we can select $\mathcal{P}$ as an arbitrary fixed polynomial, with degree say between $d = 2$ and $d = 20$. For example one we have chosen ourselves(!) or one previously seen to work for that cipher, or one based on well-known symmetric polynomials with some unknowns, cf. Section 2.3. Then if we cannot find a solution, we will enlarge the space of solutions but making more or all coefficients of $\mathcal{P}$ variable. In all cases also, all we need to do is to solve the equation above for Z (plus a variable amount of extra variables). This formal algebraic approach, if it has a solution Z, will **guarantee** that our invariant $\mathcal{P}$ holds for 1 round.

4.4 Solving the Fundamental Equation

We recall our "Fundamental Equation (FE)".

$$\mathcal{P}(Inputs) = \mathcal{P}(Outputs)$$

or

$$\mathcal{P}(a, b, c, d, e, f, g, h, \ldots) = \mathcal{P}(b, c, d, F + i, f, g, h, F + Z1 + e, \ldots)$$

The process is as we can see EXTREMELY SIMPLE: we assume that a certain equation holds for Z and we solve it for Z which is 64 binary unknowns for the ANF coefficients. The solution is found easily[13] either by standard Gröbner bases techniques, or by conversion to SAT problem as described in [13]. Our experience seems to show that this problem will rarely be actually computationally really hard. This depends on many factors such as size, degree and shape of $\mathcal{P}$ etc.

4.5 Case when $\mathcal{P}$ is fixed

In fact our problem is in many interesting cases trivial to solve. For example it is easy to see that if $\mathcal{P}$ fixed, and if L, F are not used, (or if they are fixed, or for a reduced set of equations after elimination of all monomials containing L, F) the problem boils down to solving a system of LINEAR of equations with 64 binary variables which are the coefficients Zii which define the ANF of Z.

4.6 Simultaneous Solving

In a major variant of our problem, the polynomial $\mathcal{P}$ is no longer fixed, instead some or all of its coefficients are variables. Moreover it is in general advisable to already assume that $\mathcal{P}$ has specific symmetries which decreases the number of coefficients to be determined, cf. example in Section 2.3. Then, ignoring special bits L, K, F etc, the equations are no longer linear, but rather bi-linear, with products of coefficients of $\mathcal{P}$ and of Z. More generally, the problem is multi-linear. Again, in practice this problem is solved by a SAT solver typically or by Gröbner basis algorithms.

[13] All we have to do is to, consider that equality above must hold for each coefficient of each possible monomial. Thus we can rewrite the equation above as a system of simultaneous non-linear equations which will be small degree sparse polynomials in the 64 unknown coefficients Zii which form the ANF of Z, $ii = 0, \ldots, 63$.

4.7 The Solvability Problem

A major problem is rather, **the existence of solutions**. Does this equation FE have a solution? Or, does it have solutions which would satisfy some additional requirements required to design any sort of meaningful cryptanalytic attack? This is going to be a major question to study.

There are, as the reader may guess, countless cases where this problem has **no solution** whatsoever. Our experience shows that in many cases contradictions in our equations are found very easily[14] without even examining all the parts of the equations. This means that the fact that the space for Z is very large does not necessarily help to make our system of equations solvable. Frequently we stop earlier.

There are also numerous cases where the equation becomes unusually simple and has low degree, or it disappears totally (it reduces to zero once $\mathcal{P}$ is fixed, see Appendix A). In such cases the space for the solutions Z is typically quite large, which increases the applicability of the attack to real-life cases, for example when the attacker is not quite able to choose the Boolean function, but he might be able to manipulate the cipher wiring and increase the degree of $\mathcal{P}$.

4.8 More General 5-Linear Fundamental Equation

In general we write our Fundamental Equation (FE):

$$\mathcal{P}(a, b, c, d, e, f, g, h, \ldots) = \mathcal{P}(b, c, d, F + i, f, g, h, F + Z1 + e, \ldots)$$

as sum of products where we can distinguish 5 distinct types of variables or terms:

1. Terms of type $S1, S2$ or L which are key bits unknown to the attacker.
2. Terms of type F which are IV or round-dependent constant bits known to the attacker.
3. Terms of type Zii which are coefficients inside Z.
4. Terms of type $\mathcal{P}jj$ which are coefficients of $\mathcal{P}$ if it is not fixed.
5. Terms of type $abcdklm$ which are monomials in internal I/O variables of the cipher.

Again the equation is expected to hold for all values for the small 1-letter variables abcdef and all the variables which start with capital letters are place-holders for things which play a particular role in our solving process. In general, depending on circumstances, we fix some of these terms and we determine the other by solving the Fundamental Equation. For example if $\mathcal{P}$ is known and fixed, the equation becomes 4-Linear in the sense above. Or if Z is known and fixed, the equation is again 4-Linear and can be used to determine $\mathcal{P}$. Again in special cases this equation will be linear and in most other cases, it will be nevertheless possible to solve in practice.

[14] For example if our Fundamental Equation after our substitutions has the property that if Z divides one monomial, Za also divides this monomial, and if there is a single non-zero monomial without a, then the system of equations has no solution. Thus the fact that space for Z is quite large does NOT guarantee that the set of solutions we are looking for is not empty.

4.9 Extended 5-Linear Fundamental Equation

More generally we can extend our Fundamental Equation (FE) to:

$$\mathcal{P}(a, b, c, d, e, f, \ldots) = \mathcal{P}(b, \ldots, F + Z1 + r + Z2 + m + s, \ldots) + Q(F, L)$$

where Q is an another polynomial which depends on key and IV bits. In short notation we call this type of 5-Linear Extended Fundamental Equation by the name 5EFE.

4.10 Even More General Iterative 6-Linear Fundamental Equation

In general, the approach can be **iterative**. The attacker may, for example due to a previous iteration and solving FE once, already know some specific polynomial invariant for both the input and the output of each round. We call I this quantity and I is a complex polynomial in all small-letter variables abcd.. however the value of I is known, the other values aren't.

Our Extended 6-Linear Fundamental Equation (FE) will be:

$$\mathcal{P}(a, b, c, d, e, f, \ldots) = \mathcal{P}(b, \ldots, F + Z1 + r + g, \ldots) + Q(F, L, I)$$

where we have added one extra term $Q()$ and additional inputs I:

6. Terms of type I are bits of the internal state known to the attacker.

We have seen countless examples when this happens, and there are also many keys with linear invariants which can be used here as I in the same way. A nice real-life example is given in Section 7.2.

4.11 Iterative Solving

This section should be treated as optional and is not central in this paper. The approach presented above can be iterated in order to generate further invariants. Potentially in some cases the number of equations generated will grow by iterating this approach, until the cipher is completely broken, in the same way as the so called ElimLin algorithm, cf. [8, 44]. We believe that the 6-Linear approach above can be considered as a stand-alone cryptanalysis method. We believe it will break some very simple ciphers on its own, through iteration, without the necessity to do anything else other than iteration of linear algebra steps on well-chosen sets of non-linear polynomials which will yield a monotone sequence of invariant polynomial spaces of increasing size. In addition, in ElimLin equations penetrate slowly inside the cipher. Here all equations penetrate inside the cipher and we can generate new polynomial relations on variables at any round inside the cipher. However, as in ElimLin, we expect that in most cases the number of equations will stabilize early on, and no new invariants will be generated.

4.12 Additional Steps and Dealing with Key Bits

We have not yet explained how to deal with key and IV bits in the cipher. This problem was already a major problem in [15] and general non-linear cryptanalysis is substantially more complex than Bi-Linear cryptanalysis. One obvious method, is for example to assume say $F = 0, L = 0$ where $L = S2$ and find invariants by solving the Fundamental Equation (FE). But then we have NOT yet broken any cipher, though we can distinguish any power of a certain complex iterated permutation from a random permutation, including a power (an integer) which is not known to the attacker, which is already a strong result. At the antipodes, we can simply try to eliminate ALL the monomials which contain the key and IV bits. As simple as that, and this solves our problem and allows the attacker to deduce things about the internal state of the cipher for arbitrarily large numbers of rounds. Another method, is to use the method above for $F = 0$, then for $F = 1$, combine the two systems FE equations, and solve a bigger system of simultaneous [linear] equations for the coefficients of Z. More specific non-trivial strategies are needed when dealing with $S2$ or L bits, cf. Section 6.1 below, while we try to avoid as much as possible dealing with $S1$ bits. But it is maybe too early to go into such fine technical cryptanalytic details.

We need first to study, not if all this is actually feasible, but more importantly, if the set of solutions is **not empty**. We are going to solve first some simplified cases, before we attack more complex situations.

5 Concrete Examples

An example is worth more than a thousand words. We present numerous examples chosen for their elegance and simplicity. Our initial approach was to construct toy ciphers fully compliant with the spec of T-310 cipher. Initially we found just some examples which worked on a reduced number of cipher bits, for example 12 out of 36 and other could be ignored, and in special cases only. The we realized that exactly THE same property can break another full-fledged cipher on 36 bits(!) in a more general setup. Working on fewer bits also allows to find invariants which are simpler, easier to study, or to find interesting examples with a higher degree. In most of our examples all 36 bits are used and we have a permutation on 36 bits. Then three quite interesting things happen: first of all, interesting invariants exist frequently, secondly, they ignore many of the 36 bits, (and also some of a reduced set of say 12 bits), and thirdly, in many interesting cases the invariant does not involve any of the IV and secret key bits. In our research we have worked by many successive approximations. Initially in our research we found plenty of examples of complex invariants of degree say 11. Then we looked for simpler low degree invariants. Moreover and importantly, we privilege on examples where invariants would be happening NOT for pure structural/wiring reasons, but such that the invariant holds for a larger[15] permutation due to circumstances which depend very **strongly** on the Boolean function Z and which allow all other 36-4 cipher state variables to be eliminated. In other terms in many cases we rather discard invariants which work for any Boolean function Z. We consider that such attacks are rather too trivial[16] or we consider that they are maybe too strong[17] and also too easy to detect in terms of more fundamental immutable properties of the cipher wiring, in order to be actually encountered in some real-life cryptanalysis (which is not what we do in this paper, which is a simple proof of concept, but remains our ultimate long-term objective).

Learning From Examples. In this paper we will first look at some low degree GLC invariants $\mathcal{P}$ true with probability 1. Such invariants exist in vast numbers. Then we searched for examples which work for the same invariant $\mathcal{P}$ and which are quite simple and elegant AND they work in more than one case, for example when $F = 0$ and 1, AND such that they do not exhibit another substantial weakness such as being insecure w.r.t. Linear Cryptanalysis (LC), and/or with $\mathcal{P}$ being irreducible etc. Due to a complex history of learning from already known examples which is a bit too long to describe, with creation of databases of interesting examples, in our presentation many examples start by specifying $\mathcal{P}$ without really explaining how this $\mathcal{P}$ was discovered in the first place (which is explained on this page).

[15] Such attacks are expected to scale to larger ciphers and to larger subsets of bits.

[16] When for example a smaller permutation on say 4 bits, cf. Section 5.2, is embedded inside our permutation on 20 bits which is a trivial form of a weak cipher.

[17] In Appendix A we show a nice real-life example of such an invariant.

5.1 A Toy Example with $\mathcal{P}$ of Degree 3 and 4 Variables and $F = 0$

We start with the following very simple non-linear round invariant property $\mathcal{P}$ which is particularly simple and uses only 4 variables:

$$\mathcal{P} = efg + efh + egh + fgh + fg$$

We are quite close to using a random bijective LZS on a reduced set of bits and $\mathcal{P} - fg$ is a symmetric homogenous polynomial of degree 3 in 4 variables only. However it is important to note that overall $\mathcal{P}$ is **NOT** a symmetric polynomial.

Here is an example of a long term key where this invariant works:

```
317: P=27,29,31,21,33,19,26,25,22,32,23,17,24,16,18,9,5,
10,35,13,36,30,34,11,2,28,14 D=17,25,26,35,18,34,30,32,28
```

What is the solution Z? All we need to do is to write our Fundamental Equation (FE) and substitute four variables using the ANF round equations in page 17.

$$\mathcal{P}(a, b, c, d, e, f, g, h, \ldots) = \mathcal{P}(b, c, d, F + i, f, g, h, F + Z1 + e, \ldots)$$

On the right hand side we replace $a \leftarrow b$, etc, up to $h \leftarrow F + Z1 + e$.

Our Fundamental Equation (FE) becomes:

$$F(fg + fh + gh) + Z(fg + fh + gh) + gh + fg$$

We will now assume $F = 0$. We get:

$$Z(fg + fh + gh) + fg + gh$$

Due to absence of variables p and t in $\mathcal{P}$, this equation FE contains only Z and not Y. More generally Z and Y are just two instances of the SAME Boolean function on two disjoint sets of variables.

We recall our Fundamental Equation (FE) which becomes:

$$Z(fg + fh + gh) = fg + gh$$

Then we need to substitute Z by an expression of type

$$Z00 + Z01 * L + Z02 * j + Z03 * L * j + Z04 * h + Z05 * L * h + \ldots$$

with the correct 6 variables L, j, h, f, p, d in the correct order. And if needed, we also replace Y by

$$Z00 + Z01 * k + Z02 * l + Z03 * k * l + \ldots$$

where the coefficients Zii will be the same.

The FE becomes then:

```
(LZ33+LZ41+Z32+Z40)*dfg + (LZ37+LZ41+Z36+Z40)*dfgh +
(LZ39+LZ43+Z38+Z42)*dfghj + (LZ55+LZ59+Z54+Z58)*dfghjp +
(LZ53+LZ57+Z52+Z56)*dfghp + (LZ35+LZ43+Z34+Z42)*dfgj +
(LZ51+LZ59+Z50+Z58)*dfgjp + (LZ49+LZ57+Z48+Z56)*dfgp +
(LZ33+LZ37+LZ41+LZ45+Z32+Z36+Z40+Z44)*dfh +
...
(LZ17+LZ21+Z16+Z20)*ghp
```

From here we obtain a system of simultaneous almost-linear[18] equations starting with:

```
L*Z33+L*Z41+Z32+Z40=0
L*Z37+L*Z41+Z36+Z40=0
...
L*Z17+L*Z21+Z16+Z20=0
```

There are many solutions to this equation. One example solution is:

$$Z(a,b,c,d,e,f) = 1 + a + c + d$$

with $L = 1$ which gives in our case:

$$Z1 = Z(L,j,h,f,p,d) = h + f$$

and it easy to check that our FE in Z holds:

$$(h+f)(fg+fh+gh) = (fgh+fh+gh) + (fg+fh+fgh) = fg+gh$$

This solution is linear and no one would buy a backdoor cipher without non-linear functions Z. Such a cipher actually have additional issues, for example $\mathcal{P} = e + h$ is also an invariant. However there are also plenty of **non-linear** solutions(!) which work all the same. For example:

$$Z(a,b,c,d,e,f) = a + d + ad + cd + f + af$$

which becomes

$$Z1 = Z(L,j,h,f,p,d) = L + f + Lf + hf + d + Ld$$

this one works only when $L = 1$. This completes the problem of constructing a non-linear invariant attack in our toy example. We have two concrete examples of Z and many other exist. However we are only able to construct an invariant in 1 case: here with $F = 0$ and $L = 1$. This invariant does not break T-310 cipher as F and L vary in different rounds.

