Structural Nonlinear Invariant Attacks on T-310: Attacking Arbitrary Boolean Functions

Nicolas T. Courtois

University College London, Gower Street, London, UK

Abstract. A major open problem in cryptanalysis of block ciphers is the discovery of round invariant properties of complex type. For some 25 years researchers found no convincing examples of such attacks. Finally some recent papers [53, 23, 17, 20] show how to construct polynomial invariant attacks for block ciphers. However almost all such results are somewhat weak: invariants are simple and low degree and the Boolean functions tend to be very simple if not degenerate. Is there a better and more realistic attack, with invariants of higher degree and which is likely to work with stronger Boolean functions? In this paper we show that such attacks exist and can be constructed explicitly through on the one side, the study of Fundamental Equation of [23], and on the other side, a study of the space of Annihilators of any given Boolean function. The main contribution of this paper is that to show that the "product attack" where the invariant polynomial is a product of simpler polynomials is interesting and quite powerful. Our approach is suitable for backdooring a block cipher in presence of an arbitrarily strong Boolean function not chosen by the attacker. We also show an example showing that the same type of a product of linear polynomials $\mathcal{P}$ can be an invariant for Data Encryption Standard (DES).

Key Words: block ciphers, Boolean functions, non-linearity, ANF, Feistel ciphers, weak keys, backdoors, history of cryptography, T-310, DES, Generalized Linear Cryptanalysis, polynomial invariants, multivariate polynomials, annihilator space, algebraic cryptanalysis.

1 Introduction

Only those who attempt the absurd will achieve the impossible.
– Maurits Cornelis Escher

In this paper we introduce a new **type** of attack on block ciphers.

Not quite a new attack though, as such, attacks with periodic non-linear invariants were known for some 25 years [44, 45, 24]. The novelty is rather in the general philosophy and **features** of the attack, how a single invariant may be able to work against all odds, in a surprisingly large number of cases. The attack we present has unique interesting properties which we have never encountered before. We attempt to construct a higher degree polynomial invariant attack with a **product of polynomials** in one very specific block cipher configuration.

Then we discover that the attack is quite powerful and may work well beyond this specific cipher setup and even for a completely different cipher (e.g. DES).

Product attacks are not new either. This type of attack was known to us for 15 years. However only recently we have been able to find and contemplate some actual working examples of such attacks and we discover they are quite powerful and have some interesting features.

Imagine that we could break AES in a way such that the exact specification of the S-box does not really matter. Impossible? Furthermore, imagine that the S-box could be maliciously crafted by the choice of the linear wiring inside the AES S-box which makes the cipher breakable with high probability for any choice of the non-linear component, leading to a "structural" attack (cf. later Section 1.2). Then, when the non-linear component is fixed, any linear component works with a large probability. Moreover, imagine that it is extremely hard to defend against this attack, because it belongs to a vast space with the number of possible invariants growing as 2^{2^n}, and even if one attack accidentally does not work for one S-box and one cipher wiring, another similar invariant property may work for this exact cipher setup. Overall it may seem a block cipher can be broken no matter what, and nothing is potentially really a problem for the attacker. It is also useful to note that our attacks are also able to work in presence of round constants, variable S-boxes, or other properties other than the key, which make that each round is different. We don't see any of these in AES but in many other ciphers we do, and we want our attack methodology to be extremely general so that we can (try) to apply our attack to more or less any cipher[1].

In this paper we make the impossible possible, however we do not work on AES, which would be too ambitious. For the sake of scientific research we need to test our ideas on a simpler cipher. Numerous papers propose attacks on some toy ciphers [2, 40]. However it is always better (and presumably harder and more ambitious) to work on a real-life cipher. We work on an older block cipher T-310 which was extensively used in the 1980s to encrypt government communications in Eastern Europe [35, 52, 31, 51]. How can we claim that we understand the security of more recent block ciphers such as Simon or AES if we have never yet studied major attacks on some older ciphers? It is also important to note that it is not correct to believe that AES is strong and extremely secure while T-310 is some old cipher. In reality AES is a commercial cipher with only 10 rounds designed to be faster than 3DES, while T-310 is government-grade or military-grade cipher designed to be extremely robust. The hardware complexity of T-310 is thousands of times larger than for DES or AES, cf. [35]. In T-310 more than 1500 rounds are required just to encrypt just one character of the plaintext.

[1] Will such attack still work then, or what is the success probability is another story. However it is important to note that we make many things which are traditionally fixed in block cipher cryptanalysis variable, and try to look at a broader picture going beyond just one cipher configuration.

1.1 New Types of Non-Linear Attacks on Block Ciphers

Non-linear cryptanalysis was known for at least 25 years. What is new here is:

1. That such attacks may actually be found or constructed, with polynomials of higher degree than before, e.g. (rather than quadratic or cubic cf. [53, 23, 24, 42]) and that such attacks may actually exist at all, for any block cipher.
2. Our attack works for an arbitrarily large number of rounds, which is rare but of course, was seen before, e.g. in sliding attacks, cf. [43, 18, 34].
3. Finally it does something which is even more unique. Namely it is able to somewhat ignore or circumvent the non-linear component(s) of the cipher in a novel and powerful way: so that we obtain an almost purely **"structural"** attack on a block cipher which makes our block cipher insecure, more or less ignoring the non-linear components. An attack which is very likely to work also when they are chosen to be extremely strong and also when they are secret, unknown or key dependent [46]. This was seen before in stream cipher cryptanalysis [21, 3, 22] but it is new and unprecedented for block ciphers.

In this paper we assume that attacker cannot choose the non-linear component (S-box or Boolean function) but can make some "cipher wiring" choices of the type "long-term key". Many authors study how one to make a block cipher extremely weak on purpose cf. [23, 1, 17, 48, 4]. Our main goal is to show that a block cipher can be "backdoored" even though many components and a number of rules have already been imposed[2] by the designers and are assumed to be as strong as only possible. We show that there exist extremely weak choices which are quite realistic, because they depend on a very small number of key conditions, and therefore they could occur with a relatively large probability, with a potential to work in some real-life situations. This is a realistic setting similar to what happens in symmetric cryptography at large: for example the S-boxes in Serpent have been built in order to avoid any suspicion of backdooring. For the same reason some ciphers will have no S-boxes whatsoever but may use some natural transformations such as rotations, additions etc. (ARX philosophy). In AES potentially the affine transformation inside the S-box could be manipulated, and in LFSR-based ciphers the LFSR taps could be manipulated, yet the number of "good choices" is extremely small. Then the question is can we make the cipher weak nevertheless, and with a sufficiently large probability. As a proof of concept we will show how to construct a round invariant property which - modulo a number of appropriate connections inside the cipher - works with probability of approx. 15 % over the choice of the non-linear component inside our cipher.

[2] The intention of the designers are on the contrary that there would be no way whatsoever to make the cipher weak, once some strong components are mandated like the S-boxes in DES which were designed using some classified knowledge and expertise which could not be made public at the time, cf. [27, 30].

1.2 Research Goals, Structural Attacks, Generic Attacks

Block ciphers are in widespread use since the 1970s. Their iterated structure is prone to numerous round invariant attacks for example in Linear Cryptanalysis (LC). The next step is to look at non-linear polynomial invariants with Generalised Linear Cryptanalysis (GLC) cf. [44] (Eurocrypt'95). A major open problem in cryptanalysis is the discovery of invariant properties of complex type. Researchers have found, until 2018, extremely few such attacks with some impossibility results [47, 7, 9]. Eventually recent papers [53, 23, 42, 20, 17] show how to construct polynomial invariant attacks for block ciphers, however in almost all such results the Boolean function is extremely weak and invariants are simple and of low degree [53, 23, 42, 20]. Can we do better? What kind of better outcomes can we expect?

[Weak] Structural Invariant Attacks. Most symmetric ciphers can be divided into two distinct parts: a set of relatively simple [linear] transformations which mix bits together, and a set of non-linear components (Boolean functions or S-boxes). We call a "Structural Invariant Attack" an attack where 1) there is a cipher wiring and some special Boolean function such that the cipher has an invariant property $\mathcal{P}$ which propagates for an arbitrarily large number of rounds and 2) the property $\mathcal{P}$ does not depend a lot on the non-linear components and depends strongly on the structure[3] (i.e. wiring) of the cipher. Most structural invariant attacks found to date can be seen as quite weak. We also consider that many attacks are "weak" for another reason: because typically, the choice of the Boolean function is degenerated or pathological: it is almost always of low degree, frequently it does NOT depend on some inputs. For most of the attacks in [23, 42, 20] very clearly, the probability that a cipher with a random Boolean function would be weak in a way as to enable one of the attacks will be negligible or close to zero.