5.2 An Essential Observation

The example above is one of the simplest we have seen and is a bit degenerate: there are plenty of Boolean functions which work and many are not very interesting. However this example has an interesting feature: it is highly modular and we can add or remove invariants (potentially) one by one. Here is a simple example. It is easy to see that if in our example, we put $Z = 1 + a + c + d$ which becomes $Z1 = h + f$ when $L = 1$, we get FE of the form:

$$\mathcal{P}(e,f,g,h) = \mathcal{P}(f,g,h,F+h+f+e)$$

[18] Here it is nearly linear because $\mathcal{P}$ is fixed, and most parts of it are linear. Except that some variables Zii are multiplied by L.

and we have bits e, f, g, h which do not depend on what happens in other parts of the cipher. A little toy cipher on 4 bits embedded inside a toy cipher on 15 bits. With other Boolean functions however, this very strong weakness goes away, the bits e, f, g, h **will** again depend on what happens in other parts of the cipher, and yet we keep the SAME non-linear invariant $\mathcal{P} = efg+efh+egh+fgh+fg$.

An Important Remark. Furthermore it is easy to see that when $Z = c+d$ our cipher has another really bad linear invariant $\mathcal{P} = e + h$ when $F = 0$ and for any L. By setting $Z \neq c + d$, for example $Z = a + d + ad + cd + f + af$ we have **removed** a linear invariant $e + h$ from our cipher, but we have **kept** many non-linear invariants. This is an essential feature of our approach, a potential and ability to **remove some invariants while keeping other**, independently, by manipulating Z. Later in Section 10.7 we will a more sophisticated example of this: we will modify an existing invariant in order to keep an invariant of degree 4 and remove an invariant of degree 2. This process is also known as an "Invariant Hopping" technique and is also studied in a separate paper [18].

5.3 Resistance to Linear Cryptanalysis

Moreover, here the reader will have to believe us, this is a bit harder to check, the key 317 with $Z = a + d + ad + cd + f + af$ has no linear invariants. This cipher setting is **not vulnerable to Linear Cryptanalysis (LC)** in none of the eight cases such as $L = 0$ or 1 and $F = 0$ or 1.

5.4 Example with $F = 1$

It is possible to see that when $F = 1$, the same LZS 317 also has a non-linear invariant for the same $Z = a + d + ad + cd + f + af$ which is also actually of degree 2:

$$\mathcal{P} = ef + fg + eh + gh$$

this is only when $F = 1$ and $L = 1$. Here the FE is $(Z + 1)(f + h) = 0$.

A "slight" problem is that $\mathcal{P}$ is not the same as when $F = 0$.

5.5 Can We Have the Same $\mathcal{P}$ when $F = 0$ and $F = 1$?

An interesting question is, whether it is possible to find invariants which work in several cases simultaneously, for example when $F = 0$ and when $F = 1$, i.e. we want the SAME invariant $\mathcal{P}$ to apply to two different permutations with the same key. The Fundamental Equation method applies all the same. We will just obtain two FE equations instead of 1, leading to fewer solutions for Z and frequently leading to no solutions. Making sure that this equation is actually solvable is one of the main problems in this research. Several examples showing that this can eventually be achieved are provided in the following sections.

5.6 Impossibility Results and Provable Security

If this cannot be done, we would like to be to be able to prove mathematically that FE has no solution and this attack is impossible for our cipher. Thus we can hope to obtain for certain cipher a security proof against our method of making a block cipher deliberately weak. There exist numerous negative general results of this type in cryptanalysis, cf. [4, 9, 14, 22] some of which will also apply here (in particular all those which also apply to partitioning cryptanalysis cf. [33, 9]. This paper leads indeed to a new well-defined way to **prove** a security of a cipher against a malicious Boolean function attack, where we will prove that FE has no solution. Such a proof can be done mathematically, or in a more automated way through formal algebra and known results in theory of polynomial ideals, using some Gröbner basis computations as a tool, or in a more obscure way with software such as a SAT solvers (which will output UNSAT, and some SAT solvers are also able to output a undeniable proof of UNSAT). Such a proof will exclude a very large number of attacks with a variety of Boolean functions and polynomials $\mathcal{P}$, which space can hardly be explored systematically.

Application to T-310. Interesting questions are: is the KT1 class of keys for T-310 cipher specified in 1970s provably secure against non-linear invariants? A recent paper shows that it is **not** secure already against linear invariants [24]. In a future paper we expect to be able to show it is NOT secure against non-linear invariants either, and this for a strictly larger percentage of long-term keys inside the class KT1. At this moment (for technical reasons due to reducing the number of variables the size of equations we solve and the number of key bits involved) all long-term key examples in this paper are educational toy examples without any direct real-life cryptanalytic significance and with very exceptions they are not of type KT1.

Are block ciphers dead? Not sure, there are substantial difficulties in making our attacks work. However this paper makes that almost everything which was known about the interaction between the Boolean functions and the security of block ciphers becomes obsolete. We need to start research on the security of block ciphers from ground zero **with a substantially richer space of attacks to defend against**(!) The maths and negative results, or attacks, remain. Positive claims of type this or other criterion makes ciphers secure are mostly fallacies. For example in the past researchers would imagine that one can remove a linear invariant by choosing a non-linear Boolean function, and there will be no more invariant attacks. In reality some invariant attacks can possibly remain when we alter the non-linear components and it is sometimes possible to remove simpler invariants and keep some complex ones only, cf. examples in Section 5.2 and 10.8 and [26].

What is the alternative? Interestingly, long time ago researchers have invented provably secure encryption, cf. for example [5] and these are not block ciphers.

6 Preliminary Remarks About Multiple Invariants

Our objective is to find a property which propagates for an arbitrarily large number of rounds in T-310, a real-life historical block cipher. This means that we need to find a **simultaneous** invariant which works in four cases $F = 0, L = 0$ up to $F = 1, L = 1$, or in eight cases if K is used. This is not a small problem and it appears to be the most difficult task to accomplish here. All other steps are comparatively quite easy and the method remains the same: we just need up to four/eight copies of our FE equations and solve these combined equations for a simultaneous solution. Specific examples will be shown below.

6.1 Dealing with L

Only 1 key bit is used per round. We need to (and we will be able to) find invariants early on, before the permutation becomes too complex. Moreover, as already explained, the key bits S1 we aim to eliminate them totally by using specific sets of variables on one side. Then it is possible to see that for the bits S2 a.k.a. L are not always excessively hard to deal with if we are allowed to manipulate the Boolean function. This because of the very specific way (not very strong) in which the key bit L is used in T-310. We can observe that if we restrict our attention to non-linear invariants which concern all the bits $a - t$, and Z1 non-linear function only, then $\mathcal{P}$ can use up to the whole 28 bits, which is plenty to explore for the attacker, and yet L is not used anymore[19]. We can then write and solve two FE equations for each case $L = 0$ and $L = 1$, and it is possible to see that IF both FE have solutions, we can produce a solution Z which will work for every L as follows.

Theorem 6.1.1 (Combination Theorem). We assume that in our FE problem Z2 is not used. We assume that we can solve the FE problem for Z independently in the case $L = 0$, and we obtain at least one Boolean function $Z_0(b, c, d, e, f)$, and also that we can solve the FE problem with the same $\mathcal{P}$ for $L = 1$ and obtain another Boolean function $Z_1(b, c, d, e, f)$. Furthermore we assume that at least **one** of these Boolean functions is non-zero. Then we can COMBINE the two solutions as follows:

$$Z(a, b, c, d, e, f) = a \cdot Z_1(b, c, d, e, f) + (a - 1) \cdot Z_0(b, c, d, e, f)$$

Moreover we can multiply Z by any polynomial in a and in any variable not used in either Z_0 and Z_1 (typically just a) and we still get a valid invariant.
Proof: It is sufficient to see that with the combined Boolean function $\mathcal{P}$ is an invariant property for our permutation on 36 bits, for both $L = 0$ and $L = 1$.
Remark: this method is of limited interest and does not work when $Z2$ is used.

[19] Moreover there could be further properties which eliminate L with more bits

7 Construction of Multiple Simultaneous Invariants

7.1 A Strong Invariant Example with $F = 0$ and $F = 1$

On the algebraic front, very frequently the set of solutions is an empty set. We need therefore to examine a larger variety of LZS. For example we found the following example:

```
827: P=34,32,25,30,19,28,18,35,31,33,23,36,24,22,5,1,
13,17,16,10,21,6,20,29,9,15,3 D=21,17,29,24,27,20,31,36,32
```

The substitutions are therefore, omitting the trivial ones of type $a \leftarrow b$, as follows:

$$d \leftarrow F + e$$
$$h \leftarrow F + Z1 + a$$
$$Z1 \leftarrow Z(L, c, e, l, g, r))$$

Consider the following polynomial:

$$\mathcal{P} = ae + bf + cg + dh + e + f + g + h$$

again we do not use Z2 and the fundamental equation is a remarkably simple set of 2 equations, for $F = 0$ and $F = 1$:

$$Z + Ze + a + e = 0$$
$$Ze = 0$$

One interesting solution, specifically avoiding solutions which involve first variable a is: $Z(a, b, c, d, e, f) = bde + bcde$ which becomes

$$Z1 = Z(L, c, e, l, g, r) = (e + 1)clg$$

This is an educational example chosen for its simplicity, the FE is very short and composed of two equations which do not contradict each other.

Discussion. This setup is not excessively good yet. It is weak w.r.t Linear Cryptanalysis (LC) in some cases. We have found numerous examples not weak w.r.t. LC, however the FE is more complex. Moreover in general we are still not quite happy: our quadratic invariant works only in 2 out of 4 cases, when $L = 0$ for example. We have NOT YET broken a block cipher. We need to construct polynomial invariants which work in 4 cases simultaneously, or in 8 cases simultaneously if K is also used.

7.2 A Tentative Construction of an Advanced Multiple Invariant

On this page we show that our key 827 allows a yet stronger form of attack to be constructed. Consider the following very simple polynomial:

$$\mathcal{P} = ae + bf + cg + dh$$

Let $F = 0$, then we write our FE which becomes extremely simple $Ze = 0$ and it does NOT depend on L. Therefore when $F = 0$ our invariant $\mathcal{P}$ works for any L. It remains to see what happens for $F = 1$. In this case it is possible to see that we get another invariant which is the one from the previous section:

$$\mathcal{P} = ae + bf + cg + dh + e + f + g + h$$

Interestingly the difference of the two invariants is very simple $e+f+g+h$. From this we can construct a more general invariant, 6-linear attack in the spirit of Section 4.10 with FE being of the form:

$$\mathcal{P}(a, b, c, d, e, f, \ldots) = \mathcal{P}(b, \ldots, F + Z1 + r + g, \ldots) + Q(F, L, I)$$

and

$$Q(F, L, M) = F \cdot I$$

More precisely, combining the two results, we have the following **quadruple** invariant which works for any F and any L:

$$\mathcal{P}(a, b, c, d, e, f, g, h) = \mathcal{P}(b, c, d, F + e, f, g, F + Z1 + a) + F \cdot (e + f + g + h)$$

where

$$\mathcal{P}(a, b, c, d, e, f, g, h) = ae + bf + cg + dh$$

7.3 A Combined Invariant Attack with 2 Stages

In the example above $I = e + f + g + h$ is linear and the attacker could predict the value of $\mathcal{P}(a, b, c, d, e, f, g, h) = ae + bf + cg + dh$ after an arbitrarily large number of rounds IF he can predict the values of $I = e+f+g+h$ at every round. Now, there are good chances that BOTH properties CAN be combined(!) in one single vulnerable long-term key (and polynomial I does not have to be linear). It is possible to see that both the property $\mathcal{P}(left) = \mathcal{P}(right) + F \cdot (e+f+g+h)$ occurs IF AN ONLY if a certain number of constraints C1 on $\mathcal{P}, D$ are satisfied, and moreover if another set of constraints C2 on $\mathcal{P}, D$ are satisfied, the attacker can also predict bits I at every round [possibly with a partial key guess]. The properties C1 and C2 are not always compatible and can be studied through machine learning, or through exact mathematical theorems[20]. In the case when pre-conditions C1 and C2 are compatible, which will happen sometimes[21], then we have found a way to **predict 2 bits for an arbitrary number of rounds**

[20] One example of how C1 may look like is given in Section 7.6. Numerous other examples can be found in Section 21.22 page 87 in [21] and numerous examples of exact theorems which show how pre-conditions of $\mathcal{P}, D$ lead to an invariant property can be found in [21], see for example Theorem J.1.1. on page 173.

[21] In Section 21.14. [21] we see one example when two such conditions are compatible.

of our cipher and long term keys which satisfies both conditions C1 and C2 can easily be constructed[22].

7.4 A Concrete Example of a Multiple Invariant or How to Backdoor T-310

In the previous example we have not shown if conditions C1 and C2 can be made to be compatible. An interesting question is, can we do better, and find a simpler example when an invariant can be constructed. Moreover, can this be done on demand, say a key we have not chosen, for example the same key 827 as above, which will be a good indication that this sort of invariants are quite common.

```
827: P=34,32,25,30,19,28,18,35,31,33,23,36,24,22,5,1,
13,17,16,10,21,6,20,29,9,15,3 D=21,17,29,24,27,20,31,36,32
```

The answer is yes and in order to approach this problem we add a slight amount of arbitrary constraints to the FE equation: for example we select two or three higher degree coefficients of the polynomial $\mathcal{P}$ and fix them to one. This allow us to generate a large variety of solutions to the FE equation. In particular there exist a number of degenerate cases where there is no need for four/eight FE equations, because some of them identical(!). Such cases are very nice as a proof of concept because of their simplicity. The discovery procedure is therefore essentially trial and error. For example we consider the following very simple polynomial:

$$\mathcal{P} = a+b+c+ac+d+bd+e+ce+f+df+g+ag+eg+h+bh+fh$$

Again we replace $a \leftarrow b$, etc, with $d \leftarrow F+e$ and

$$h \leftarrow F + Z1 + a$$
$$Z1 \leftarrow Z(L,c,e,l,g,r))$$

the fundamental equation is then particularly simple:

$$Z = Z(c+g)$$

[22] In general this is done using a SAT solver, the solving complexity of this problem is very low, the coding is however very complicated.

All this can be illustrated on the following picture made by UCL student Marios Georgiou [32]. Each cycle can be seen as cancelling all the monomials without F and without Z which appear inside for the purpose of writing the FE.

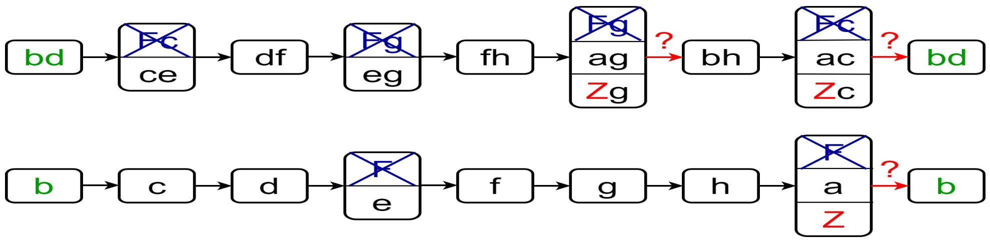

Fig. 5. A detailed explanation for our invariant which shows terms which cancel.

Blue pieces are those which eventually cancel out, remembering that all F are identical and belong to the same round. The red transitions with a question mark are those which may or not work depending on Z (ultimately they will work if FE has a solution so additional terms could be generated here and cancelled elsewhere). The green and black terms are those which stay and belong to our invariant $\mathcal{P}$ above. This figure also explains that $\mathcal{P}$ is eseentially a sum of 2 symmetric polynomials with 2 cycles of length 8 (however we checked that it is NOT true that an invariant for 8 rounds simpler/shorter than $\mathcal{P}$ would exist).

Now because the FE does not depend on neither F nor L we do not need four copies of it but just one. Here is one solution:

$$Z = e + be + ce + bce + bf + bcf + bef + bcef$$

We have an invariant on 8 bits 29-36 the key feature of which is that it will completely ignore 4 bits which come from other parts of the cipher. This is shown on the picture below made by UCL student Marios Georgiou [32].

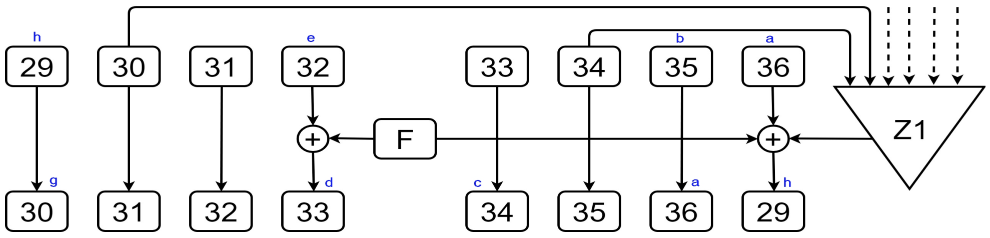

Fig. 6. A detailed explanation of how our invariant can work while completely ignoring 4 bits which enter Z1.

This completes a construction of a non-linear round invariant. We have checked that there is no linear invariant in any of the 4 cases and therefore Linear Cryptanalysis (LC) does not work here. Our **non-linear invariant** $\mathcal{P}$ works in all eight cases and therefore it propagates **for an arbitrary number of rounds** for any key and for any IV.

7.5 Extension for Boolean Functions Using *a*

In the previous solution, we do not use the L variable (a.k.a. 'a' letter in the Boolean function ANF). A Boolean function not using one variable in a block cipher looks very bizarre. An interesting question is whether the same LZS can also be made weak and work in all 4 cases with a Boolean function using 'a'. The answer is yes and one method achieve this (observed my UCL student Marios Georgiou [32]) is again to solve the same equation:

$$Z = Z(c + g)$$

and we multiply the resulting solution Z by $(1 + a)$. The resulting Boolean function does the job and it depends on a, and therefore it depends also on L if we use $Z1$ which we do here.

Remark. This sort of extension or improvement can also be achieved using our Thm. 6.1.1 page 29.

7.6 Pre-Conditions for Our Multiple Invariant

It is possible to see that (as observed by UCL student Marios Georgiou [32]) the invariant observed in Section 7.4 can be constructed systematically [the number of long-term keys which have the same property will be very large, of the order of 2^{70}] and it will occur each time the following set of conditions are satisfied:

$$\begin{cases} D(9) = 32 \\ D(8) = 36 \\ (1 + c + g)Z(L, P[1-5]) \equiv 0 \\ Z \not\equiv 0 \end{cases}$$

In particular this implies that one of the $P(1-5)$ values must be equal to bit 34, which corresponds to letter 'c'.

8 More Complex Examples

Until now our examples exploited primarily a tiny subset of 36 bits and moreover the LZS examples specified were not bijective and therefore not secure w.r.t. the already known ciphertext-only attack of [22]. Below we present two more convincing examples which are secure against the attack of [22] and where inputs from different parts of the cipher are better mixed together.

8.1 An Example Using Z1 and Z2

We found the following nice yet non-trivial example:

```
799: P=25,35,21,31,26,2,28,29,33,23,12,27,20,13,5,14,
18,32,1,16,30,8,7,34,6,24,9 D=0,4,16,8,12,36,32,24,28
```

The round function is a bijection on 36 bits for any key and any IV. There is no linear invariants and there is **no weakness of any sort** w.r.t to any previously known attack on T-310. We then consider the following polynomial quadratic invariant as follows:

$$bi+cj+dk+il+ej+fk+gl+eh+fm+gn+ho+mp+an+bo+cp+ad+bm+cn+do+$$

$$ip+aj+bk+cl+de+fi+gj+hk+lm+en+fo+gp+ah$$

which polynomial is irreducible and has 32 monomials of degree 2. The fundamental equation is then (all eight versions are identical):

$$Yd+Yh+Yj+Yn+Zb+Zf+Zl+Zp$$

Solving this FE leads to the following solution (not unique):

$$Z = 1+b+c+d+e+be+ce+de+f+bf+cf+df+ef+bcdef$$

This FE and this solution is not trivial in two ways: there is no obvious linear solution Z, and also because the four inputs of Z and Y which are actually used in Z and Y are NOT [23] at the same positions in Z and in Y. In the same way as in Section 7.5, this Boolean function could be multiplied by any polynomial in a. Here again we obtain a perfect multiple invariant which works for any F, for any K, L and for any number of rounds.

[23] The difference lies in the last variables 'k' and 'j', one of them replaces input 5 the other input 6. Therefore Z and Y are annihilated in a (slightly) different way.

8.2 An Example of an Invariant with Z1 and Z4

We focus on 8+8 bits 29-36 and 5-12 pertaining to $Z1$ and $Z4$ only and we will strictly be avoiding anything between g2 and g7. Let $g27 \overset{def}{=} g2 + g7$ cf. Fig. 2. In order to generate such an invariant, we have done a detailed analysis of cycles which involve various products of degree 2, some transitions are mandatory, some will depend on the Di values. We ignore the boxes with blue crosses which we hope might eventually be eliminated later inside the final FE which is not yet found. We show here a glimpse of what we obtained in this case:

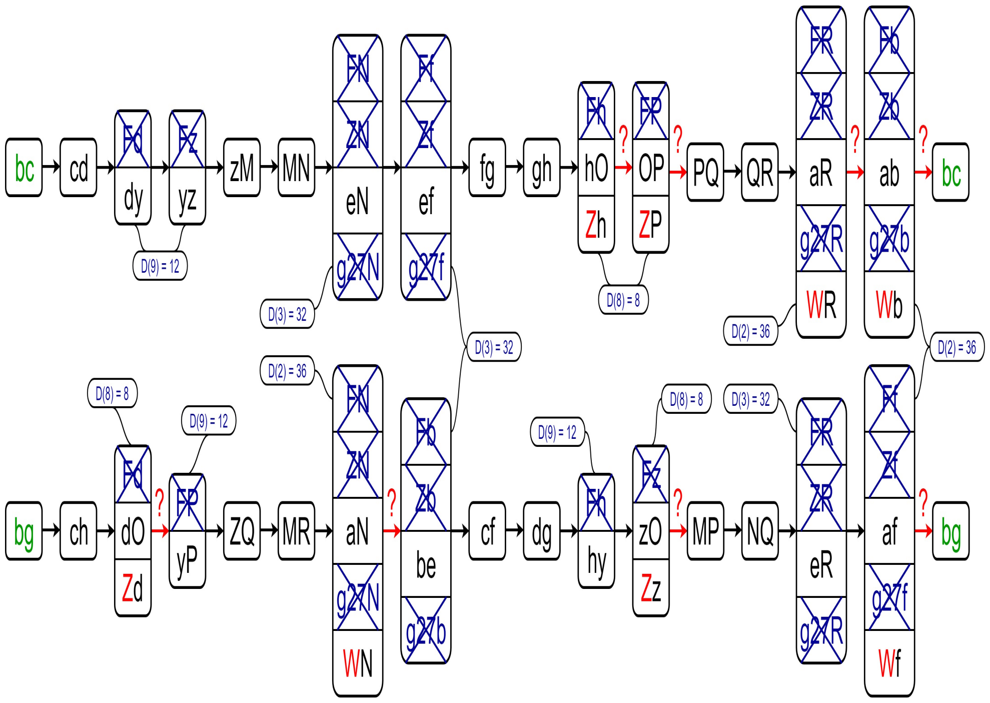

Fig. 7. A detailed analysis of transitions we aim at using in our invariant. The boxes with crosses are terms we hope to cancel later. Transitions in red with ? depend on Z and will eventually work only if our final FE equation has a solution.

This example was found by UCL student Marios Georgiou by paper and pencil with some help of software [32]. Many attempts are needed and most attempts fail. By trying to put several such cycles together we obtain the following plausible invariant:

$$\mathcal{P} = bc+cd+dy+yz+zM+MN+eN+ef+fg+gh+hO+OP+PQ+QR+aR+ab+bg+ch+ dO + yP + zQ + MR + aN + be + cf + dg + hy + zO + MP + NQ + eR + af.$$

Here is an example of a LZS where this $\mathcal{P}$ works:

```
714: P=11,7,30,29,33,1,20,17,2,15,14,27,36,24,18,8,19,
23,28,32,4,16,31,9,35,5,13 D=16,36,32,24,4,28,20,8,12
```

Our $\mathcal{P}$ is irreducible and the FE is:

$$NW + PZ + RW + Wb + Wf + Zd + Zh + Zz$$

In general, it is possible to see that the conditions which are needed for this invariant to work are the following four:

$$\begin{cases} D(2) = 4 \cdot 9 \\ D(3) = 4 \cdot 8 \\ D(8) = 4 \cdot 2 \\ D(9) = 4 \cdot 3 \end{cases}$$

These conditions lead to a specific configuration as follows:

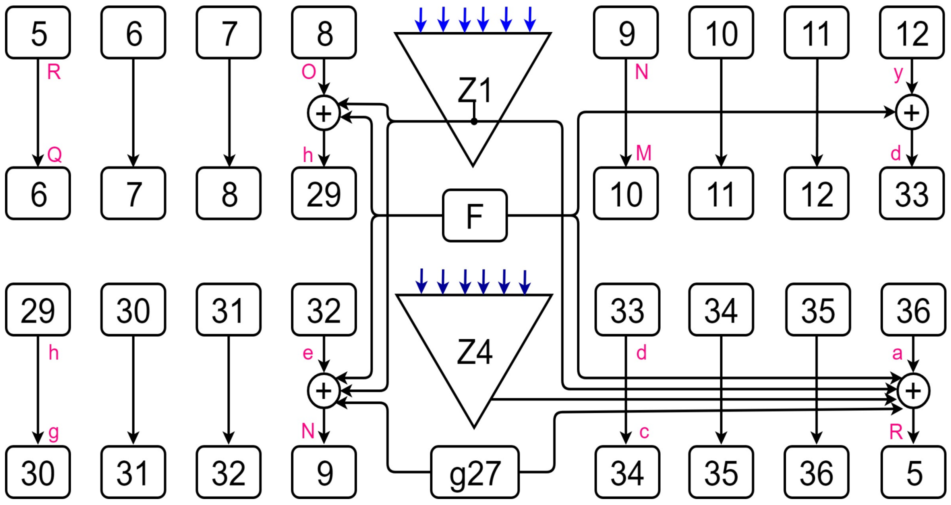

Fig. 8. Some important connections inside our invariant for key 771.

Solving the FE. It remains to find a solution to $NW + PZ + RW + Wb + Wf + Zd + Zh + Zz \equiv 0$. With LZS 714 above, one possible solution is

$$Z = 1 + dc + cb + fb + b + c + de + df + db + e + f + d + eb.$$

We get another invariant which works for any number of rounds and any key. It is bijective and has no linear invariants and no previously known attacks work.

Remark. This invariant connects two remote parts of the cipher and works only in a case where Z is weak and the LZS is quite special. this can also happen with KT1 keys and for any Z, see Section B.

8.3 An Invariant with Z1, Z2 and Z4

In the same way we have constructed another yet more complex invariant which involves three Boolean functions Z1,Z2 and Z4 (and yet it deliberately avoids $Z3$ and all the 120 secret key bits of type $S2$). This invariant can be found in [27].

9 Additional Examples

9.1 An Example of An Invariant Which Involves Key Bits

Below we show another example with a permutation on 36 bits. An interesting feature of this example is that it follows the Extended 5-Linear FE (5EFE) framework of Section 4.9 of type

$$\mathcal{P}(a,b,c,\ldots) = \mathcal{P}(b,\ldots,F+Z1+\ldots,\ldots) + Q(F,L)$$

where Q will actually depend on $L = S2$ key bit at a given round. It is basically a proof of concept of an invariant where key bits are involved and which is therefore suitable and can be directly used in key recovery attacks.

```
771: P=31,35,16,25,21,24,12,27,29,8,33,23,36,14,3,11,
19,32,22,4,9,28,2,20,6,1,30 D=20,0,12,16,8,36,32,24,28
```

The round function is a bijection on 36 bits for any key and any IV, (thus it not vulnerable to the attack of [22]), and there is no linear invariants and there is no other weakness w.r.t to previously known attacks. We consider the following excessively simple polynomial:

$$\mathcal{P} = b+c+d+f+g+eg+h+fh+i+j+k+fl+m+n+o$$

which polynomial is irreducible and has only three non-linear monomials, so it one of the simplest examples. Moreover it is not a symmetric polynomial in the slightest, showing that a wider variety of solutions is possible(!).

The fundamental equation is then (all eight versions are identical):

$$Z+b+f+fl+l+p$$

It has a unique solution:

$$Z = c+a+f+b+e+be$$

which has only one(!) non-linear term. Again we obtain a perfect multiple invariant which works for any F, for any K, L and for any number of rounds and we have:

$$Q(F,L) = L$$

We have obtained an invariant which allows to obtain sums combinations of key bits $S2$ used in different rounds. This corresponds to an open problem left on the last page of [23] where only attacks which work with $S1$ are found. Here we can do key recovery for $S2$ as well. However we do it for a special Boolean function, for the original Boolean function the problem remains not solved.

9.2 Lessons About Linear Cryptanalysis

It is widely believed that even though Differential Cryptanalysis (DC) was known already in the 1970s, cf. [6], and Linear Cryptanalysis (LC) was not. This is clearly not true and it appears that Linear Cryptanalysis was also already known in 1970s, cf. [24]. There are numerous works in cryptography in which security is proven against Linear Cryptanalysis. We are now going to show a funny example: a non-linear invariant with only **one** non-linear term which nevertheless works for any number of rounds, any key and any IV. It was found by paper and pencil explorations by UCL student Marios Georgiou [32]. The invariant is:

$$\mathcal{P} = v + w + x + O + P + Q + R + uv$$

one LZS where this invariant works is:

```
120: P=33,1,36,3,2,20,26,21,28,7,4,18,5,17,29,27,6,9,
34,8,24,14,16,35,15,12,32 D=20,16,8,12,28,4,36,32,24
```

and the FE is $W + u + v + y + uv + vw$ with a **unique**[24] solution $Z = c + e + be + ce + f$. We have checked that this cipher is a key-dependent bijection on 36 bits and is secure w.r.t. linear invariants. In other terms it is resistant to all previously known attacks including recent ciphertext-only attacks in [22].

[24] See Section 12.1 and last point 4. in Section 12.2 to see why this matters.

9.3 An Example with Two Monomials

And here is example with only two non-linear monomials. It was found by UCL student Marios Georgiou [32]. We consider

$$\mathcal{P} = an + gn + u + v + w + x + O + P + Q + R$$

One long-term key which works is

```
993: P=17,4,8,5,24,28,21,33,18,26,14,19,34,2,1,15,11,
27,12,16,29,30,36,22,35,23,9 D=0,36,4,8,12,20,24,28,32
```

the FE is

$$W + a + an + bo + gn + ho = 0$$

which equation has again a unique[25] solution: $Z = c + ad + ed + cf + bf$.

Again this cipher setup gives a bijection on 36 bits secure w.r.t. all previously known attacks.