Invariants with Reduction to Zero. Recent papers [23, 20] demonstrate the existence of polynomial invariant attacks which work for **any** Boolean function, based on the fact that all coefficients in a certain algebraic equation are equal to 0, cf. [23]. However this attack is not very realistic either. Previous attacks working for any Boolean function is an attack which would also work when the non-linear components are secret or unknown were not very convincing. Designing such attacks seems to impose too many strong constraints on other parts of the cipher so that it becomes too weak (e.g. Appendix of [23]) and that maybe no one would accept to use such a contrived cipher (e.g. weak ciphers in [25]). If a cipher has a backdoor, this fact should be well hidden and its components should not only look very strong, but even actually should be very strong in terms of non-linearity etc. [6].

[3] For example the block cipher GOST has a strong structural property: it splits very neatly into two loosely connected parts, cf. Fig. 3 and Fig. 4 in [29] leading to advanced structural truncated differential attacks [28, 30].

Strong Structural Invariant Attacks. At the end, we consider a more realistic attack scenario than in previous works. We assume that the attacker knows the non-linear components [they are public and were chosen to be very strong] and that he can now adapt his attack to these components: for example look for weak keys and weak long-term keys for one given Boolean function. This brings the question of the existence of "Strong Structural Invariant Attacks" which would work when the S-boxes or Boolean functions have not been chosen by the attacker, and are potentially random or extremely strong, or just have no[4] property which would be helpful for the attacker (except properties which can hardly be avoided at all, if only we are allowed multiple attempts, e.g. multiple polynomial invariants exist).

How Does It Compare to Current Theory of Feistel Ciphers. A crucial notion in the theory of Feistel cipher is the notion of a "generic attack", cf. for example [49, 50] which attacks work for a certain maximum number of rounds, for example for up to 7 rounds, and then the cipher is provably secure (under ideal round functions such as pseudo-random functions). Our notion of "Strong Structural Invariant Attacks" is quite close to a notion of a "generic attack". From one point of view it is weaker or slightly weaker: our invariants work sometimes, not always, for some round functions. When they work, we have properties which are able to propagate for any number of rounds, which is rather unthinkable with traditional types of "generic attack" in [49, 50]. From another point of view our notion is substantially stronger: we obtain attacks which are able to work for an arbitrarily large number of rounds (at least in certain cases).

1.3 Long Term and Short Term Keys vs. Malicious Cipher Wiring

Our attack is very different than what is typically called a weak-key attack: in many weak key attacks the proportion of weak keys is negligible and therefore such attacks are not very practical with some exceptions [39]. We are interested in realistic attack scenarios where the proportion of weak keys is relatively large.

In our study of structural attacks on block ciphers we need to make a useful difference between a "short term key" which will be typically different for each transmission, and a "long term key" which will be used by all users for example for one year. Many commercial block ciphers such as DES or AES do not have such "long term key", however ciphers designed for important government/industry applications tend to have such a "long term key". For example a modified DES (a common practice in the industry) will have a long term key: an additional secret. A long-term key is a standard feature in T-310, one of the most important block ciphers of the Cold War [35, 52, 31, 51] and it takes a form of specifying the internal connections of the cipher implemented as a printed circuit board with a typical lifespan of one year. A natural question is one of the feasibility of "backdooring" a block cipher, or whether it is possible to make the cipher weak on purpose, specifically through this type of "wiring" choice,

[4] A well-known folklore result (with endless variants) is that almost all Boolean functions do not have any property we would wish them to have, see for example [10, 11].

such as for example the P-box in DES. Older literature would not use the word backdooring, but rather use more politically correct terms such the "design" and "security analysis" of the cipher. However the final objective yesterday and today is the same. We should observe that either "backdooring" is possible OR the designers have done a good job to made it strong in all[5] cases. Overall for each cipher which has a "long term key" and for many other where we can still modify the cipher wiring without permission from the authors, the wiring can potentially be chosen maliciously. We will then say that we have a **"Strong** Structural Invariant Attack" if one can construct an insecure setup or wiring with a large broad applicability or/and large success probability which does not require any other (non-linear) components of the cipher to be also weak. In this paper we show how this can be done for T-310. When the modification of the long-term key is officially allowed, for example in T-310, a feasibility of a weak or malicious choice is an important and eminently practical question.

2 Non-Linear Cryptanalysis

The concept of cryptanalysis with non-linear polynomials a.k.a. Generalized Linear Cryptanalysis (GLC) was introduced by Harpes, Kramer, and Massey at Eurocrypt'95, cf. [44]. A key question is the existence of round-invariant I/O sums: when a value of a certain polynomial is preserved after 1 round. Many researchers have in the past failed to find any such properties, cf. for example Knudsen and Robshaw at Eurocrypt'96 cf. [47]. Bi-Linear and Multi-Linear attacks were introduced [24, 25] for Feistel ciphers branches specifically. The number of such attacks grows as 2^{2^n}, and many authors stress that systematic exploration is not at all feasible [7]. For this reason, there are extremely few positive results on this topic [53, 23, 42, 17] and any method to approximate the solution is valuable, as one working examples for one cipher will typically allow the researchers to find or construct similar attacks also for another cipher.

In this paper and unlike in [53] we focus on invariants which work for 100 % of the keys and we focus on stronger invariants which hold with probability 1. In addition we look at a strong us case: an expensive government cipher in which the block cipher is used for encryption in an extremely low-data rate mode, a lot more costly than 3DES, AES, cf. [35]. Here most cryptanalytic attacks simply do not work (!). However all this complexity is not that useful if we are able to construct powerful non-linear invariants which work for any number of rounds.

2.1 Mathematical Theory of Invariants.

In mathematics there exists an extensive theory of multivariate polynomial algebraic invariants going back to 1850s [38] which deals almost always with invariants w.r.t. linear transformations(!) and has very rarely considered invariants with more than 5 variables and in finite fields of small size. In our work we study

[5] Not quite true for T-310, for example in [33] we read that inside the so called class of KT1 keys mandated by the designers, as many as 3 % of are weak w.r.t non-linear cryptanalysis. This paper is about the remaining 97 % of KT1 keys.

invariants w.r.t **non-linear** transformations and 36 or more variables over $GF(2)$ and those which simultaneously hold in more than one case [required]. More details about how this compares to what is done elsewhere in mathematics can be found in Section 1.5 of [23].

A well-known polynomial invariant with applications in symmetric cryptography is the cross-ratio, [25] and the "whitening paradox" cf. [25, 26].

2.2 Discovery of Advanced Non-Linear Invariant Attacks

There are two major types of invariants in recent research: linear sub-space invariants [7, 9] and proper non-linear polynomial invariants [53, 25, 23]. Several authors [7, 9, 23, 8] including here study both. Our product attack is also a linear subspace attack when all the polynomials in our product are affine which is very frequently[6] which also frequently the case elsewhere [19, 17].

The existence of some invariants does not imply that they can be found. Finding such properties was so far considered as extremely hard. There are two major approaches to our problem: combinatorial and algebraic. A nice algebraic approach it through solving the so called Fundamental Equation cf. [23]. Solving such equation(s), or rather several such equations simultaneously, **guarantees** that we obtain a Boolean function and the polynomial invariant $\mathcal{P}$ which propagates for any number of rounds. However nothing guarantees that the FE equation has any solutions. In this paper we show how to construct just one higher degree invariant which has however the property that [with or without some extra wiring adjustments] it frequently has solutions: it can work with a high probability for any non-linear component used.

2.3 On the Bootstrapping Problem in Cryptanalysis

This paper is not only about a specific attack on an important Cold War cipher, but also about cryptanalysis of symmetric ciphers at large (also inside hash functions, MACs etc). Research in block cipher cryptanalysis has suffered from a bootstrapping problem: we have hardly ever found any invariant attacks, except when the set of all possible attacks is not too large, e.g. in Linear Cryptanalysis (LC). An excessively rich space of attacks have been ignored, and we could not find many interesting attacks because we failed to see or imagine how new attacks could even look like. New examples of working attacks (to imitate in further attacks) are crucial. One cannot claim to ever have studied the security major Feistel ciphers such as 3DES, cf. Section 11 below, or SHA-256, without study of earlier Feistel ciphers, in order to realize that a new extremely large class of invariant attacks has never been studied, or almost, cf. [24]. We essentially need to re-start research on the security of block ciphers[7] from scratch. We also

[6] A important example of attack where a polynomial is of degree 7 and is a sum of two products and where (as it seems) this works better than the best product attack in this case is given in Appendix A.2.