9.4 Further Observations on Our Example with Two Monomials

This cipher setup has an interesting partitioning of products in W: half of them $an + gn$ are connected to inputs, and half are connected to outputs with $bo + ho$. A whole lot of bits inside the cipher are never used in the invariant, yet all [26] the 6 bits of Z are used inside the invariant. In particular this invariant has a short cycle such that the keys bits are not used inside the cycle. Finally we remark that $g3$ is used twice inside this cycle and all the non-linear products occur outside of the cycle.

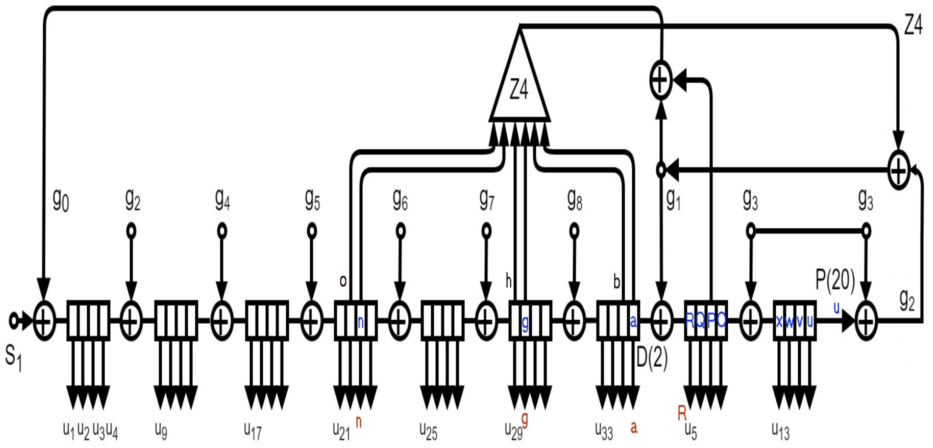

Fig. 9. Our construction of an invariant with only two non-linear terms.

[25] The idea is that if we are able to find cases where the solution is unique, it means that maybe we could have a backdooring method which is more specific to a very specific cipher setup.

[26] See Section 12.1 to see why this matters.

10 Higher Degree Invariants

Most invariants we have found or presented so far are of degree 2. The same is true for invariants studied in [49] and also in [15]. There are substantial difficulties which make the study of higher degree invariants problematic: mainly just the large size and the computing power needed to compute them. How can we ever hope to find any higher degree attacks, knowing that the space for possible attacks is extremely large? Can we at all display any such invariants inside this paper? In order to be able to do this and as a proof of concept, we do it in one case where the number of variables and monomials used is not too large or/and the polynomials have some special structure which makes them easier to discover and manipulate and also just to print inside a paper. In this section we provide examples with higher degrees up to 8. Systematic exploration is already completely infeasible at degree 2. This of course just a proof of concept and countless more complex examples exist.

10.1 An Example with Three Monomials and Invariant of Degree 3

Here is another example with only three non-linear monomials. It was found by UCL student Vlad Popa. We consider

$$\mathcal{P} = h + m + n + o + p + O + P + Q + emS + fnT + goU$$

One long-term key which works is

```
P=34,23,25,30,19,5,1,21,29,4,32,24,36,22,28,18,13,17,
16,10,35,6,20,31,9,15,3 D=12,4,20,16,28,24,32,8,36
```

the FE is

$$Y = h + Vph + Sem$$

which equation has again a unique solution: $Z = c + abc + def$.

10.2 More Examples of Invariants of Degree 3

A similar example with a unique solution and just one cubic term was found by another UCL student Abby Taylor, where $Z = e + ad + ac + abc + bf + be + bde + b + f + d + a$ and $\mathcal{P} = R + Qvu + Qu + x + wu + v$ and one long-term key which works is

```
P=34,32,25,30,19,28,18,35,31,33,23,36,24,22,5,1,13,
17,10,4,16,15,6,14,5,13,2,D=12,4,20,16,8,24,32,28,36
```

Another example was found by Carlota Ortega-Vega: $\mathcal{P} = Pyg+Py+p+h+g$ and one long-term key which works is

```
P=34,32,25,18,19,28,7,12,30,6,11,29,24,22,5,1,13,17,
10,4,16,15,35,14,5,13,2,D=12,4,20,16,8,21,32,28,36
```

and $Z = abc + ab + c + def + de$.

10.3 Another Example of An Invariant of Degree 3

Here is another example of an invariant of degree 3. In this example the polynomial $\mathcal{P}$ is irreducible.

```
338: D=5,21,33,3,30,25,32,24,36,P=32,31,29,27,18,20,33,22,34,25,35,24,28,5,30,3,15,9,8,2,11,13,10,16,7,12,4
R1=1+1 D4=2 at degree 3
P=e+f+ef+g+eg+fg+h+eh+fh+efh+gh+egh+fgh+i+ei+fi+efi+gi+egi+fgi+hi+fhi+ghi+j+ej+fj+hj+ehj+fhj+ij+eij+fij+hij+k+ek+fk+efk+gk+
egk+fgk+ik+fik+gik+jk+ejk+fjk+ijk+l+el+fl+efl+gl+egl+fgl+il+fil+gil+jl+ejl+fjl+ijl+m+fm+gm+hm+fhm+ghm+im+fim+gim+jm+hjm+ijm+
km+fkm+gkm+jkm+lm+flm+glm+jlm+n+en+gn+hn+ehn+ghn+in+ein+gin+hin+jn+hjn+ijn+kn+ekn+gkn+ikn+jkn+ln+eln+gln+iln+jln+mn+hmn+imn+
kmn+lmn+o+eo+fo+efo+go+ego+fgo+ho+eho+gho+io+eio+fio+hio+jo+ejo+fjo+hjo+ko+eko+gko+iko+jko+lo+elo+glo+ilo+jlo+mo+fmo+gmo+hmo+
imo+jmo+kmo+lmo+no+eno+gno+ino+jno+mno+p+ep+fp+efp+gp+egp+fgp+hp+ehp+fhp+ip+eip+gip+hip+jp+ejp+fjp+ijp+kp+ekp+fkp+ikp+lp+elp+
flp+ilp+mp+fmp+gmp+hmp+imp+jmp+kmp+lmp+np+enp+gnp+hnp+jnp+knp+lnp+mnp+op+fop+gop+hop+iop+jop+kop+lop+nop+q+fq+gq+hq+fhq+ghq+iq+
fiq+giq+jq+hjq+ijq+kq+fkq+gkq+jkq+lq+flq+glq+jlq+nq+hnq+inq+knq+lnq+oq+foq+goq+hoq+ioq+joq+koq+loq+noq+pq+fpq+gpq+hpq+ipq+jpq+
kpq+lpq+npq+r+er+gr+hr+ehr+ghr+ir+eir+gir+hir+jr+hjr+ijr+kr+ekr+gkr+ikr+jkr+lr+elr+glr+ilr+jlr+mr+hmr+imr+kmr+lmr+or+eor+gor+
ior+jor+mor+pr+epr+gpr+hpr+jpr+kpr+lpr+mpr+opr+qr+hqr+iqr+kqr+lqr+oqr+pqr+s+es+fs+efs+gs+egs+fgs+hs+ehs+ghs+is+eis+fis+his+js+
ejs+fjs+hjs+ks+eks+gks+iks+jks+ls+els+gls+ils+jls+ms+fms+gms+hms+ims+jms+kms+lms+ns+ens+gns+ins+jns+mns+ps+fps+gps+hps+ips+jps+
kps+lps+nps+qs+fqs+gqs+hqs+iqs+jqs+kqs+lqs+nqs+rs+ers+grs+irs+jrs+mrs+prs+qrs+t+et+ft+eft+gt+egt+fgt+ht+eht+fht+it+eit+git+hit+
jt+ejt+fjt+ijt+kt+ekt+fkt+ikt+lt+elt+flt+ilt+mt+fmt+gmt+hmt+imt+jmt+kmt+lmt+nt+ent+gnt+hnt+jnt+knt+lnt+mnt+ot+fot+got+hot+iot+
jot+kot+lot+not+qt+fqt+gqt+hqt+iqt+jqt+kqt+lqt+nqt+rt+ert+grt+hrt+jrt+krt+lrt+mrt+ort+qrt+st+fst+gst+hst+ist+jst+kst+lst+nst+rst
```

We refer to Appendix C for the definition of $R1$ and $D4$.

10.4 A Product Example Invariant of Degree 3

UCL student Romain Dumon has proposed the following example:

$$\begin{cases} A \overset{def}{=} (N+R) \\ B \overset{def}{=} (M+Q) \\ C \overset{def}{=} (z+P) \end{cases}$$

here the FE becomes:

$$AC(U+S+P+z+W)=0$$

an example of LZS is:

```
P=34,32,25,30,19,28,18,35,31,33,23,36,24,22,5,1,13,17,
16,10,11,7,9,5,4,2,3,D=21,2,4,24,27,20,31,36,32
```

this works with both linear and non-linear Z, for example $Z=a+b+e+f$.

10.5 A Super Simple Example of An Invariant of Degree 4

UCL student Charles Albert Desbaux has proposed the following example:
$\mathcal{P} = t + srgf + h$
here the FE becomes:

$$FE = Y + fgrs + ghst + h + t$$

which has a unique solution $Z = a + b + cdef + abcd$ and an example of LZS is:

```
P=34,32,25,30,19,28,29,17,30,18,31,19,24,22,5,1,13,17,
16,15,1,2,4,24,14,13,3 D=4,8,16,36,28,20,31,24,32
```

10.6 An Example of An Invariant of Degree 4

We start by yet another invariant we found such that $\mathcal{P}$ is irreducible

For example we consider:

$$\mathcal{P} = eg + fh + eo + fp + gm + hn + mo + np$$

which is equal to (as observed by UCL student Aidan Patrick):

$$\mathcal{P} = (e + m) \cdot (g + o) + (f + n) \cdot (h + p)$$

and the following key which is of type KT1:

```
551: P=17,4,33,12,10,8,5,11,9,30,22,24,20,2,21,34,1,25,
13,28,14,16,36,29,32,23,27 D=0,12,4,36,16,32,20,8,24
```

the FE becomes

$$Y(g + o) = m(g + o)$$

and one solution is $Z = 1 + d + e + f + de + cde + def$.

Higher Degree Invariants Now we study higher degree invariants. Such invariants if they exist are NOT uniquely determined, they may depend on the choice of the Boolean function in the previous step (unless the solution was unique). Here considering only invariants with the same 8 variables we found that there exist only 2 polynomial invariants: 1 of degree 2 as above, and one another of degree 4.

$$\mathcal{P} = efgh + fghm + eghn + ghmn + efho + fhmo + ehno + hmno+$$
$$efgp + fgmp + egnp + gmnp + efop + fmop + enop + mnop$$

This contradicts a very naive idea that the number of invariants must grow as the degree grows[27]. No method to explore all possible invariants with more variables is known and possibly such method does not[28] exist. Overall there are no invariants of degrees other than 2 and 4 for 8 variables. In particular no linear approximations true with probability 1 exist for LZS 551 and this is also true using all 36 variables.

More Details This second polynomial is NOT irreducible and it factors as $(e + m)(f + n)(g + o)(h + p)$.

For degree 4 the FE becomes $Yfgh+Yfgp+Yfho+Yfop+Yghn+Ygnp+Yhno+Ynop+fghm+fgmp+fhmo+fmop+ghmn+gmnp+hmno+mnop$ which is the same as:

$$m(h + p)(g + o)(f + n) = Y \cdot (h + p)(g + o)(f + n)$$

This decomposition implies that any solution Z which is a solution to the previous FE will also work here.

[27] When we limit the set of variables there will be a fixed set of invariants. In broader sense we expect that yes, the vulnerability against all possible polynomial invariant attacks of degree at most d increases very substantially when the degree d increases, cf. [26].

[28] In spite of all this we believe that in this case invariants of higher degrees such as 5 or 6 do not exist, and for yet higher degrees, it is very hard to know for sure, they could exist.

Invariants for Two or More Rounds Moreover if we look at invariants which work for 2 rounds of encryption and not for 1 round, it is possible to see that our polynomial splits into two invariants which will work for two rounds:

$$\mathcal{P} = (e+m)\cdot(g+o)$$

and

$$\mathcal{P}' = (f+n)\cdot(h+p)$$

moreover it is easy to see that after one round we have the following two transitions $\mathcal{P} \mapsto \mathcal{P}' \mapsto \mathcal{P}$ or in other terms we have

$$(e+m)\cdot(g+o) \mapsto (f+n)\cdot(h+p) \mapsto (e+m)\cdot(g+o)$$

where the first transition is trivial and works for any LZS, and the other is non-trivial and works for this LZS 551 and with the special Boolean function.

Remark. In Appendix B we find another very similar example with invariants of high degree which also works for KT1 keys. It uses twice as many variables. This example in Appendix B is however degenerated and everything is essentially a consequence of a linear attack of a certain particular shape. In this section there is no linear attack of any sort, and yet we get complex higher degree invariants.

10.7 A More Autonomous Example of An Invariant of Degree 4

In the previous section we found an invariant of degree 4 but an invariant of degree 2 also did exist. Is it possible to find an invariant of degree 4 such that no invariant at degrees 1,2,3 exists for all the 36 variables? Yes and it is quite easy. We can for example try to modify the example from the previous section. the key 551 can also do that, all we have to do is to select a different Boolean function. We define:

$$\begin{cases} A \overset{def}{=} (e+m) \\ B \overset{def}{=} (f+n) \\ C \overset{def}{=} (g+o) \\ D \overset{def}{=} (h+p) \end{cases}$$

We recall from the previous pages that in order for $ABCD$ to be invariant the Boolean function needs to satisfy:

$$mDCB = YDCB$$

and in order for $AC + BD$ to be an invariant, it needed to satisfy:

$$YC = mC$$

All we have to do now is to find a solution Y for example for this polynomial equation:

$$mB = YB$$

which is different than the most trivial solution $Y \equiv m$ and which then will not satisfy the other equation. For example $1 + n + nm + f + mf$ using the same variable names. This solution satisfies one FE and not the other FE and this is everything we need. An actual solution forces us to modify LZS 551 very slightly: we just need to make sure that letters f and n are actually inputs of Y. Only two modifications are needed. Here is an example solution:

```
558: P=17,4,33,12,10,8,23,24,31,25,16,10,20,2,21,34,
1,25,13,28,14,16,36,29,32,23,27 D=0,12,4,36,16,32,20,8,24
Z=1+a+ab+c+bc
```

We have checked that no other invariants at degree up to 3 exist with all the 36 state variables for $1, 2, 3, 4, 5, \ldots$ and many other numbers of rounds, all this in contrast with special invariants for 2 or 4 rounds we have seen above. All these invariants have been removed by changing the Boolean function.

10.8 A Yet Stronger Example

It is also possible to solve an even more complex equation:

$$mBC = YBC$$

in such a way that we use as inputs of Z all the 5 variables which appear in this equation and that a solution does not satisfy the previous equation $mB = YB$.

One solution is as follows:

```
550: P=17,4,33,12,10,8,22,23,24,31,30,20,20,2,21,34,
1,25,13,28,14,16,36,29,32,23,27 D=0,12,4,36,16,32,20,8,24
Z(a,b,c,d,e,f)=1+b+c+d+aef+abef
```

It happens that our solution satisfies the following equation:

$$(Y + m)(f + n) = (Rnf + Rf)go$$

where R happens to be the 6th input of Y in the LZS above and could also be another letter. We found another example where the only invariant is of degree 4 and no other invariants at degree up to 3 exist with all the 36 state variables for various numbers of rounds. Just one invariant remains after changing the Boolean function.

10.9 Another Strong Autonomous Example of Degree 4

UCL student Anthony Romm has proposed the following example:

$$\begin{cases} A \overset{def}{=} (u + O) \\ B \overset{def}{=} (v + Q) \\ C \overset{def}{=} (w + P) \\ D \overset{def}{=} (x + R) \end{cases}$$

which is an interesting example because the two sets of 4 letters do NOT follow the same order. The invariant is $ABCD$ for 1 round, there is no invariant of lower degree, not even for a very simple Boolean function such as:

$$Z = ed + a$$

an example of LZS is:

```
P=15,23,6,10,19,32,2,30,22,29,21,3,36,4,31,18,9,17,34,
28,16,25,14,5,13,35,24 D=12,8,20,28,16,4,36,32,24
```

moreover it is to see that this example works for an excessively large number of Boolean functions, the Fundamental Equation is:

$$BCD(W + u) = 0$$

this example is somewhat better than the one on the previous page, because W is multiplied by 3 linear terms which 1) adds more symmetries 2) yields more solutions (one simple solution is $Z = ed + a$ above). However a serious drawback of this example is that it does not seem to have any "truly convincing" solutions for Z, in our view, where the Boolean function would not have annihilators of degree 2 and which would require all the 6 input variables of Z to be within the 8 inputs of $ABCD$.

10.10 Automatic Invariant Finder

Five UCL students have made an open source tool for constructing long-term keys and Boolean functions which may work for a given polynomial invariant. The authors are Kristelle Feghali, Julien Nahum, Brian Gunawan, Raymond Tan, Wayne Tsui. The tool is written in Python and SAGE and can be executed directly in SAGE. The current version is able to discover properties with 1 of the 4 Boolean functions being active. It enumerates possible solutions and recognises automatically if they cannot be correct. Current limitations are that only one Boolean function in W, X, Y, Z can be used at one time, and the FE should not contain terms which are non-linear in W, X, Y, Z. It outputs one or several solutions which will be correct if the Boolean function inputs can be matched to the correct inputs of one of the W, X, Y, Z. This mapping of 6 inputs to $abcdef$ is still in beta testing. The source code is included in Appendix D.

11 Invariants of Degree 8 with Complex and Secure Boolean Functions

In a similar way we have found invariants of degree 6 and 8 with an even more complex FE leading to an increased number of possibilities to solve it for the Boolean function Z and to an increased complexity for such solutions, and in general to high degree invariants such as possibly other lower degrees invariant do not exist (depending which solution Z for FE we select at the end). We give here two examples which were found by paper and pencil with help of computer simulations.

```
264: P=5,30,11,20,19,32,2,33,21,10,18,25,28,31,17,35,
16,13,6,36,3,15,23,1,14,27,7 D=20,28,24,36,16,8,12,4,32
265: P=1,20,33,34,15,13,27,6,10,23,21,25,16,14,2,4,3,
19,35,29,26,9,5,22,7,11,17 D=36,28,24,32,20,8,12,16,4
```

Then we define:

$$\begin{cases} A \overset{def}{=} (i+m) \\ B \overset{def}{=} (j+n) \\ C \overset{def}{=} (k+o) \\ D \overset{def}{=} (l+p) \\ E \overset{def}{=} (y+O) \\ F \overset{def}{=} (z+P) \\ G \overset{def}{=} (M+Q) \\ H \overset{def}{=} (N+R) \end{cases}$$

With these notations we have computed the FE for the following polynomial:

$$\mathcal{P} = ABCDEFGH$$

The fundamental equation is the same for both LZS 264 and 265 and it factors very neatly as:

$$FE = BCDFGH \cdot ((O+y+Y)W + (i+m)Y)$$

Now we can see the difference between 264 and 265. Key 264 was chosen so that our equation has a very simple degenerate solution. Key 265 was chosen so that it has the following solution:

$$ad+bc+be+de+ef+acd+acf+ade+bcf+bdf+cef+abcd+abce+abef+bcdf+abcde+acdef+1+a+e+c$$

which differs from the original Boolean function by only one linear term, the last letter was changed from f to c.

This attack also works for MANY other Boolean functions, and the space of the Boolean functions where this attack works is incredibly large. More information about this can be found in another paper [26].

12 Preliminary Conclusion

This completes our proof of concept for embedding a backdoor inside the T-310 cipher by solving the Fundamental Equation for Z. We are able to produce a variety of strong polynomial invariants which work for any number of rounds and for a cipher setup which is secure w.r.t. to all previously known attacks. In some cases, the ONLY difference between our cipher and the original government cipher is that the LZS is not standard (more frequently the Boolean function is also not standard). It is sufficient now to change[29] the LZS printed circuit board inside the cipher, in order to obtain a weak cipher with a non-linear property valid for an arbitrarily large number of rounds. We should add that no other cipher we heard of, uses such excessively large numbers of rounds as T-310 in the actual encryption process, leading to an excessively large hardware cost, cf. [25]. Our attack shows that in some cases, all this does not help (!).

Very few attacks in symmetric cryptanalysis work when the number of rounds is very large, for example slide attacks specifically, cf. for example [7] or previous attacks with algebraic/polynomial invariants [16, 17].

12.1 What is Wrong with Our Boolean Functions?

Nothing is wrong, as we will explain below. In most of our proof of concept examples the Boolean function is very special and not very strong or quite trivial, and has generally a lower degree than expected. This is however **not a problem** and is due to the fact that we impose yet very few constraints and our examples have been chosen for elegance and simplicity and are characterised by a short and sparse FE which is unlikely to be the case in general. An important point is that at this moment we have **artificially** imposed that many inputs of these Boolean functions must be eliminated and the function maybe does not really depend on these inputs [or depends on them in a somewhat degenerate or deficient way]. This inevitably leads to quite peculiar Boolean functions which for example do not depend on some bits. Such Boolean functions are not very convincing if our goal was to backdoor a block cipher. Then we expect that the Boolean function will be more severely constrained and will become unique and more complex and will use all the 6 variables, which is possible see example in Section 9.4. Three examples where the solution is already unique, but not yet sufficiently complex, are given in Sections 9.2 and 9.4 or 9.1. The complexity of the solutions increases with the degree of the FE also, see for example Section 10.8. In general, when the degree of $\mathcal{P}$ increases, we expect to find many more/stronger invariants $\mathcal{P}$ which work for a larger proportion of the space of all Boolean functions on 6 variables, and also many invariant which lead to unique solutions which are very peculiar and occur for strange reasons, thus leading to more convincing backdoors the very existence of which would not be at all obvious. At then end our problem is solved in Section 11 where the Boolean function we use differs only by one linear term from the original Boolean function used during the Cold War. Further study and analysis of this attack can be found in [26].

[29] Such an upgrade would typically be done once per year, cf. [21].

12.2 Open Problems

Therefore we are asking the following questions for future research to answer.

1. Can non-trivial[30] non-linear invariants be constructed or found for any of the original historical keys such as KT1 keys for example LZS 26?
2. In particular, the Boolean function could be extremely weak, it is first a question of a proof of concept and if the KT1 conditions and these particular LZS do not prevent our attacks (making them totally impossible). One partial solution is provided in Section 10.6 and another in Appendix B.
3. How can an LZS with a polynomial invariant be constructed on demand for any[31] Boolean function not chosen by ourselves? For example such as a random Boolean function? Or just the original Boolean function used in T-310?
4. In addition is it possible to find or construct an **isolated and rare** invariant such that it will work with an extremely low probability say less than 2^{-25} for another LZS, and that it will with an extremely low probability say less than 2^{-25} for another Boolean function with similar characteristics?

12.3 New Public Key Cryptosystems?

Another open problem is whether it is sensible to build a **new public key cryptosystem** in the following way:

1. We specify an efficient algorithm for finding a random triple $(LZS, Z, \mathcal{P})$, (very much like in this paper, however more work on what is the best way of doing this is needed),
2. However if we know only the LZS and the Boolean function Z, which pair would be the public key, and we do not know how they have been constructed, it should computationally infeasible to find the invariant $\mathcal{P}$ (which would be the private key).

We should note that the space for $\mathcal{P}$ is double exponential size w.r.t. block cipher size n. No public cryptosystem with such property was yet invented. So we are looking at a very interesting question. Such a cryptosystem could be quantum-resistant just because of this complexity (large size of the private key). In practice, for our public key cryptosystem to be secure it would be sufficient that the discovery of $\mathcal{P}$ has time complexity which is not polynomial. There is therefore a big gap and this in our favour, i.e. even if $\mathcal{P}$ has numerous properties which makes it discovery easier, their discovery could still be completely intractable in theory or in practice.

[30] For example such that $\mathcal{P}$ is an irreducible polynomial and no linear invariants exist, on both accounts not degenerate and different than what we see in Appendix A.2.

[31] The answer is probably yes and this problem is almost already solved in Section 11 where the Boolean function we use differs only by one letter compared to the original historical Boolean function.

13 Conclusion

One of the major open problems in block cipher cryptanalysis is to discover new **specific** types of invariant properties which can hold for a larger number of rounds. In this paper we study non-linear Boolean polynomial invariants. Previous researchers have had great difficulties to make this approach work and the space of the possibilities is too large for systematic exploration. In this paper we have turned the problem of non-linear cryptanalysis upside-down. We fix the combinatorial structure of the cipher, look at the round ANF formulas, and try to find a Boolean function Z and also simultaneously determine a suitable polynomial invariant $\mathcal{P}$, which allows to achieve the desired property. This is done by solving the so called Fundamental Equation (FE) or several such equations combined. Such invariants have the capacity to avoid more complex parts of the cipher and also many key bits. Then, we encounter a major problem: several FE equations need to have a common solution. This can eventually be achieved(!), cf. Sect. 7.1 and 7.4. Iterative attacks are also possible, cf. Sect. 4.11 and 7.3.

This paper shows how a specific structure and internal wiring of a block cipher can be translated into a relatively simple FE equation, which can be used to study which specific non-linear invariants may exist (or not) for this cipher. Stronger invariants can now be defined and characterized algebraically, $\mathcal{P}$ must be such that most or all the coefficients of FE reduce to zero, see Appendix A. Our main contribution is to show that the attacker does not need to randomly search for a vulnerable non-linear component Z (and for $\mathcal{P}$). Weak Boolean functions and specific polynomial invariants $\mathcal{P}$ can be **determined** – by solving our FE equation(s). Our approach is constructive, completely general and can be applied to almost any block cipher: we directly write the Fundamental Equation from the ANFs, substitute variables inside, and attempt to solve our FE(s).

Future research. We anticipate that the success rate of this approach will be very different for different families of ciphers. If just one round function is very complex and uses many key bits, with too many constraints to satisfy simultaneously, our approach is likely to fail, or solving FE will become difficult.

Positive Results. In current research on backdoors in block ciphers there are many impossibility results [4, 9] but extremely few possibility results [16, 14]. Partitioning cryptanalysis [36] properties can be quite obscure, (weak ciphers seem to occur accidentally, and complex ciphers seem secure for no reason [47]). Polynomial invariants are way more intelligible. We discover that weak ciphers follow clear rules and a whole range from simple to increasingly complex invariant properties can now be characterized, studied and computed explicitly.

On Our Specific Cipher. What is incredible is that this approach works, **at all**, for at least one real-life block cipher T-310. Several factors help to make this happen: extremely few key and IV bits are used in one round, there is some freedom in the choice of the internal wiring with a strong triangular structure, and the degree of the Boolean polynomials is limited to 6. In contrast the space of possible polynomials Z and $\mathcal{P}$ is extremely large. In future research we expect to show that the proportion of keys for which FE has a solution can be computed exactly cf. [24], and that it is strictly increasing as the degree of $\mathcal{P}$ grows.

References

1. Ange Albertini, Jean-Philippe Aumasson, Maria Eichlseder, Florian Mendel and Martin Schläffer: Malicious Hashing: Eve's Variant of SHA-1 in SAC 2014, pp.1-19, Springer LNCS 8781.
2. R. Aragona, A. Caranti, M. Sala: *The group generated by the round functions of a GOST-like cipher, Ann. Mat. Pura Appl., 196 (2016), 1–17.*
3. *Arnaud Bannier, Nicolas Bodin, and Eric Filiol: Partition-Based Trapdoor Ciphers,* `https://ia.cr/2016/493`
4. C. Beierle, A. Canteaut, G. Leander, Y. Rotella: *Proving resistance against invariant attacks: how to choose the round constants,* in Crypto 2017, Part II. LNCS, 10402, pp. 647–678, Springer 2017.
5. C. Berbain, H. Gilbert, and J. Patarin: *QUAD: A Practical Stream Cipher with Provable Security,* In Eurocrypt 2005, LCNS, Springer, 2005.
6. Nicolas T. Courtois, Theodosis Mourouzis, Michał Misztal, Jean-Jacques Quisquater, Guangyan Song: *Can GOST Be Made Secure Against Differential Cryptanalysis?,* In Cryptologia, vol. 39, Iss. 2, 2015, pp. 145-156.
7. Nicolas Courtois, Gregory V. Bard, David Wagner: *Algebraic and Slide Attacks on KeeLoq,* In FSE 2008, pp. 97-115, LNCS 5086, Springer, 2008.
8. Nicolas T. Courtois, Iason Papapanagiotakis-Bousy, Pouyan Sepehrdad and Guangyan Song: *Predicting Outcomes of ElimLin Attack on Lightweight Block Cipher Simon,* In proc. of Secrypt 2016, `http://discovery.ucl.ac.uk/1521419/1/Courtois_SECRYPT_2016_104.pdf`
9. Marco Calderini: *A note on some algebraic trapdoors for block ciphers,* last revised 17 May 2018, `https://arxiv.org/abs/1705.08151`
10. Arthur Cayley: *On the Theory of Linear Transformations,* In Cambridge Math. J. 4, 193-209, 1845.
11. Tony Crilly: *The rise of Cayley's invariant theory (1841–1862),* In Historia Mathematica, Vol. 13, Iss. 3, August 1986, pp. 241-254 `https://doi.org/10.1016/0315-0860(86)90091-1`
12. Jean A. Dieudonné, James B. Carrell (1970), *Invariant theory, old and new,* Advances in Mathematics, reprinted in Advances in Mathematics, Boston, MA, Academic Press, 4: 1–80, ISBN 978-0-12-215540-6.
13. Nicolas Courtois, Gregory V. Bard: *Algebraic Cryptanalysis of the Data Encryption Standard,* In Cryptography and Coding, 11^{th} IMA Conference, pp. 152-169, LNCS 4887, Springer, 2007.
14. Marco Calderini: *On Boolean functions, symmetric cryptography and algebraic coding theory,* Ph.D. in Mathematics, Supervisor: Prof. Massimiliano Sala, University of Trento, Italy, April 2015
15. Nicolas Courtois: *Feistel Schemes and Bi-Linear Cryptanalysis,* in Crypto 2004, LNCS 3152, pp. 23-40, Springer, 2004.
16. Nicolas Courtois: *The Inverse S-box, Non-linear Polynomial Relations and Cryptanalysis of Block Ciphers,* in AES 4 Conference, Bonn May 10-12 2004, LNCS 3373, pp. 170-188, Springer, 2005. Extended version: `https://ia.cr/2005/251.pdf`
17. Nicolas Courtois: *The Inverse S-box and Two Paradoxes of Whitening,* Long extended version of the Crypto 2004 rump session presentation, *Whitening the AES S-box,* Available at `http://www.minrank.org/invglc_rump_c04.zip`. Main theorem appears in Appendix B of the extended version of [16].
18. Nicolas T. Courtois; *Invariant Hopping Attacks on Block Ciphers,* accepted at WCC'2019, Abbaye de Saint-Jacut de la Mer, France, 31 March – 5 April 2019.