[7] The same applies to research on Boolean functions, the results on Boolean functions in Section 6 are eminently practical and allow construction of attacks on block ciphers working for any number or rounds, very much unlikely the 99 % of all results ever published in this space.

claim that it is **necessary** to study the attacks on T-310 precisely, rather than say, study similar attacks on 3DES. This is because T-310 offers unprecedented flexibility and uses a limited number of key bits in each round which makes that the space of attacks is particularly rich and illuminating. Frequently attacks can be transposed from one place to another, more complex attacks can be constructed from simpler attacks by manipulating the roots of the fundamental equation cf. [23, 19, 17], and simpler attacks can then be removed, cf. Invariant Hopping method [19]. In contrast attacks on ciphers such as DES are expected to be very hard to find, and maybe can be constructed precisely by imitating specific classes of attacks earlier discovered for T-310. Or we will have some partial impossibility results [7, 9] and for 50 more years we will fail to see that there are also possibility results. Further explanations and discussion of this question can be found in Section 1.8 of [23]. Essential insights about what makes that non-linear invariant attacks are actually possible can also be found in [17].

2.4 On Phase Transitions in Cryptanalysis

This paper provides a major example of a phase transition. A sort of "holy grail" or highly desirable property in algebraic cryptanalysis at large, which researchers have been aiming at for decades and rarely found any, cf. [16]. Phase transitions also happen with SAT solvers [14, 15, 41]. There is double phase transition observed in this paper as follows. When the polynomial $\mathcal{P}$ is of degree 1, we get linear cryptanalysis which is not excessively powerful, cf. [33] and where the complexity is very quickly degraded as the number of rounds grows. Then when the degree is 2, we already get some properties true with probability 1 working for any number of rounds, first transition. At this stage however we get attacks working only for a handful of very weak or degenerate Boolean functions, if at all, and the proportion of Boolean functions for which the attack works is negligible or even somewhat double-negligible[8]. Then if we increase the degree to say 4, we can start "playing" with multiple solutions, for example some simple attacks of degrees say 1,2 can be "removed" yet an invariant property of degree 4 will remain, see [19] and Section 10 in [23]. Then at degree 8 a second phase transition happens: due to the structure of the ring of the Boolean functions with numerous divisors of zero, the attack becomes almost inevitable: it works for any Boolean function with some probability (this paper).

2.5 On the Future of Symmetric Cryptanalysis

This paper takes the whole research area in symmetric cryptography to the whole new level. It renders almost entirely obsolete hundreds of research papers on symmetric ciphers and Boolean functions, which results can now be seen as neither very interesting nor extremely relevant in applied cryptography, or rather superseded by new stronger attacks operating in substantially different attack scenarios. Such research is abundant and have been tolerated for very long

[8] Which means that we get exactly one solution only inside a double exponential space of possible Boolean functions, several examples where the solution Z is unique can be found in Section 9 of [23].

time based on the widely admitted (yet problematic and perverse) assumption, that properly written quality research and funded by major governments, is necessarily good, and does not need to be relevant, or to explain correctly to the reader what the implications or practical applications of this research might be.

This paper also substantially increases the amount of algebra and mathematics we are likely to see in future routine symmetric cryptography research. It raises very substantially the game for anyone who would like to design or influence any future symmetric encryption standard. It is clear that no AES candidate from 2000 can be considered to be properly designed and properly studied, by the standards of this paper. Symmetric cipher designers must essentially start working from scratch. Even though there are highly expert researchers who has already in the recent years studied the question of resistance against invariant and partitioning attacks, cf. [7, 9], these researchers have operated so far on pure syntactic level, without a corpus of positive examples and working attacks and therefore it is hard to imagine that the right questions ranking high in practical importance has ever yet been studied. A proper "practical theory" of invariant attacks on block ciphers need yet to be constructed[9] and it could take many years for the crypto community to regain confidence in their ability to design secure block ciphers. A plausible scenario is actually also that many future industrial standards for example in TLS, blockchain or 5G mobile phones are going to ban block ciphers totally, due to temporary inability to evaluate their security in any satisfactory way, and we are going to finally use some provably secure encryption such as QUAD from Eurocrypt 2005 [5].

[9] At the very center of this theory there will be a study of the so called "lack of unique factorization events", see [17].

3 Polynomial Invariant Attacks on Block Ciphers

We call $\mathcal{P}$ a polynomial invariant if the value of $\mathcal{P}$ is preserved after one round of encryption, i.e. if $\mathcal{P}(\text{Inputs}) = \mathcal{P}(\text{Output ANF})$. This concept can be applied to any block cipher except that such attacks are notoriously hard to find cf. [7]. In this paper we are going to work with one specific block cipher with 36-bit[10] blocks. The main point is that any block cipher round translates into relatively simple Boolean polynomials, if we look at just one round. We follow the methodology of [23] in order to specify the exact mathematical constraint, known as the Fundamental Equation or FE, so that we could have a polynomial invariant attack on our cipher. Such an attack will propagate for any number of rounds (if independent of key and other bits). In addition it makes sense following [23] to consider that the Boolean function is an unknown, at least temporarily. We denote this function by a special variable Z. We aim at showing that our attack works if and only if Z is a solution to a certain algebraic equation [with additional variables]. An interesting feature of making Z a variable is to see that even if Z is extremely strong, key dependent or unknown, some attacks will nevertheless work. Another benefit of studying the FE in general is that our research is full of good surprises, there are numerous things which work better than expected. For example in countless interesting cases, the FE equation will not depend on F, and frequently will not depend on K, L either, though sometimes the opposite is required, see [34]. Another major case is when the FE reduces to zero, and the attack works for any Boolean function, see Appendix of [23] and [20].

3.1 Notation and Methodology

In this paper the sign + denotes addition modulo 2, and frequently we omit the sign * in products. Let $x_1, \ldots, x_{36}$ be the inputs of our block cipher which are bits $\in \{0, 1\}$. For the sake of compact notation we frequently use short or single letter variable names

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

and we follow the backwards numbering convention of [23] with $a = x_{36}$ till $z = x_{11}$ and further we use $M = x_{10}$ till $V = x_1$ which allows to avoid other capital letters such as F, K, L or W, X, Y, Z used elsewhere. We consider that each round of encryption is identical except that they can differ only in some "public" bits called F (and known to the attacker) and some "secret" bits called $S1 = K$ and $S2 = L$. In round m these bits are sometimes also called $K = s_{m,1}$ and $L = s_{m,2}$ and they ARE different for different m. However we will omit to specify the round number m because at the end we are constructing **one round** invariants extending to any number of rounds and these variables get eliminated totally. This framework covers most block ciphers ever made except that some

[10] Block size could be increased and our attacks and methods would work all the same.

ciphers would have more "secret" or "public" bits in one round. 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 $e_1 \dots e_6$ will be some 6 of the other variables. In practice, the e_i will represent a specific fixed subset of 6 variables of type a-z, or other such as L. At the end we will have four expressions of type say $Z(L, c, k, l, f, h)$ for each Boolean function Z1-Z4 which later must be replaced by a full formula like:

$$Z \leftarrow Z00 + Z01 * L + Z02 * c + Z03 * Lc + \dots + Z62 * cklfh + Z63 * Lcklfh$$

where $Z00 - Z63$ are the yet unknown coefficients of the ANF for $Z()$.

Polynomial Invariants. We are looking for arbitrary polynomial invariants. For example we could have some combinations of symmetric homogenous polynomials like: $\mathcal{P}(a, b, \dots) = P42 * (abc + abd + acd + bcd) + \dots$ However in general solutions may also be complex irreducible polynomials which are far from being symmetric, cf. [23]. In this paper polynomials will be constructed primarily as products of simpler polynomials. One of the main points in discovery of innovative attacks on block ciphers is that most previous attacks polynomial invariants were of degree 2 [24, 23, 53]. In this paper the degree of the invariant becomes 8.

3.2 Constructive Approach Given the Cipher Wiring

Our attack methodology starts[11] from an arbitrary block cipher specified by its ANFs for one round. Specific examples will be shown for T-310, an old Feistel cipher with 4 branches and undoubtedly the most important block ciphers of the Cold War with some 3,800 cipher machines in active service in 1989 [35, 52, 31, 51]. This cipher offers great **flexibility** in the choice of the internal wiring which will be entirely compatible with original historical hardware. The hardware encryption cost with T-310 is hundreds of times bigger than AES or 3DES, cf. [35]. Does it make this cipher very secure? Not quite, if we can construct algebraic invariants which work for any number of rounds. The block size is 36 bits and the key has 240 bits.

[11] Our approach is to find invariant attack starting from arbitrary rounds ANFs is at the antipodes compared to [25, 26] where the ciphers are very special.

3.3 ANF Coding of One Full Round

We number the cipher state bits from 1 to 36 where bits $1, 5, 9 \dots 33$ are those freshly created in one round, cf. Fig 1. Let $x_1, \dots, x_{36}$ be the inputs and let $y_1, \dots, y_{36}$ be the outputs. One round of our cipher can be described as 36 Boolean polynomials out of which only 9 are non-trivial:

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

Two things remain unspecified: the P and D boxes or the internal wiring. In T-310 this specification is called an LZS or *Langzeitschlüssel* which means a long-term key. We simply need to specify two functions $D : \{1 \dots 9\} \to \{0 \dots 36\}$, $P : \{1 \dots 27\} \to \{1 \dots 36\}$. For example $D(5) = 36$ will mean that input bit 36 is connected to the wire which becomes $U5 = y_{17}$ after XOR of Fig. 1. Then $P(1) = 25$ will mean that input 25 is connected as v1 or the 2nd input of $Z1$. We also apply a special convention where the bit S1 is used instead of one of the $D(i)$ by specifying that $D(i) = 0$.

3.4 The Substitutions.

Overall one round can be described as 36 Boolean polynomials of degree 6; out of which only 9 are non-trivial. One round of encryption is viewed as a sequence of substitutions where an output variable is replaced by a polynomial algebraic expression in the input variables. the full formulas in the general case are simply obtained by hard coding all the $P(i), D(j)$ values inside the formulas above.

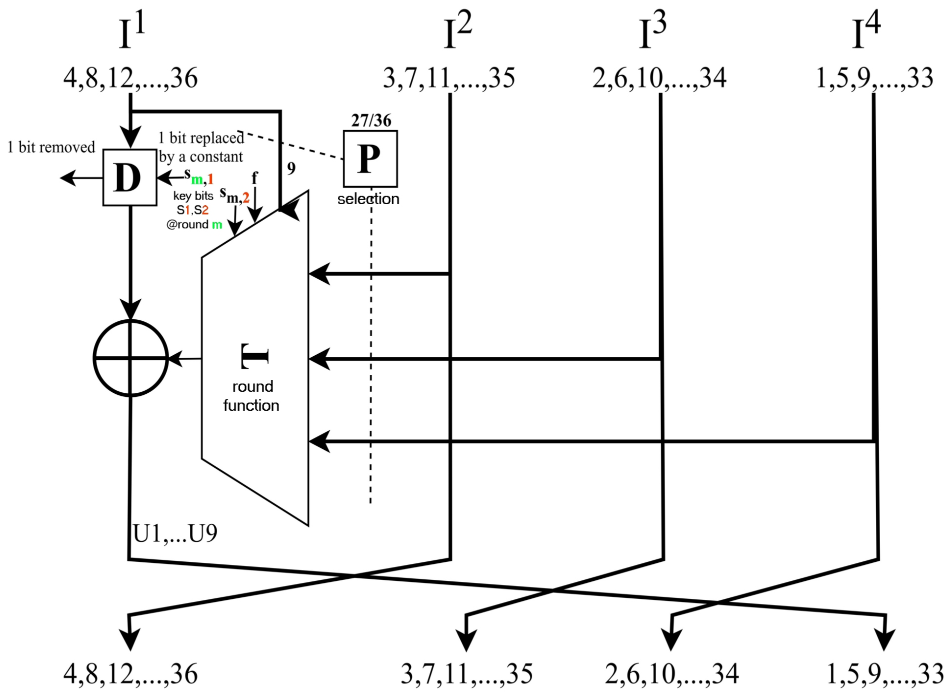


Fig. 1. T-310: a peculiar sort of Compressing Unbalanced Feistel scheme.

Here below is a simplified concrete example following the cipher specification step-by-step for the long-term key 551 used in [23]. More examples can be found in [23].

$$a \leftarrow b$$
$$b \leftarrow c$$
$$c \leftarrow d$$
$$d \leftarrow F + i$$
$$[\ldots]$$
$$[\ldots]$$
$$V \leftarrow F + Z1 + O + Z2 + q + L + Z3 + i + Z4 + k + K$$

In order to have shorter expressions to manipulate we frequently replace $Z1 - Z4$ by shorter abbreviations Z, Y, X, W respectively. We also replace S2 by a single letter L (used at 2 places). The other key bits $S1 = K$ will only be used if some $D(i) = 0$.

4 The Fundamental Equation

In order to break our cipher we need 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 we would get exactly the same polynomial expression $\mathcal{P}$, i.e. $\mathcal{P}(\text{Inputs}) = \mathcal{P}(\text{Output ANF})$ are equal as multivariate polynomials. We obtain:

Definition 4.1 (Compact Uni/Quadri-variate FE). Our "Fundamental Equation (FE)" to solve is a sum of two polynomials like:

$$\mathcal{P}(\text{Inputs}) + \mathcal{P}(\text{Output ANF}) = 0$$

or more precisely

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

where again $Z1 - Z4$ are replaced by Z, Y, X, W. In the next step, Z will be replaced by an Algebraic Normal Form (ANF) with 64 binary variables which are the coefficients of the ANF of Z, and there will be several equations, and four **instances** Z, Y, X, W of the same Boolean function:

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

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

Likewise we will also replace $Z2$:

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

and likewise for $X = Z3$ and $W = Z4$ and the coefficients $Z00 \ldots Z63$ will be the same inside $Z1 - Z4$, however the subsets of 6 variables chosen out of 36 will be different in $Z1 - Z4$. Moreover, some coefficients of $\mathcal{P}$ may also be variable.

In all cases, all we need to do is to solve the equation above for Z, plus a variable amount of extra variables e.g. $Z63$. This formal algebraic approach, if it has a solution, still called Z for simplicity, or $(\mathcal{P}, Z)$ will **guarantee** that our invariant $\mathcal{P}$ holds for 1 round. This is, and in this paper we are quite lucky, IF this equation does not depend on three bits F, K, L. This is the discovery process of [23] which we do not use here. We rather work with basic paper and pencil maths and build our attack from scratch in stages.

5 Construction of An Attack

Our approach is to try to build some simple invariant attacks which initially will not work at all, but they will make us consider under which conditions they might work. The sort of result we are looking for is that a certain polynomial is an invariant if and only if a certain equation is true, the same as the FE approach before, except that we want to help to make it happen in several stages with some simplifications on the way. We recall that the Fundamental Equation FE is an equation such that an invariant attack $\mathcal{P} \to \mathcal{P}$ for 1 round holds if and only if FE is zero, cf. Def. 4.1 and Def. 4.2. We also consider more general transitions of type

$$\mathcal{P} \to \mathcal{Q}$$

and for some fixed number of rounds, which should be interpreted as the value of polynomial $\mathcal{P}$ for the input variables should be equal to the value of polynomial $\mathcal{Q}$ for the output variables. In other terms the sum of these two values is zero, and informally we talk about "I/O Sums" which we already had in linear cryptanalysis [ignoring key bits] and here we do non-linear cryptanalysis (and we also try to eliminate all the key bits). It is important to see that some transitions are obvious and unconditional. For example in T-310, cf. Fig. 1, if bc denotes the product of variables 35 and 34 in our standard (backwards) numbering scheme, and cd is the product of variables 34 and 33, we have:

$$cd \to bc$$

which is un unconditional transition true in all cases. Here the polynomial $\mathcal{P} = cd$ taken and the input of the cipher is always equal to the value of the polynomial $\mathcal{Q} = bc$ at the output of one round of encryption. Now some other transitions are less obvious and will work if a certain I/O sum of two polynomials will be zero [which will maybe never happen with certainty]. For example in T-310 we always have the trivial transitions $d \to c \to b \to a$ and at degree 2

$$cd \to bc \to ab$$

and we are likely to be in trouble with the next step. The problem in T-310 is actually that a is a multiple of 4, cf. Fig. 1. In general a is lost (erased) in the next round and for example $y_{33} = F + x_{D(9)}$ is created cf. general formulas in Section 3.3 where y_{33} is denoted by the letter d. Now imagine that $D(9) = 36$ and $F = 0$ which means that $a \to d$ and also $ab \to cd$ which happens when $F = 0$, i.e. not always. In this case, in the same way as with FE, we will call a **Transition Equation** or TE the sum of two polynomials for a transition of type $\mathcal{P} \to \mathcal{Q}$ for any number of rounds (typically just 1 round) such that the property works if and only if our equation is zero. Here our pre-condition is $D(9) = 36$ and our **Transition Equation** or TE is simply $TE = F$. In other terms if we insure that TE is 0 our two 4-round property works. Both our FE and TE are just I/O sums or sums of two polynomials, the input polynomial and another (same or not) polynomial in 36 variables after the exact substitutions defined on page 13: such as $a \leftarrow b$ and $d \leftarrow F + a$ knowing that the pre-condition $D(9) = 36$ is what makes that $d \leftarrow F + a$ here.