19. Nicolas Courtois: *Data Encryption Standard (DES),* slides used in GA03 Introduction to Cryptography and later in GA18 course Cryptanalysis taught at University College London, 2006-2016, `http://www.nicolascourtois.com/papers/des_course_6.pdf`
20. Nicolas Courtois and Willi Meier: *Algebraic Attacks on Stream Ciphers with Linear Feedback,* Eurocrypt 2003, Warsaw, Poland, LNCS 2656, pp. 345-359, Springer. A long, extended version of this paper is available from `www.nicolascourtois.com`.
21. Nicolas T. Courtois, Klaus Schmeh, Jörg Drobick, Jacques Patarin, Maria-Bristena Oprisanu, Matteo Scarlata, Om Bhallamudi: *Cryptographic Security Analysis of T-310,* Monography study on the T-310 block cipher, 132 pages, received 20 May 2017, last revised 29 June 2018, `https://ia.cr/2017/440.pdf`
22. Nicolas T. Courtois, Maria-Bristena Oprisanu: *Ciphertext-only attacks and weak long-term keys in T-310,* in Cryptologia, vol 42, iss. 4, pp. 316-336, May 2018. `http://www.tandfonline.com/doi/full/10.1080/01611194.2017.1362065`.
23. Nicolas Courtois, Marios Georgiou and Matteo Scarlata: *Slide Attacks and LC-Weak Keys in T-310,* Accepted for publication, will appear in Cryptologia in 2019.
24. Nicolas Courtois, Maria-Bristena Oprisanu and Klaus Schmeh: *Linear cryptanalysis and block cipher design in East Germany in the 1970s,* will appear in Cryptologia in 2018.
25. Nicolas Courtois, Jörg Drobick and Klaus Schmeh: *Feistel ciphers in East Germany in the communist era,* In Cryptologia, vol. 42, Iss. 6, 2018, pp. 427–444.
26. Nicolas T. Courtois: *Structural Nonlinear Invariant Attacks on T-310: Attacking Arbitrary Boolean Functions,* `https://ia.cr/2018/1242.pdf`, received 28 Dec 2018.
27. Nicolas T. Courtois, Marios Georgiou: *Constructive Non-Linear Polynomial Cryptanalysis of a Historical Block Cipher,* At `http://arxiv.org/abs/1902.02748`.
28. Nicolas T. Courtois: *New Frontier in Symmetric Cryptanalysis,* Invited talk at Indocrypt 2008, 14-17 December 2008. Extended version of slides presented: `http://www.nicolascourtois.com/papers/ front_indocrypt08.pdf`.
29. Jörg Drobick: *T-310/50 ARGON,* a web page about T-310 cipher machines `http://scz.bplaced.net/t310.html`
30. Jörg Drobick: *T-310 Schlüsselunterlagen,* a web page which enumerates several different known long-term keys for T-310 from 1973-1990, consulted 21 January 2017, `http://scz.bplaced.net/t310-schluessel.html`
31. H. Feistel, W.A. Notz, J.L. Smith, *Cryptographic Techniques for Machine to Machine Data Communications,* Dec. 27, 1971, Report RC-3663, IBM T.J.Watson Research.
32. Marios Georgiou: *Weak Keys and Cryptanalysis of a Cold War Block Cipher,* Master thesis, MSc Information Security, supervised by Nicolas T. Courtois, University College London, Computer Science Department, 2018, extended version, 19 Jan 2019, `https://arxiv.org/pdf/1901.06504.pdf`
33. Carlo Harpes: *Partitioning Cryptanalysis,* Post-Diploma Thesis, Signal and Information Processing Lab., Swiss Federal Institute of Technology, Zurich, March 1995.
34. C. Harpes, G. Kramer, and J. Massey: *A Generalization of Linear Cryptanalysis and the Applicability of Matsui's Piling-up Lemma,* Eurocrypt'95, LNCS 921, Springer, pp. 24-38.
35. Carlo Harpes: *Cryptanalysis of iterated block ciphers,* PhD thesis, No 11625, Swiss Federal Int. of Tech., ETH Series in Information Processing, Ed. J. L. Massey, Hartung-Gorre Verlag Konstanz, 1996, ISBN 3-89649-079-6, ISSN 0942-3044.

36. C. Harpes, J. L. Massey: *Partitioning cryptanalysis,* In FSE 97, LNCS 1267, pp. 13–27, 1997.
37. Thomas Jakobsen, Carlo Harpes: *Non-Uniformity Measures for Generalized Linear Cryptanalysis and Partitioning Cryptanalysis,* in Pragocrypt'96, 1996.
38. Thomas Jakobsen: *Higher-Order Cryptanalysis of Block Ciphers.* Ph.D. thesis, Dept. of Math., Technical University of Denmark, 1999.
39. Lars R. Knudsen, Matthew J. B. Robshaw: *Non-Linear Characteristics in Linear Cryptoanalysis,* Eurocrypt'96, LNCS 1070, Springer, pp. 224-236, 1996.
40. Mitsuru Matsui: *Linear Cryptanalysis Method for DES Cipher,* Eurocrypt'93, LNCS 765, Springer, pp. 386-397, 1993.
41. Pawel Morawiecki: *Malicious Keccak,* `https://ia.cr/2015/1085`
42. Jacques Patarin, Valérie Nachef, Côme Berbain: Generic Attacks on Unbalanced Feistel Schemes with Contracting Functions, in Asiacrypt 2006, pp. 396-411, LNCS 4284, Springer 2006.
43. Kenneth G. Paterson: *Imprimitive Permutation Groups and Trapdoors in Iterated Block Ciphers,* In FSE 1999, pp. 201-214, Springer 1999.
44. Petr Susil, Pouyan Sepehrdad, Serge Vaudenay, Nicolas Courtois: *On selection of samples in algebraic attacks and a new technique to find hidden low degree equations.* In International Journal of Information Security vol. 15 iss. 1, pp. 51-65, Springer, 2016.
45. Referat 11: *Kryptologische Analyse des Chiffriergerätes T-310/50. Central Cipher Organ, Ministry of State Security of the GDR, document referenced as 'ZCO 402/80', a.k.a. MfS-Abt-XI-594, 123 pages, Berlin, 1980.*
46. Ralph Wernsdorf: The one-round functions of the DES generate the alternating group, In proc. of EUROCRYPT'92, LNCS 658, pp. 99–112, Springer, 1993.
47. R. Sparr and R. Wernsdorf: *Group theoretic properties of Rijndael-like ciphers,* In Discrete Appl. Math, 156 (2008), 3139–3149.
48. Klaus Schmeh: *The East German Encryption Machine T-310 and the Algorithm It Used,* In Cryptologia, vol. 30, iss. 3, pp. 251–257, 2006.
49. Yosuke Todo, Gregor Leander, and Yu Sasaki: *Nonlinear invariant attack: Practical attack on full SCREAM, iSCREAM and Midori64,* In Journal of Cryptology, pp 1–40, April 2018.

A FE with Reduction to Zero and Structural Invariants

In this paper we show that a Boolean function can be chosen in order to create a specific invariant. An interesting question is whether the Fundamental Equation can be reduced to zero (!), i.e. given a certain fixed $\mathcal{P}$ it is simply equal to zero and it holds for any Z. When this happens, we will obtain a polynomial invariant which **works for any Z** and therefore it also works also for the original Boolean function used in the T-310 cipher in 1970s-1990. Here below we give one example when this happens. In this example a multiple FE equation can be solved for 8 cases simultaneously, because all the 8 equations are empty. We define key 898 by:

```
898: P=35,17,19,25,17,25,22,19,31,30,29,26,20,36,23,5,
2,27,16,11,28,33,7,15,21,12,3 D=33,23,27,26,28,24,32,28,36
```

A computer simulation shows that the following irreducible polynomial with 43 terms is an invariant in this case:

$$eg+fh+gi+hi+ej+hj+ij+ek+fk+jk+fl+il+jl+hm+km+lm+en+in+fo+io+jo+$$

$$mo+gp+jp+kp+np+hq+kq+lq+oq+er+ir+pr+fs+is+js+ms+qs+gt+jt+kt+nt+rt$$

and a quick computation shows that the FE is indeed reduced to zero. This invariant is more complex than previously, it involves Z1, Z2 and a larger number of variables. It works for T-310 for any key, any IV and for an arbitrary number of rounds. In this example a linear invariant also exists: $e + f + g + h + l + m + n + o + p + q + r + s + t$ and both linear and non-linear invariants can be characterized (and computed!) from the fact that our FE reduces to zero. Both linear and non-linear invariants exist here and penetrate deeply inside the cipher. Now because the FE reduces to zero, **no Boolean function can make this cipher setup secure** against round invariant attacks (and also against partitioning attacks in general).

A.1 Another Example

In this example, the key is a bijection on 36 bits, it uses Z1,Z2 and Z4, and we have again an invariant with FE reduced to 0 as follows:

```
848: P=18,32,2,15,34,30,33,28,13,9,21,27,25,24,22,20,10,
5,26,31,11,4,35,16,7,23,36 D=4,36,32,16,20,12,8,28,24
```

the invariant is:

$$\mathcal{P} = gh+hi+gj+ij+hk+jk+gl+il+kl+hO+jO+lO+gP+iP+kP+OP+hQ+jQ+lQ+$$

$$PQ+gR+iR+kR+OR+QR+hS+jS+lS+PS+RS+gT+iT+kT+OT+QT+ST+$$

$$hU + jU + lU + PU + RU + TU + gV + iV + kV + OV + QV + SV + UV$$

This invariant works in all 8 cases.

A.2 Example with a KT1 Key

Let $K = 0$. A similar type of invariant can be achieved with a KT1 key.

```
881: P=4,20,33,8,1,28,5,19,9,32,11,17,24,13,21,18,15,
25,12,16,35,22,23,29,36,30,34 D=0,36,4,8,12,20,24,28,32
```

```
ab+bm+an+mn+bo+no+ap+mp+op+bq+nq+pq+ar+mr+or+qr+bs+ns+ps+rs+at+mt+ot+qt+st+bu+nu+pu+ru+tu+av+mv+ov+qv+sv+uv+bw+nw+pw+rw+tw+vw+ax+mx+
ox+qx+sx+ux+wx+by+ny+py+ry+ty+vy+xy+az+mz+oz+qz+sz+uz+wz+yz+bM+nM+pM+rM+tM+vM+xM+zM+aN+mN+oN+qN+sN+uN+wN+yN+MN+bO+nO+pO+rO+tO+vO+xO+
zO+NO+aP+mP+oP+qP+sP+uP+wP+yP+MP+OP+bQ+nQ+pQ+rQ+tQ+vQ+xQ+zQ+NQ+PQ+aR+mR+oR+qR+sR+uR+wR+yR+MR+OR+QR+bS+nS+pS+rS+tS+vS+xS+zS+NS+PS+RS+
aT+mT+oT+qT+sT+uT+wT+yT+MT+OT+QT+ST+bU+nU+pU+rU+tU+vU+xU+zU+NU+PU+RU+TU+aV+mV+oV+qV+sV+uV+wV+yV+MV+OV+QV+SV+UV
```

which has $169 = 13^2$ terms and is equal to

$$(n+b+p+r+t+v+x+z+N+P+R+T+V)(a+m+o+q+s+u+w+y+M+O+Q+S+U)$$

This invariant works only in 4 cases. It only works when $K = 0$. It remains an open problem to find an invariant which works in 8 cases for a KT1 key.

Remark. In all the keys where FE reduces to zero known to us there is always also a linear invariant. Furthermore it is possible to verify that if we call $\mathcal{A}$ the first sum, and if we call $\mathcal{B}$ the second sum, $\mathcal{AB}$ is a non-linear invariant and also $\mathcal{A} + \mathcal{B}$ is a linear invariant for the same cipher setup LZS 881. We do not claim that using this exact invariant gives anything not already found in classical Linear Cryptanalysis. These are degenerate examples.

In Appendix B we present a yet more complex and more elaborate example with a KT1 key which remains entirely degenerated: everything is essentially a consequence of a linear attack, which however is not a random linear attack, it is linear attack of a certain particular shape.

A.3 More Degenerate Examples

Further degenerate examples will be linear combinations of several degenerate examples such as above. In this case the polynomial $\mathcal{P}$ could be irreducible (!). A detailed study shows that up to 13 linear properties can be made to hold simultaneously in T-310, cf. Section 21.16 in [21] and up to 10 can be obtained for KT1 keys. This leads to plenty of non-linear invariants which can be constructed from linear invariants (only if not using $F, S1, S2$ bits) which can be multiplied or added together. Moreover, in some combinations of invariants with $F, S1, S2$ bits, these bits could be eliminated leading to somewhat less trivial types of degenerate invariants.

A.4 Some More Concrete Degenerate Examples

We discover some of LC-weak keys from [21] which do not use F, K, L also have interesting non-linear invariants. In most cases this does not work, however sometimes it does. We give some examples below:

```
606 and 607:
e+f+g+h+i+j+k+l+u+v+w+x+y+z+M+N
eg+fh+gi+hj+ek+ik+fl+jl+gu+ku+hv+lv+ew+iw+uw+fx+jx+vx+gy+ky+wy+hz+lz+xz+eM+iM+uM+yM+fN+jN+vN+zN
ef+fg+eh+gh+fi+hi+ej+gj+ij+fk+hk+jk+el+gl+il+kl+fu+hu+ju+lu+ev+gv+iv+kv+uv+fw+hw+jw+lw+vw+ex+gx+ix+kx+ux+wx+fy+
hy+jy+ly+vy+xy+ez+gz+iz+kz+uz+wz+yz+fM+hM+jM+lM+vM+xM+zM+eN+gN+iN+kN+uN+wN+yN+MN
efg+efh+efk+efl+efw+efx+efM+efN+egh+egj+egl+egv+egx+egz+egN+fgh+fgi+fgl+fgu+fgx+fgy+fgN+ehj+ehk+ehv+ehw+ehz+ehM+
fhi+fhk+fhu+fhw+fhy+fhM+ghi+ghj+ghu+ghv+ghy+ghz+fik+fil+fiw+fix+fiM+fiN+gij+gil+giv+gix+giz+giN+hij+hik+hiv+hiw+
hiz+hiM+ejk+ejl+ejw+ejx+ejM+ejN+gjl+gju+gjx+gjy+gjN+hjk+hju+hjw+hjy+hjM+ijk+ijl+ijw+ijx+ijM+ijN+ekl+ekv+ekx+ekz+
ekN+fkl+fku+fkx+fky+fkN+hku+hkv+hky+hkz+ikl+ikv+ikx+ikz+ikN+jkl+jku+jkx+jky+jkN+elv+elw+elz+elM+flu+flw+fly+flM+
glu+glv+gly+glz+ilv+ilw+ilz+ilM+jlu+jlw+jly+jlM+klu+klv+kly+klz+fuw+fux+fuM+fuN+guv+gux+guz+guN+huv+huw+huz+huM+
juw+jux+juM+juN+kuv+kux+kuz+kuN+luv+luw+luz+luM+evw+evx+evM+evN+gvx+gvy+gvN+hvw+hvy+hvM+ivw+ivx+ivM+ivN+kvx+kvy+
kvN+lvw+lvy+lvM+uvw+uvx+uvM+uvN+ewx+ewz+ewN+fwx+fwy+fwN+hwy+hwz+iwx+iwz+iwN+jwx+jwy+jwN+lwy+lwz+uwx+uwz+uwN+vwx+
vwy+vwN+exz+exM+fxy+fxM+gxy+gxz+ixz+ixM+jxy+jxM+kxy+kxz+uxz+uxM+vxy+vxM+wxy+wxz+fyM+fyN+gyz+gyN+hyz+hyM+jyM+jyN+
kyz+kyN+lyz+lyM+vyM+vyN+wyz+wyN+xyz+xyM+ezM+ezN+gzN+hzM+izM+izN+kzN+lzM+uzM+uzN+wzN+xzM+yzM+yzN+eMN+fMN+iMN+jMN+
uMN+vMN+yMN+zMN

617:
c+e+f+g+h+i+j+k+l+O+P+Q+R+S+T+U+V
a+b+ac+bc+ae+be+ce+af+bf+ef+ag+bg+cg+fg+ah+bh+eh+gh+ai+bi+ci+fi+hi+aj+bj+ej+gj+ij+ak+bk+ck+fk+hk+jk+al+bl+el+gl+il+kl+
  n+cn+en+fn+gn+hn+in+jn+kn+ln+o+co+eo+fo+go+ho+io+jo+ko+lo+p+cp+ep+fp+gp+hp+ip+jp+kp+lp+q+cq+eq+fq+gq+hq+iq+jq+kq+lq+
  r+cr+er+fr+gr+hr+ir+jr+kr+lr+s+cs+es+fs+gs+hs+is+js+ks+ls+t+ct+et+ft+gt+ht+it+jt+kt+lt+aO+bO+cO+fO+hO+jO+lO+nO+oO+pO+
  qO+rO+sO+tO+aP+bP+eP+gP+iP+kP+nP+oP+pP+qP+rP+sP+tP+OP+aQ+bQ+cQ+fQ+hQ+jQ+lQ+nQ+oQ+pQ+qQ+rQ+sQ+tQ+PQ+aR+bR+eR+gR+iR+kR+
  nR+oR+pR+qR+rR+sR+tR+OR+QR+aS+bS+cS+fS+hS+jS+lS+nS+oS+pS+qS+rS+sS+tS+PS+RS+aT+bT+eT+gT+iT+kT+nT+oT+pT+qT+rT+sT+tT+OT+
  QT+ST+aU+bU+cU+fU+hU+jU+lU+nU+oU+pU+qU+rU+sU+tU+PU+RU+TU+aV+bV+eV+gV+iV+kV+nV+oV+pV+qV+rV+sV+tV+OV+QV+SV+UV
```