Many further steps are needed in constructing the attack but the key point is that we are going to try to make this polynomials do something interesting and eventually construct a working invariant polynomial in several steps, while trying to make as few assumptions as possible [e.g. $D(9) = 36$], so that our attack will apply to a large class of keys. We will now start from scratch and we do no longer assume that $D(9) = 36$.

5.1 Eliminating FKL

In this process there are some heuristics, for example we will try to get rid of the secret key variables K, L and the F variables corresponding to the IV or public round-dependent constant, all to be eliminated early on. This approach is suitable for a human[12] and we work with paper and pencil without a computer. One of key problems with T-310 is that some sums are very long, for example $y_9 = F + Z1 + x_{P(6)} + Z2 + x_{P(13)} + L + Z3 + x_{P(20)} + x_{D(3)}$ and the most such expressions contain F and many of the $Z1 - Z4$. A potential solution is that we can eliminate many (but maybe not all) variables quite easily by XORing together consecutive freshly created outputs for example 9 and 13.

We have for example:

$$y_9 + y_5 = x_{D(3)} + W(x_{P(21)}, \ldots, x_{P(26)}) + x_{D(2)}.$$

and similarly

$$y_{25} + y_{21} = x_{D(7)} + Y(x_{P(7)}, \ldots, x_{P(12)}) + x_{D(6)}.$$

where W is a shorter name for the function $Z4$ and Y is another name for $Z2$ which functions are identical except that the inputs are typically connected to two disjoint sets of variables.

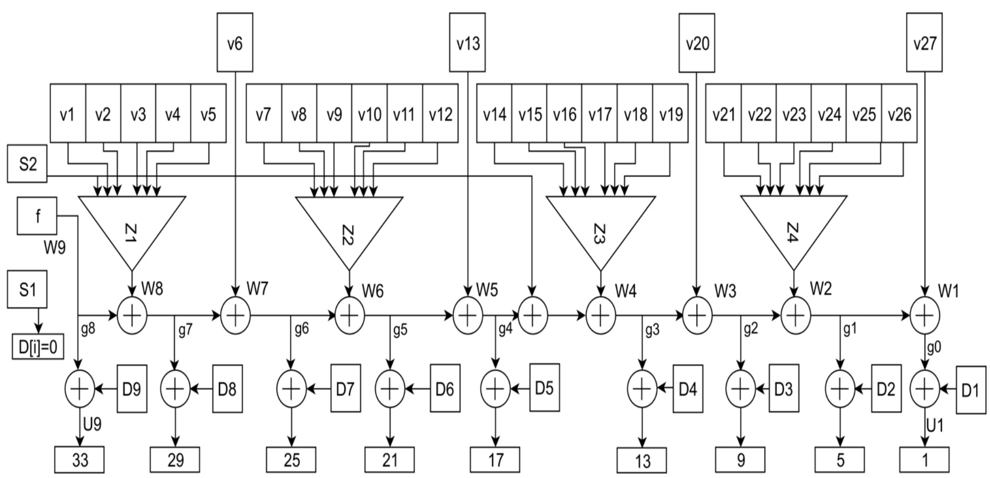


Fig. 2. The internal structure of one round of T-310 block cipher.

[12] Rather than having very complex polynomials where some variables would maybe disappear at the end, which approach would be potentially suitable only if we used powerful formal algebra or constraint satisfaction tools.

5.2 Connecting Two Remote Ends

A direct inspiration for our attack are some remarkable recent attacks where two distant parts of the cipher "talk" to each other while eliminating all the other bits, however complex, involving countless state, key and IV bits, A nice example involving Z1, Z4 and key 714 is studied Section 8.2 in [23] and in more[13] detail in Section 7 of [20]. This type of "complete" elimination was previously seen in stream cipher attacks cf. [21, 3] with further generalisations in [22], but was never seen before in block cipher cryptanalysis. Here we proceed in a different way from scratch and this paper is self-contained. Results in [20, 23] are a model for us except that we are aiming at something yet stronger, we want to eliminate up to 64 additional variables Z00-Z63. More precisely we wish to eliminate also the Boolean function in a certain specific way, not completely, but rather that even a very strong Boolean function not chosen by the attacker could be made to work with reasonable chances of success.

5.3 Connecting $Z2$ and $Z4$

Unlike in the very recent constructions of [23, 20], we will work on the couple of $Z2$ and $Z4$ and avoid $Z1$ because one of the inputs of $Z1$ is the key bit $L = S2$ which we want to avoid. Likewise we avoid $Z3$ as the same key bit L is XORed to its output. Starting from our two basic equations of Section 5.1 above, we now aim at connecting together $Z2, Z4$ a.k.a. Y, W in the simplest possible way, by assuming simply that for example (order inside pairs does not matter):

$$\begin{cases} \{D(2), D(3)\} = \{6 \cdot 4, 7 \cdot 4\} \\ \{D(6), D(7)\} = \{2 \cdot 4, 3 \cdot 4\} \end{cases}$$

This assumption is simply meant to connect together the two equations of Section 5.1: left hand side of one become the right had side on the other after 3 steps, i.e. taking into account transitions like $25 \to 26 \to 27 \to 28$. At this moment it is just wishful thinking: trying to create some sort of peculiar dual-cyclic connection between our two equations. More precisely, we are aiming at a closed cycle with 8 steps as follows, hypothetically:

$$5, 9 \to 6, 10 \to 7, 11 \to 8, 12 \to ?\ 21, 25 \to 22, 26 \to 23, 27 \to 24, 28 \to ?\ 5, 9$$

where the question marks relate to transitions which are not true and will actually never become true. Accordingly, we define the following 8 polynomials:

[13] The second paper shows how to **construct** such an attack through intersection of polynomial spaces in several steps, cf. Section 7.3. in [20] and Appendix C in [23].

$$\begin{cases} A \stackrel{def}{=} (m+i) & \text{which is bits } 24, 28 \\ B \stackrel{def}{=} (n+j) & \text{which is bits } 23, 27 \\ C \stackrel{def}{=} (o+k) & \text{which is bits } 22, 26 \\ D \stackrel{def}{=} (p+l) & \text{which is bits } 21, 25 \\ E \stackrel{def}{=} (O+y) & \text{which is bits } 8, 12 \\ F \stackrel{def}{=} (P+z) & \text{which is bits } 7, 11 \\ G \stackrel{def}{=} (Q+M) & \text{which is bits } 6, 10 \\ H \stackrel{def}{=} (R+N) & \text{which is bits } 5, 9 \end{cases}$$

and then our cycle becomes

$$H \to G \to F \to E \to ?\ D \to C \to B \to A \to ?\ H$$

Now we are going to work on polynomials of type say BD and it easy to see that $BD \to AC$ after one round, and again we are stuck because the polynomial AC contains two multiples of 4 which are $8, 12$ which are erased inside this round so that we do not see an easy transition into something very simple.

Now we are ready for a big jump. As we cannot apparently achieve a lot with low degree invariants, cf. [23], we will directly consider an invariant of degree 8.

5.4 An Interesting Question

The question is given our 4 assumptions on $D(2), D(3), D(6), D(7)$, under what conditions we could have $P = ABCDEFGH$ to be an invariant for 1 round, in other terms what is the FE? The answer is remarkably simple.

Theorem 5.5 (A Simple Degree 8 Non-Linear Invariant Attack). Let

$$\mathcal{P} = ABCDEFGH$$

then $\mathcal{P}$ is a non-zero polynomial[14] and it is a one-round invariant if and only if the FE is equal zero for any input of the cipher and any F, K, L and this FE does not in fact depend on any of F, K, L and it is actually equal to

$$FE = BCDFGH \cdot ((Y+E)W(.) + AY(.))$$

Note. Here the bits which are used as inputs to $W()$ and $Y()$ are exactly those which would be used inside the cipher and we study them in more detail later.