In all these cases FE is reduced to 0. However it is not obvious to see if the non-linear invariants observed are always a simple consequence of simpler linear invariants. For key 606 and 607 the second longer quadratic polynomial for example is divisible by two linear polynomials $N + f + h + j + l + v + x + z$ and $M + e + g + i + k + u + w + y$ in the same way as in Section A.2. Also the polynomial of degree 3 has a linear factor. However the first quadratic polynomial $eg + fh + \ldots + vN + zN$ is irreducible and has no linear factors whatsoever(!).

B An Elaborate Example of A Trivial Invariant Attack Based On Linear Properties for a KT1 Key

KT1 keys are expected to be more secure than other keys and many KT1 keys were used to encrypt real-life government communications, see [22, 21]. More recent research shows that some percentage of these keys however are quite weak, see [23, 24]. The study of KT1 keys is not easy due to the very substantial complexity of the specification which defines these keys, see [45, 21, 22]. Here is yet another example how KT1 keys may fail us badly. An open problem until recently (and inside the previous update of this paper) was if any KT1 keys could be broken by a non-linear invariant. This problem is now solved here below with LZS 113 which has several distinct and independent invariants with 16 variables. With this LZS the FE reduces to 0 and invariants work for any Boolean function.

```
113: D=0,4,28,12,8,20,32,36,24,P=28,3,33,17,31,36,5,13,9,2,26,14,32,11,21,8,4,25,24,16,23,1,15,29,20,12,30
R1=1+4 D4=0 at degree 4
efgh+efgl+efgx+efgN+efhk+efhw+efhM+efkl+efkx+efkN+eflw+eflM+efwx+efwN+efxM+efMN+eghj+eghv+eghz+egjl+egjx+egjN+eglv+eglz+egvx+egvN+egxz+
egzN+ehjk+ehjw+ehjM+ehkv+ehkz+ehvw+ehvM+ehwz+ehzM+ejkl+ejkx+ejkN+ejlw+ejlM+ejwx+ejwN+ejxM+ejMN+eklv+eklz+ekvx+ekvN+ekxz+ekzN+elvw+elvM+
elwz+elzM+evwx+evwN+evxM+evMN+ewxz+ewzN+exzM+ezMN+fghi+fghu+fghy+fgil+fgix+fgiN+fglu+fgly+fgux+fguN+fgxy+fgyN+fhik+fhiw+fhiM+fhku+fhky+
fhuw+fhuM+fhwy+fhyM+fikl+fikx+fikN+filw+filM+fiwx+fiwN+fixM+fiMN+fklu+fkly+fkux+fkuN+fkxy+fkyN+fluw+fluM+flwy+flyM+fuwx+fuwN+fuxM+fuMN+
fwxy+fwyN+fxyM+fyMN+ghij+ghiv+ghiz+ghju+ghjy+ghuv+ghuz+ghvy+ghyz+gijl+gijx+gijN+gilv+gilz+givx+givN+gixz+gizN+gjlu+gjly+gjux+gjuN+gjxy+
gjyN+gluv+gluz+glvy+glyz+guvx+guvN+guxz+guzN+gvxy+gvyN+gxyz+gyzN+hijk+hijw+hijM+hikv+hikz+hivw+hivM+hiwz+hizM+hjku+hjky+hjuw+hjuM+hjwy+
hjyM+hkuv+hkuz+hkvy+hkyz+huvw+huvM+huwz+huzM+hvwy+hvyM+hwyz+hyzM+ijkl+ijkx+ijkN+ijlw+ijlM+ijwx+ijwN+ijxM+ijMN+iklv+iklz+ikvx+ikvN+ikxz+
ikzN+ilvw+ilvM+ilwz+ilzM+ivwx+ivwN+ivxM+ivMN+iwxz+iwzN+ixzM+izMN+jklu+jkly+jkux+jkuN+jkxy+jkyN+jluw+jluM+jlwy+jlyM+juwx+juwN+juxM+juMN+
jwxy+jwyN+jxyM+jyMN+kluv+kluz+klvy+klyz+kuvx+kuvN+kuxz+kuzN+kvxy+kvyN+kxyz+kyzN+luvw+luvM+luwz+luzM+lvwy+lvyM+lwyz+lyzM+uvwx+uvwN+uvxM+
uvMN+uwxz+uwzN+uxzM+uzMN+vwxy+vwyN+vxyM+vyMN+wxyz+wyzN+xyzM+yzMN
R1=1+3 D4=0 at degree 3
efg+efh+egh+fgh+fgi+fhi+ghi+egj+ehj+ghj+gij+hij+efk+ehk+fhk+fik+hik+ejk+hjk+ijk+efl+egl+fgl+fil+gil+ejl+gjl+ijl+ekl+fkl+ikl+jkl+fgu+fhu+
ghu+gju+hju+fku+hku+jku+flu+glu+jlu+klu+egv+ehv+ghv+giv+hiv+ekv+hkv+ikv+elv+glv+ilv+klv+guv+huv+kuv+luv+efw+ehw+fhw+fiw+hiw+ejw+hjw+ijw+
elw+flw+ilw+jlw+fuw+huw+juw+luw+evw+hvw+ivw+lvw+uvw+efx+egx+fgx+fix+gix+ejx+gjx+ijx+ekx+fkx+ikx+jkx+fux+gux+jux+kux+evx+gvx+ivx+kvx+uvx+
ewx+fwx+iwx+jwx+uwx+vwx+fgy+fhy+ghy+gjy+hjy+fky+hky+jky+fly+gly+jly+kly+gvy+hvy+kvy+lvy+fwy+hwy+jwy+lwy+vwy+fxy+gxy+jxy+kxy+vxy+wxy+egz+
ehz+ghz+giz+hiz+ekz+hkz+ikz+elz+glz+ilz+klz+guz+huz+kuz+luz+ewz+hwz+iwz+lwz+uwz+exz+gxz+ixz+kxz+uxz+wxz+gyz+hyz+kyz+lyz+wyz+xyz+efM+ehM+
fhM+fiM+hiM+ejM+hjM+ijM+elM+flM+ilM+jlM+fuM+huM+juM+luM+evM+hvM+ivM+lvM+uvM+exM+fxM+ixM+jxM+uxM+vxM+fyM+hyM+jyM+lyM+vyM+xyM+ezM+hzM+izM+
lzM+uzM+xzM+yzM+efN+egN+fgN+fiN+giN+ejN+gjN+ijN+ekN+fkN+ikN+jkN+fuN+guN+juN+kuN+evN+gvN+ivN+kvN+uvN+ewN+fwN+iwN+jwN+uwN+vwN+fyN+gyN+jyN+
kyN+vyN+wyN+ezN+gzN+izN+kzN+uzN+wzN+yzN+eMN+fMN+iMN+jMN+uMN+vMN+yMN+zMN
R1=1+2 D4=0 at degree 2
ef+fg+eh+gh+fi+hi+ej+gj+ij+fk+hk+jk+el+gl+il+kl+fu+hu+ju+lu+ev+gv+iv+kv+uv+fw+hw+jw+lw+vw+ex+gx+ix+kx+ux+wx+fy+hy+jy+ly+vy+xy+ez+gz+iz+
kz+uz+wz+yz+fM+hM+jM+lM+vM+xM+zM+eN+gN+iN+kN+uN+wN+yN+MN
eg+fh+gi+hj+ek+ik+fl+jl+gu+ku+hv+lv+ew+iw+uw+fx+jx+vx+gy+ky+wy+hz+lz+xz+eM+iM+uM+yM+fN+jN+vN+zN
e+f+g+h+i+j+k+l+u+v+w+x+y+z+M+N
```

For the same key we found 5 homogenous non-linear invariants of degrees 1,2,2,3,4. We found no additional invariants at any higher degree. Not all invariants can be easily found with limited computing power.

Remark. The presence of invariants of degree 1 is a bit disappointing and better examples exist. In Section 10.6 we have seen another example of a weak KT1 key without any linear invariants, and with degrees 2,4 only. then the invariant of degree 2 can further be removed (!). However in this example with LZS 551/558/550 above cf. Section 10.6, the attack works only for a special Boolean function and has 8 variables. The invariant(s) in this section are more powerful and more complex. They use 16 variables and they work for ANY Boolean function.

B.1 Detailed Analysis of Our Invariant Attack with KT1 Key 113

Let I4, I3, I2a and I2b and I1 be the 5 invariants above. In order to understand better what is going on here, we observe that the unique invariant of degree 4 can be factored as a product of 4 linear polynomials and all the 5 polynomials of degrees 4,3,2,2,1 are closely related as follows:

$$\begin{cases} A \overset{def}{=} (e+i+u+y) \\ B \overset{def}{=} (f+j+v+z) \\ C \overset{def}{=} (g+k+w+M) \\ D \overset{def}{=} (h+l+x+N) \end{cases}$$

With these notations we can reveal that all the 5 invariants are linearly independent[32] but algebraically dependent and they are related as follows:

$$\begin{cases} I4 = A \cdot B \cdot C \cdot D \\ I3 = A \cdot B \cdot C + A \cdot B \cdot D + A \cdot C \cdot D + B \cdot C \cdot D \\ I2a = (A+C) \cdot (B+D) \\ I2b = A \cdot C + B \cdot D \\ I1 = A+B+C+D \end{cases}$$

Remark. We remark that NOT all symmetric polynomials in A, B, C, D are invariants for our cipher, only some are.

B.2 Invariants for More Than One Round

Moreover if we look at invariants which work for 4 rounds of encryption and not for 1 round, we found that $D \to C \to B \to A \to D$ is an invariant for linear cryptanalysis with a period of 4 rounds. Then furthermore we found the following four cubic invariants which work for 2 rounds but NOT for 1 round. These invariants use all the 16 bits:

$$\begin{cases} A \cdot B \cdot C + A \cdot C \cdot D \\ A \cdot B \cdot D + B \cdot C \cdot D \end{cases}$$

and for 4 rounds we have four invariants using 12 bits only each:

$$\begin{cases} A \cdot B \cdot C \\ A \cdot B \cdot D \\ A \cdot C \cdot D \\ B \cdot C \cdot D \end{cases}$$

The sum of these invariants is $I3$ which we have seen before.

B.3 Insights and Internal Operation For Key 113

We show how the non-trivial substitutions which are involved in our invariants look like:

[32] They are also essentially independent in terms of space partitions they induce which are NOT distinct.

$$\begin{aligned}
&[\ldots]\\
e &\leftarrow f\\
f &\leftarrow g\\
g &\leftarrow h\\
h &\leftarrow (F+Z+a)\\
Z &\leftarrow Z(L,i,T,d,t,f))\\
i &\leftarrow j\\
j &\leftarrow k\\
k &\leftarrow l\\
l &\leftarrow (F+Z+a)+e\\
&[\ldots]
\end{aligned}$$

$$\begin{aligned}
&[\ldots]\\
u &\leftarrow v\\
v &\leftarrow w\\
w &\leftarrow x\\
x &\leftarrow (F+Z+a)+(Y+e+L+X)+y\\
X &\leftarrow Z(z,p,O,S,l,m)\\
y &\leftarrow z\\
z &\leftarrow M\\
M &\leftarrow N\\
N &\leftarrow (F+Z+a)+(Y+e+L+X)+u+i\\
&[\ldots]
\end{aligned}$$

The key point about the invariants we have found is that they connect two remote part of the cipher (similar as in Section 8.2) and that all the other variables get eliminated. In most of our invariants F and L are eliminated. Here it is rather $(F+Z+a)$ and $(Y+e+L+X)$ get eliminated, which is again just two variables which however depend on a substantially larger number of bits inside the cipher. All this makes that 20 out of 36 variables are not even used in out invariant yet the invariant does not depend on the Boolean function (with FE reduced to 0).