Proof of Thm. 5.5: When necessary we will distinguish input and output-side polynomials (which are like different versions of the same polynomial) by an index in the exponent such as A^o vs. A^i which can be defined for any polynomial A in the same way as in Def. 4.1: for example $A^i = A(a, b, c, d, e, f, g, h, \ldots)$, i.e. $A^i = A$ and there is no change, and we will transform the output polynomial in order to eliminate all the output variables and use only input variables,

[14] Important to check each time we multiply many terms.

$$A^o = A(\phi(\text{Inputs})) = A(b, c, d, F + i, f, g, h, F + Z1 + e, \ldots)$$

where ϕ is the transformation induced by 1 round of encryption and where ϕ(Inputs) denotes a sequence of 36 polynomial expressions of output-side variables $(a^o, \ldots, V^o)$ expressed as Boolean function of the 36 input-side variables [with some extra variables such as secret key variables]. For example the variable a is replaced by polynomial b and d by $F+i$. In other words they are written as formal polynomials in B_{36} corresponding to the ANF expressions of one round of encryption (and as a function of inputs of this round). Our usage of exponents is similar as in the mathematical (Hilbertian) invariant theory. Our exponents can be simply interpreted as transformations on polynomials, or more precisely as operations belonging to a certain group of transformations acting on a set of Boolean polynomials $\mathcal{P}$ or A or other say $(azM + b) \in B_{36}$. In this sense exponent A^i corresponds to the identity transformation, and our output transformed polynomial A^o would be denoted by A^ϕ in classical invariant theory, where ϕ is our cipher round function $\phi : 0,1^{36} \to 0,1^{36}$. This function ϕ typically depends also on 3 extra variables F, K, L however frequently, the final result, the polynomial A^ϕ will **not** depend on these variables F, K, L, or these variables get eliminated, and each time this happens, there is no ambiguity, A^ϕ is well defined and equal to A^o.

Later this distinction will not even be needed as there will be no ambiguity: at the end the $FE = \mathcal{P} + \mathcal{P}^\phi = \mathcal{P}^i + \mathcal{P}^o$ is expected to be written using input side (and maybe internal secret/public variables FKL) ONLY, and all the output-side variables will be eliminated, thus removing any ambiguity. We recall our assumption:

$$\begin{cases} \{D(2), D(3)\} = \{6 \cdot 4, 7 \cdot 4\} \\ \{D(6), D(7)\} = \{2 \cdot 4, 3 \cdot 4\} \end{cases}$$

and in Section 5.1 we have already established that

$$H^o = y_9 + y_5 = x_{D(3)} + W(.) + x_{D(2)} = W(.) + A^i$$

and

$$D^o = y_{25} + y_{21} = x_{D(7)} + Y(.) + x_{D(6)} = Y(.) + E^i$$

which expressions can be reinterpreted as showing how the polynomials D and H on the output side can be rewritten as expressions using only input-side variables and the combination of bits used here were already chosen in Section 5.1 in such a way that all bits FKL are already eliminated. Following Def. 4.1 all we have to do is to add the polynomial $ABCDEFGH$ to itself on the output side after substitutions with input-side variables. This latter polynomial is equal to:

$$A^oB^oC^oD^oE^oF^oG^oH^o = B^iC^iD^i(Y(.) + E^i)F^iG^iH^i(W(.) + A^i) =$$

$$BCDFGH(Y(.) + E^i)(W(.) + A^i) = BCDFGH(YW + YA + EW + EA)$$

Finally we add the last expression to the input polynomial $ABCDEFGH$ and obtain that the FE is equal to (sum of input and output side polynomials)

$$ABCDEFGH + BCDFGH(YW + YA + EW + EA) =$$

$$BCDFGH(AE)+BCDFGH(YW+EW+AY+AE) = BCDFGH((Y+E)W+AY)$$

which is the exact result we need:

$$FE = BCDFGH \cdot ((Y(.) + E)W(.) + AY(.))$$

5.6 Another Proof - Exploiting a Cycle of Length 8

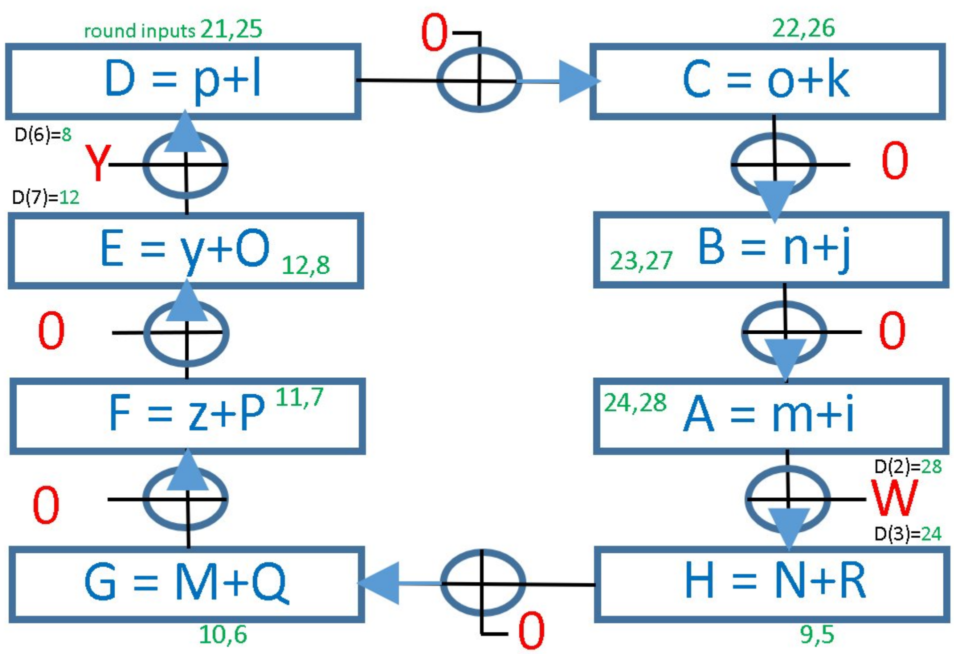


Fig. 3. Connections between the 8 linear factors used in our second attack with $\mathcal{P} = ABCDEFGH$ and LZS265. We multiply all the 8 polynomials.

There is an alternative method to see why this attack works which is about 8 polynomials $ABCDEFGH$ forming a cycle, cf. Fig. 3. The main idea is that we multiply all the 8 polynomials on our cycle while the polynomials W and Y in red, need to be annihilated by some factors in our product, cf. our more general attack and construction in Section 12.

5.7 Solving the FE

Constructing non-linear invariants is sometimes a complex process which requires to manipulate complex polynomials with a computer, cf. Section 7.3. in [20] and Appendix C in [23]. In this paper the result is quite simple and was found by paper and pencil. We need to make sure that FE is equal to zero, or solve:

$$0 = BCDFGH \cdot \big((O + y + Y)W(x_{P(21)}, \ldots, x_{P(26)}) + (i + m)Y(x_{P(7)}, \ldots, x_{P(12)})\big)$$

Interestingly in this expression we do not enumerate the inputs of the first Y which are also $Y(x_{P(7)}, \ldots, x_{P(12)})$, which is because we aim at eliminating the term $(O + y + Y)$ completely later on by making sure that the product of other terms is always zero. We also have a polynomial where F, K, L are already eliminated. However there are many other variables. At the first sight it may seem that our equation which is polynomial of degree 18 in as many as 12 input variables plus 63 variables specifying the Boolean function is unlikely to be systematically equal to 0. What we are going to show now is simply mind-blowing: this polynomial can be made to be exactly zero in all 2^{12} cases, even though **no effort** will be made whatsoever to use a particularly simple Boolean function or even to influence it. In fact we have chosen our example really well and something incredible will happen. This equation will become more easily solvable after a certain choice of which 12 bits will be connected to the 12 inputs of $Y()$ and $W()$ is made. Then an exponentially large space of Boolean functions will just disappear, they will be annihilated, and our equation will be partially reduced to zero in a later Section 6: not always but with a large probability.

Before we get there, the first step is to request that in our single equation which is a sum of two terms, both components are zero. For example we will aim at insuring simultaneously that:

$$\begin{cases} CFH \cdot W(x_{P(21)}, \ldots, x_{P(26)}) = 0 \\ BDG \cdot Y(x_{P(7)}, \ldots, x_{P(12)}) = 0 \end{cases}$$

which will imply that our FE is equal to 0 in all cases, i.e. our one round invariant attack works for any key and any F. At this stage it may be hard to believe but we will see that it is sufficient to put as inputs to W the 6 inputs of CFH in a well-chosen order, not every order works but many do, and the inputs of Y needs to be the 6 inputs of BDG in another order, and that both orders must be compatible (both functions W and Y are just two instances of the same Boolean function). Moreover this actually works for any possible way to split the product $BCDFGH$ in two products of degree 3 for example $(BDG)(CFH)$ provided that later the inputs of W and Y are suitably aligned. Here is for example one solution which works well as we will see later:

```
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
```

below we explain is full detail how such a key can be created.

5.8 The Satisfiability Problem

Initially we want to do something quite which may seem extremely difficult. Make sure that a polynomial equation of degree 18 in as many as 12 input variables and 64 extra variables $Z00 - Z63$ which represent the coefficients inside the ANF of our Boolean function, cf. Def. 4.2 is systematically equal to 0 in all 2^{12} cases, and that we will do it for an absolutely general Boolean function in 6 variables which can be random and is not chosen by the attacker, i.e. the solution should work for as many choices of $Z00 - Z63$ as only possible. The we made it harder be requesting that the two terms of the equation are simultaneously zero. Overall we recall that we decided to insure simultaneously that for example:

$$\begin{cases} CFH \cdot W(x_{P(21)}, \ldots, x_{P(26)}) = 0 \\ BDG \cdot Y(x_{P(7)}, \ldots, x_{P(12)}) = 0 \end{cases}$$

and that the 6 inputs of CFH will become inputs of W and vice versa for Y. For example with our example 265 above we have $P(24) = 22$, which corresponds to the 4-th input d of W. In fact we have already assigned the right 12 variables at some "right" 6+6 places inside LZS 265.

Now we need to see that this alone it not yet sufficient to make the attack work. We need to make sure that the polynomial CFH is an annihilator of the Boolean function W, and that simultaneously BDG is an annihilator for the second instance of the same Boolean function [two conditions]. But is this at all possible that a Boolean function **chosen at random** or not chosen by the attacker would have such peculiar annihilators such as CFH? The answer will be provided in the next section. At this moment we have reduced [modulo checking that all the bits are indeed at the right positions] the problem of backdooring a block cipher to a problem of annihilating a Boolean function **twice** with two annihilators of the form $(a+b)(c+d)(e+f)$ for two disjoint sets of 6 variables.

This is going to be a lot easier than one might expect.

6 Basic Results on Boolean Functions

Let B_n be the ring of all Boolean functions in n variables with $n = 6$ with the usual addition and multiplication of polynomials modulo 2 with all the usual[15] Boolean ring reductions such as $x^2 = x$ already built-in inside our definition multiplication of polynomials in B_n. We recall that an annihilator of a Boolean function $f \in B_n$ is a function $g \in B_n$ such that the product of these functions $f \cdot g$ is zero: i.e. $\forall_{x \in \{0,1\}^n} \;\; f(x) \cdot g(x) = 0$. There are countless divisors of zero in the ring B_n and annihilators exist in vast[16] quantities for any Boolean function. What is more surprising is that annihilators with special properties which will be exactly those which we need in our attack also exist with a large probability. First of all, is it normal that a Boolean function of degree 6 would have an annihilator of degree 3? Yes or almost, we have:

Theorem 6.1 (Algebraic Immunity Degree Bound). Let Z be a Boolean function with $n = 6$ variables $abcdef$. Then either Z of $Z+1$ has at least one annihilator of degree 3.

Proof of Thm. 6.1. Following Thm 6.0.1 in [21] there exists a function g of degree at most 3 such that

$$gZ = h$$

where h has degree of at most 3. In addition following Theorem C.0.1. in the Appendix, of [21], there exists a solutions g such that either $h = 0$ or $h = g$. In the first case $gZ = 0$ and we found an annihilator g of degree 3. In the second case $gZ = g$ and so $g(Z+1) = g + g = 0$ and we found an annihilator of degree 3 for Z+1.

Remark. It is NOT true however that every function Z has an annihilator of degree 3, a nice counter-example is $abcdef + 1$ which has no annihilators at any degree less than 6. It is possible to see that such counter-examples are very few. In practice for most Boolean functions both Z and $Z+1$, this even and also for Boolean functions generated uniformly at random, we get numerous annihilators of degree 3.

6.2 Annihilators of a Special Form

In addition to the existence of annihilators of degree 3, and in addition to the fact that h is likely to very special (or that there exists annihilators g s.t. h is very special) it is interesting that with large probability there exist annihilators such that g is also very special. In what follows we are going to establish some simple yet significant results on this question showing that even though Z is an arbitrary Boolean function, the space of annihilators is very likely to contain

[15] This is a standard approach and implemented in all good algebra software for example in SAGE we have the constructor BooleanFunction and the "PolyBori" implementation.

[16] For example for every function $Z \in B_n$ any multiple of $Z+1$ is an annihilator for Z.

some very interesting and highly symmetric polynomials. These facts are essentially consequences of the fact that the space of annihilators of Z is an ideal, i.e. if $fZ = 0$ then $fgZ = 0$, and it is also a highly structured finite partially ordered set (for divisibility) which being finite is also a lattice, i.e. every two elements have LUB/Sup and in GLB/Inf and where the top element which dominates all the other is 0. This lattice contains many some special quite symmetric elements which dominate many other in the partial order. We start by an auxiliary result.

Theorem 6.3 (Special Type of Annihilators). Let $Z()$ be a Boolean function with $n = 6$ variables $abcdef$ chosen uniformly at random. We consider the following multiple of $Z()$: $T = Z(a+b)(c+d)(e+f)$. We do it many times and by linear algebra [or by a careful study of how different monomials imply also the presence of other monomials] we observe that this polynomial lives in a linear space of dimension only 8. Consequently if $Z()$ is chosen at random then $T()$ is equal to 0 (as a polynomial) with high probability equal to 2^{-8}.

Proof of Thm. 6.3: This result comes from numerous symmetries imposed on $T()$ by the linear factors. The linear space of all possible polynomials of type $Z(a + b)(c + d)(e + f)$ is finite and a quick computation with SAGE maths software shows that its dimension is only 8. This means that every polynomial in this space can be seen as being of the form

$$a_0Z_0 + \ldots + a_7Z_7$$

where $a_i \in \{0, 1\}$ and Z_i are some polynomials forming a basis. Then we need to see that the probability distribution of the 8-tuples a_i must be uniform, this is because it is an image of the input space G_n which is an Abelian group for addition, through a linear application $Z \mapsto Z(a+b)(c+d)(e+f)$ and all preimages for any 8-tuple a_i are cosets w.r.t. H defined as sub-group of annihilators of $(a+b)(c+d)(e+f)$, which must be of equal sizes as they define a partition of B_6 and in fact any polynomial $\in B_n$ can be viewed as bijection which maps one coset into another coset. Given the uniform distribution, the probability that our polynomial is zero is exactly 2^{-8}.
Another Alternative Proof of Thm. 6.3: two alternative methods to prove the same result are given in Appendix B.6 and Appendix B.7 - B.13.

Finally here is the main result which we need in block cipher cryptanalysis:
Theorem 6.4 (Existence of Structured 2x2x2 Annihilators). Let Z be a Boolean function with $n = 6$ variables $abcdef$ chosen uniformly at random. The probability that $Z(a+b)(c+d)(e+f) = 0$ or similar for at least one permutation of variables $abcdef$ is approximately equal to 0.05.

Proof: For any 6 variables there are $\binom{6}{2} \cdot \binom{4}{2}/3! = 15$ ways to select three pairs of variables if we consider as identical all the 3! permutations of the three sets of 2. In the first approximation, each of these choices has a probability of 2^{-8} of being zero, cf. Thm. 6.3 above. If all these 15 choices of a polynomial of type say $Q = (a+f)(c+b)(d+e)$ were independent we would obtain that the probability that none of the 15 choices works is about $(1 - 2^{-8})^{15} \approx 0.94$. Therefore we would obtain a result $Pr \approx 0.057$. A computer simulation shows that the exact result is a bit smaller, we obtained approximately 5.0%.

7 Putting It All Together

We recall that in order for our invariant $\mathcal{P}$ to be preserved after one encryption round, we need to solve [in terms of the wiring of the cipher and the choice of the Boolean function] the following two equations:

$$\begin{cases}(a+b)(c+d)(e+f)\cdot W(a,b,c,d,e,f)=0\\(a'+b')(c'+d')(e'+f')\cdot Y(a',b',c',d',e',f')=0\end{cases}$$

In our solution we are going use the same annihilator for both W and Y, which is not even necessary or required (therefore there exist even more ways of backdooring this cipher than the simple method we present here). Following our Thm. 6.4, this works for 5% of all Boolean functions, and all we need to do is to assign in arbitrary way the 2 time 3 sets of 2 inputs inside each Boolean function. For example, in order to identify $C \stackrel{def}{=} (k+o)=x_{22}+x_{26}$ to the sum $(a+b)$ for W we must insure that $P(21)=22$ and $P(22)=26$ or vice versa. This must be done for all 12 variables $a,b,c,d,e,$ assigned at $P(7)$ to $P(12)$ and the variables a',b',c',d',e',f' are assigned at $P(21)$ to $P(26)$. All this is actually already done in key 265 above and there are numerous ways to do it. Actually a careful reader will see that the order of bits inside LZS265 is such that our FE requires to mandate rather exactly that:

$$\begin{cases}(a+d)(b+c)(e+f)\cdot W(a,b,c,d,e,f)=0\\(a'+d')(b'+c')(e'+f')\cdot Y(a',b',c',d',e',f')=0\end{cases}$$

and this is actually what we do in our proof of concept example. This version is preferred[17] and permuting a,b,c,d,e,f is part of the game[18].