C Analysis of Polynomial FE Spaces and Assessment of Various Keys

An interesting question is whether some keys are fundamentally more resistant against non-linear cryptanalysis than other keys. This is what we are going to study here. This is done through row-echelon elimination of polynomial spaces. We will work with a fixed $D : \{1 \dots 9\} \to \{0 \dots 36\}$, but we are not going to specify $P : \{1 \dots 27\} \to \{1 \dots 36\}$. The polynomial spaces we consider are exactly those which we need in order to compute the Fundamental Equation (FE). For this reason we will call them "polynomial FE spaces" or just "$\mathcal{P} - FE$ spaces". We proceed as follows:

1. We consider all monomials for $\mathcal{P}$ with all the 36 variables up to a maximum degree d. Let D_0 be the dimension of this space.
2. We consider all possible monomials M in 36 variables of degree $\leq d$.
3. We substitute in M all the variables with the round ANF expressions of Section 3.5 except that we do not go as far as replacing everything by concrete variables for example $h \leftarrow F + Z1 + e$ as in later Section 3.7. Such transformed monomial is called M'.
4. Now we are going to fix a part of the long-term key: we will replace all the values which of type $x_{D(i)}$ including when $D(i) = 0$, by the actual variables 1 to 36, or $K = S1$ as previously. We also replace four more variables which are $G = x_{P(6)}$, $H = x_{P(13)}$, $I = x_{P(20)}$ and $J = x_{P(27)}$ and should be 4 disjoint variables from any of the $x_{D(i)}$. All these 9+3 variables are replaced by 1-letter variables $a - V$ representing some concrete variable inside the $x_1 - x_{36}$. This transforms each polynomial M' into a new polynomial M''.
5. It is easy to see WHEN $\mathcal{P}$ is fixed, then the FE is obtained as an image of a **linear** application: it is simply equal to a XORing of several polynomials of type $M + M''$ as above, one each time $\mathcal{P}[M] = 1$ where by $\mathcal{P}[M]$ we define the coefficient of the monomial M inside $\mathcal{P}$.
6. Now we perform row-echelon elimination on our polynomials and eliminate **all** polynomials which are multiples of F but not F itself. Let D_1 be the dimension of this space.
7. Now we perform row-echelon elimination on our polynomials and eliminate **all** polynomials which are multiples of K and L but not K, L themselves. Let D_2 be the dimension of this space.
8. Finally an interesting question it then if ALL ordinary products of degrees up to d such as ab can be totally eliminate from FE. Contrary to other dimensions, this dimension is usually quite small and frequently it is just 0. Yet in many interesting cases it is NOT 0, as we will show below. Let D_4 be the dimension of this space.
9. Finally let R_0 be the dimension of equations which are reduced to 0 in our whole process, i.e. the dimension of the kernel of our linear application which takes $\mathcal{P}$ as input and produces FE.
10. Likewise let R_1 be the dimension of equations which are reduced to 0 except that we allow FE containing monomials of degree 1 in variables F, K, L. All

other monomials containing F, K, L are not allowed an should be eliminated. This also corresponds to the kernel for a truncated linear function[33].

Below we give some examples.

Table 1. Dimensions of $\mathcal{P} - FE$ spaces for a strong real-life long-term key 31

LZS	d	D_0	D_1	D_2	D_4	R_0	R_1
31	1	74	36	36	0	0	0
31	2	1334	631	597	0	0	0
31	3	15614	7176	6581	0	0	0
31	4	133424	59535	46929	0	0	0

C.1 Do Polynomial $\mathcal{P} - FE$ Spaces Matter?

For example we can compare two different keys one of which is a real-life key from 1980 known as 31 and KT1 type, on which no attack was ever found, cf. [21], and two highly vulnerable keys, 993 is not of type KT1 yet weak, cf. Section 9.4, and 881 is of type KT1 yet very weak, cf. Appendix A A.2.

Table 2. Comparison of $\mathcal{P} - FE$ spaces for keys 31, 881 and 993 at degree 3.

LZS	d	D_0	D_1	D_2	D_4	R_0	R_1
31	3	15614	7176	6581	0	0	0
881	3	15614	7176	6580	4	0	1
993	3	15614	7176	6581	1	0	0

We will now drop reporting of D_0 and D_1 which do not vary a lot across different keys, and will report primarily D_2, D_3 and D_4 which vary more frequently.

C.2 On the Significance of D_4

The dimension D_4 is one of the main places where the behaviour of different long-term keys varies very substantially. It is also very consistent across different types or classes of the keys, so that one might almost attempt to guess to which category a given key belongs just based on this figure.

We recall that D_4 occurs after we have eliminated all monomials containing F, $G - J$ and $K - L$, which are the most serious obstacles in designing our annihilator attacks, and also all monomials which contain only cipher state variables $x_1 - x_{36}$ denoted by letters $a - V$ but not those where these variables are multiplied by Z, W, X, Y which are placeholders for a Boolean function.

[33] We consider the same linear application where we erase these 3 terms F, K, L in the output FE.

It is important to explain and understand the role of the equations which enter D_4. In general they do NOT have to exist for our attacks to work, see for example the attack of Section 7.1 where we have $FE = Z + Ze + a + e$ and the part $a + e$ remains and was not annihilated by the specific choice of $\mathcal{P}$. However it is interesting to see that sometimes this enormous set of polynomials CAN at all be eliminated, as it is not obvious that it would ever be eliminated. It is easy to see that when this happens and $D_4 > 0$, we obtain a space where our FE is guaranteed to have at least one solution $Z \equiv 0$, as ALL terms are now divisible by one of the four variables Z, W, X, Y which are the four instances of the same Boolean function. Such cases are interesting, because sometimes if it has one trivial solution, it will have other solutions which are not trivial, which for example is the case in Section 8.2. Moreover, if we find a polynomial invariant-to-be outside of Z_4 where however the FE has no solution, we could add to this equation any linear combinations of polynomials of type D_4 in order to increase the chances that it will have a solution.

Table 3. $\mathcal{P} - FE$ for spaces various keys

LZS	d	D_2	D_4	R_1	KT1	LC-weak	2∃B-weak
31	3	6581	0	0	Y	N	N
606	3	6016	1	5	N	Y	Y
881	3	6580	4	2	Y	Y	Y
827	3	6581	4	0	N	N	Y
799	3	6581	32	0	N	N	Y
787	3	6581	3	2	Y	Y	Y
848	3	6578	36	1	N	Y	Y
771	3	6020	13	0	N	N	Y
113	3	6016	0	4	Y	Y	Y

In this table we say that the key is 2∃B-weak if it is vulnerable to non-linear invariants with $d = 2$ for some Boolean function, even very special and even when $Z \equiv 0$. When $D_4 > 0$ for a certain d then the key is guaranteed to be d∃B-weak at least with $Z \equiv 0$.

D Automatic Invariant Finder Source Code

In this section we list a stable version of the tool. The first part is the basic simulation tool written by Raymond Tan, Wayne Tsui and Brian Gunawan. The second part is the solving tool written by Kristelle Feghali and Julien Nahum.

```
# Symbolic variables
R.<S1,S2,F,W,X,Y,Z,a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v
    ,w,x,y,z,M,N,O,P,Q,R,S,T,U,V,D1,D2,D3,D4,D5,D6,D7,D8,D9,P1
    ,P2,P3,P4,P5,P6,P7,P8,P9,P10,P11,P12,P13,P14,P15,P16,P17,
    P18,P19,P20,P21,P22,P23,P24,P25,P26,P27> =
    BooleanPolynomialRing()
phi_inputs = [V, U, T, S, R, Q, P, O, N, M, z, y, x, w, v, u,
    t, s, r, q, p, o, n, m, l, k, j, i, h, g, f, e, d, c, b, a
    ]

# Each round of phi is a permutation of 36 variables
# Input: Dictionary of concrete values to substitute symbolic
    vars (symbolic variables are used for any values not
    provided)
# Output: List of 36 bits of values (after permutation)
def phi(subsitutions=None):
    inputs = map(lambda x: x.subs(subsitutions), phi_inputs)

    # P: {1, ..., 27} -> {1, ..., 36}
    # Here, P1 refers to input[P(1)] and so on.
    p_out = [P1,P2,P3,P4,P5,P6,P7,P8,P9,P10,P11,P12,P13,P14,
    P15,P16,P17,P18,P19,P20,P21,P22,P23,P24,P25,P26,P27]

    outputs = [None] * 36

    # Compute trivial cases
    for i in range(36):
        if i % 4:
            outputs[i] = inputs[i-1]

    # Compute nontrivial cases
    t_out = map(lambda x: x.subs(subsitutions), fn_T(p_out))
    for i in range(len(t_out)):
        outputs[i*4] = t_out[i]

    return outputs

# Function T
# Input: List of 27 outputs of P
# Output: List of 9 outputs of T, U1-U9
def fn_T(p_out):
    return [F + Z + p_out[5] + Y + p_out[12] + S2 + X + p_out
    [19] + W + p_out[26] + D1,
            F + Z + p_out[5] + Y + p_out[12] + S2 + X + p_out
    [19] + W + D2,
            F + Z + p_out[5] + Y + p_out[12] + S2 + X + p_out[19] + D3,
            F + Z + p_out[5] + Y + p_out[12] + S2 + X + D4,
            F + Z + p_out[5] + Y + p_out[12] + D5,
            F + Z + p_out[5] + Y + D6,
            F + Z + p_out[5] + D7,
            F + Z + D8,
            F + D9]

# Utility function to convert from letter notation (a-z, M-V) to *zero-based* indexes for indexing the result of phi.
# E.g. 'a' -> 35, 'M' -> 9, 'V' -> 0
def get_round_index(input_char):
    char_idx = ord(input_char)
    if ord('a') <= char_idx <= ord('z'):
        result = 35 - (char_idx - ord('a'))
    elif ord('M') <= char_idx <= ord('V'):
        result = 9 - (char_idx - ord('M'))
    else:
        raise ValueError('Invalid character input')
    return result

# Given an output polynomial, returns the input polynomial.
# Input: Output polynomial (all terms must be a-z, M-V)
# Output: Input polynomial that results in the output polynomial
def get_input(P_out, assumptions=None, num_rounds_invariant=1):
    r = phi(assumptions)
    for _ in range(num_rounds_invariant):
        substitutions = {var: r[get_round_index(str(var))] for var in P_out.variables()} # e.g. 'M' -> r[9]
        P_out = P_out.subs(substitutions)
    return P_out
```

Now we list the second part, the solving tool written by Kristelle Feghali and Julien Nahum.

```
"""
Find some invariants automatically.
Takes as input a P_output as a string.

D variables are used to cancel some elements from the FE (not cancelled by input).
Function only supports simple cases, where in the FE you have only one Z (and less than 6 different variables used).

Note: finder does not ALL solutions. All possibles values for Ds are not tested to be able to reduce computations.

"""
```

```
def simpleAutoFinder(pout):
    P_out = eval(pout)

    # Compute P_in without assumptions, to see what D and P are found
    temp_P_in = get_input(P_out, {})
    availableAssumptions = []
    for item in str(temp_P_in).split(' + '):
        if len(item)>1 and not '*' in item:
            availableAssumptions.append(item)
    # Sort so that D are before P
    availableAssumptions = sorted(availableAssumptions)

    # Now we find elements that it would be nice to cancel
    # i.e. elements in the FE, which are not Z1,2,3,4
    # but also elements already present in the FE that cannot easily be canceled (i.e. in a product)
    couldBeCancelled = []
    alreadyPresentElements = []
    for item in str(temp_P_in + P_out).split(' + '):
        if len(item) == 1 and not item in ['W','X','Y','Z']:
            couldBeCancelled.append(item)
        if '*' in item:
            for letter in item.split('*'):
                if len(letter) == 1 and not letter in alreadyPresentElements:
                    alreadyPresentElements.append(letter)

    # Now that we have available assumptions and possibly interesting values we can start computing
    # some solutions

    # To increase chances of success, we want to cancel as many elements as we can with D and P
    baseAssumptions = []

    baseAssumption = {}
    assumptionsValuesUsed = []
    for assumption in availableAssumptions:

        if assumption[0] == 'D':
            baseAssumption[assumption] = None

            # then the value must be a multiple of 4
            for val in couldBeCancelled:
                if val in 'SOyuqmiea' and not val in assumptionsValuesUsed:
                    baseAssumption[assumption] = eval(val)
                    assumptionsValuesUsed.append(val)
                    break
```

```python

            # couldnt find a value to cancel, look for already existing values
            if baseAssumption[assumption] == None:
                for val in alreadyPresentElements:
                    if val in 'SOyuqmiea' and not val in assumptionsValuesUsed:
                        baseAssumption[assumption] = eval(val)
                        assumptionsValuesUsed.append(val)
                        break

            # Still no possible value found for this D, so we take a random one
            if baseAssumption[assumption] == None:
                for val in 'SOyuqmiea':
                    if not val in assumptionsValuesUsed:
                        baseAssumption[assumption] = eval(val)
                        assumptionsValuesUsed.append(val)
                        break

            # Check that assumption was set
            if baseAssumption[assumption] == None:
                raise Exception('Could not find any possible value for assumption '+str(assumption))

        elif assumption[0] == 'P':

            # We're done with the Ds, we add the base assuption to list of assumptions
            if len(baseAssumptions) == 0:
                for val in 'VUTSRQPONMzyxwvutsrqponmlkjihgfedcba':
                    temp = baseAssumption.copy()
                    temp[assumption] = eval(val)
                    baseAssumptions.append(temp)

            else:
                # There are already some assumptions in baseAssumptions, we extend them all
                newAssumptions = []

                for assump in baseAssumptions:
                    for val in 'VUTSRQPONMzyxwvutsrqponmlkjihgfedcba':
                        temp = assump.copy()
                        temp[assumption] = eval(val)
                        newAssumptions.append(temp)

                baseAssumptions = newAssumptions

```

```python
    print("P input without assumptions: {}".format(temp_P_in))
    print("{} assumptions created.".format(len(baseAssumptions)))

    # Now we check all computed assumptions
    success = []
    for index, assumption in enumerate(baseAssumptions):
        if index % 100 == 0 and index>0:
            print("{} assumptions tested, {} found successfull.".format(index, len(success)))
        if checkAssumption(P_out,assumption):
            success.append(assumption)

    if len(success) > 0:

        print("Success! {} invariant(s) was/were found!".format(len(success)))

        for invariant in success:

            P_in = get_input(P_out, invariant)
            FE = P_in + P_out

            # We redo same calculations as in checkAssumption, but without checks as we already know that it works
            Z = []
            elements = []
            groupsOfElements = []
            for elem in str(FE).split(' + '):
                if elem in 'WXYZ':
                    Z.append(elem)
                else :
                    groupsOfElements.append(elem)
                    if len(elem) == 1:
                        if not elem in elements:
                            elements.append(elem)
                    elif '*' in elem:
                        for e in elem.split('*'):
                            if not e in elements:
                                elements.append(e)

            # Now we show our solutions :
            finalAssumption = {}
            finalAssumption[Z[0]] = eval(' + '.join(groupsOfElements))

            print("\n\n------------------------------------------")
            print("P output: {} ".format(P_out))
            print("P input: {}".format(P_in))
            print("Base assumption: {} ".format(invariant))
            print("Final assumption: {} ".format(finalAssumption))
            print("FE: {}".format(FE))


    else:
        print("No solutions found.")
        if len(baseAssumptions) < 100:
            print("couldBeCancelled",couldBeCancelled)
            print("Following assumptions were tested:")
            for baseAssumption in baseAssumptions:
                print(baseAssumption)



"""
Returns true if it manages to find final assumption creating an invariant
"""
def checkAssumption(P_out, baseAssumption):
    P_in = get_input(P_out, baseAssumption)
    FE = P_in + P_out

    # Now we simply isolate the Z, and make sure we have less than 7 variables left to work with
    Z = []
    elements = []
    groupsOfElements = []
    for elem in str(FE).split(' + '):
        if elem in 'WXYZ':
            Z.append(elem)
        else :
            groupsOfElements.append(str(elem))
            if len(elem) == 1:
                if not elem in elements:
                    elements.append(elem)
            elif '*' in elem:
                for e in elem.split('*'):
                    if not e in elements:
                        elements.append(e)
            else:
                return False

    if len(Z) > 1 or len(elements) > 6:
        return False


    # Now we check our solution :
    finalAssumption = {}
    try:
        finalAssumption[Z[0]] = eval(' + '.join(
    groupsOfElements))
    except:
        return False
#          print(groupsOfElements)
#          for element in groupsOfElements:
#              print (element,type(element))
#          print(' + '.join(groupsOfElements))

    if FE.subs(finalAssumption) == 0:
        return True

    else:
        return False
```

Finally here is an example of how to use this tool:

```
simpleAutoFinder(" l + t + k*s*U + j*r*T + U + m + S + T")
```