7.1 The Bijectivity Problem

A slight technical difficulty is also that we want our long-term key to be a bijection on 36 bits. The reason for this requirement is that if it is not bijective, the cipher is already broken by a powerful ciphertext-only attack, see [32]. To achieve this in practice is not so difficult: we have used a version of a free open source tool described in Appendix I.11 of [31] which allows to create many random bijective LZS keys keys with arbitrary specific constraints on $D()$ and $P()$. It appears that this process works with probability 1, i.e. once we have specified our constraints as above (four values for the $D(i)$ and 12 values for the $P(i)$) there are still 5+15 coefficients being numbers between 0 and 36 which can be adjusted to make a key which is bijective (and looks like a secure key w.r.t all

[17] We could also have a version with $(a+b)(c+d)(e+f)$, all one needs to do to search for bijective LZS with permuted constraints, see Section 7.1. It is also important to see that there are many solutions, including making arbitrary splits of $BCDFGH$ as explained earlier. Now for example it is impossible to make a KT1 key work with the present attack, this is because KT1 keys require that $P(15)$ is an input of $Z3$, and that P is injective, while here 21 must be an input of either $Z2$ or $Z4$.

[18] It is easy to see that if permuting these bits is not allowed our attack still works with a lower success probability closer to 1 %, cf. footnotes in Section 13.

previously known attacks cf. [32]). The space of long-term keys is simply enormous and our attack does not seem to contradict the bijectivity requirement in any way. We obtained for example the following solution:

```
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
```

and there are plenty of other possibilities: we can also assign inputs a, b of W to be D, or F etc and we are always as it seems able to adjust the missing 15+5 coefficients to form a bijective LZS.

7.2 How to Backdoor T-310

As any Boolean function works in our attack with large probability, cf. Thm. 6.4 all we have to do now is to try some super secure Boolean functions and expect that the cipher will be broken sort of accidentally. In fact, the accident will happen 5 % of the time. For example with key 265 (and with any other variant of our attack) we have for example the following solution:

$$Z'(a,b,c,d,e,f) = ad + bc + be + de + ef + acd + acf + ade + bcf + bdf + cef +$$
$$abcd + abce + abef + bcdf + abcde + acdef + 1 + a + e + c$$

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

Remark 1: It is easy to verify that $Z'(a,b,c,d,e,f)(a+d)(b+c)(e+f) = 0$.

Remark 2: We apologize that our attack does not work for the original Boolean function but we believe that this purely accidental; not because this Boolean function was well[19] chosen or it is at all a strong or secure choice[20].

---

[19] It is rather simply a mater of good versus bad luck, and the chances that the original Boolean function would work without any modification were quite high, about 0.05.

[20] In fact it is very likely that there exists a slight modification of the current attack which also works with the original Boolean function.

8 An Attack Can Hide Another

In Appendix A we show a more general theorem which essentially triples the success probability of our attack (it works in 3 cases out of 4) and in each case we get essentially the same attack. For example with same LZS 265, we can have the following invariant for one round:

$$\mathcal{P} = (A+1)(B+1)(C+1)(D+1)EFGH$$

This invariant works in the same way as the main result in this paper and the success probability is exactly the same. An example Boolean function for which this invariant works is $bc + d + abd + acd + abcd + e + ae + be + ce + bce + abde + cde + bcde + abcde + f + bf + abf + cf + bcf + abdf + bcdf + abcdf + ef + bef + cef + acef + bcef + def + bdef + bcdef + 1$. This example was found by simply trying about 100 different random Boolean functions with LZS 265 and the exact $\mathcal{P}$ above. The attack mechanism is exactly the same cf. Appendix A.

8.1 A Combined Attack with 1+2 Invariants

We expect to obtain even more invariants such that if one does not work for a cipher, another invariant may work. Overall it is easy to see that the cumulative power of the attack described in this paper is at least about $3 \cdot 5 = 15\%$ for a fixed Boolean function chosen at random. This estimation can be justified as follows. Following Thm. 6.4 we get 5 % over the choice of a random Boolean functions in B_6 if adapt the 6+6 inputs of W and Y to our specific degree 8 invariant. It is easy to see that all our 1+2 results such as Thm. 5.5 and Thm. A.1 give the same result of 5 %, cf. notes in Appendix A. Furthermore we have done some testing with 1+2 different invariants $\mathcal{P}$ constructed inside this paper and we have observed that the probability that any two of these attacks would occur simultaneously is not very high. This allows to add these probabilities which leads to about 15 % success rate for one fixed Boolean function.

8.2 Beyond 1R invariants

In addition we expect that the success probability increases when we consider invariants which are periodic after 2 or more rounds, for example of type $\mathcal{P} \to \mathcal{Q} \to \mathcal{P}$ where $\mathcal{Q} \neq \mathcal{P}$. Some examples of such invariants can be found in Section 10 and Appendix B in [23]. This is expected to further improve our 15 % result in the future. Such attacks are however substantially harder to study: the size of polynomials which need to be shown to be identical grows very substantially.

9 Cryptanalytic Relevance of Round Invariant Attacks

Many cryptanalytic attacks do not matter at all for the security of encryption systems in real-life attacks scenarios. Or so it may seem initially. The T-310 cipher actually operates in a very secure and super-paranoid low-data rate encryption mode which makes that many attacks even if they seem very strong, still do not seem to break it in sense of decrypting some communications or recovering the secret key, cf. [33]. However if we dig a big deeper we will find that we can actually break T-310 and recover the key and decrypt communications. We see two major methods to achieve this:

9.1 Decryption Oracle Attacks

We refer to [34] to see one major method in which round invariant attacks such as studied and constructed in this paper affect the security of T-310 in real-life attack scenarios. Such attacks exist, but they require a lot more work. The actual attack is a certain type of long-distance[21] slide attack with a decryption oracle, which is far from being obvious and has additional requirements[22]. One example of a relevant non-linear attack with LZS 771 is given in [34].

9.2 Higher-Order Correlation Attacks

A second major way to exploit these properties is to see that many such attack create partitions of the space of 2^{36} elements into two sets of **unequal** sizes which inevitably leads to powerful ciphertext-only higher-order correlation attacks. Basically we get partitions which will bias the cipher internally in a pervasive way which works for any number of rounds. This modifies very deeply the joint probability distributions for numerous sets of internal variables and is likely to introduce some biases in such distributions. Given the fact that the key bits are repeated in a periodic way in this cipher, and that the plaintext in a natural language will also be strongly biased cf. [32], such biases lead directly to ciphertext-only key recovery attacks.

This particular method to exploit polynomial invariants was first proposed in [20]. The actual exploitation requires many additional technical steps. Several examples of such biases appear in a separate paper [20]. We given here one example which comes from the Section 10 of [23]. Let

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

and we use LZS 551[23] from [23]:

```
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
```

This gives a remarkably simple FE being simply

$$Yg + Yo + gm + mo$$

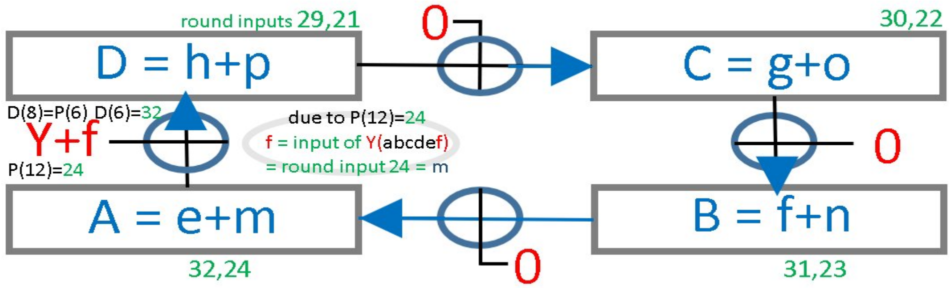


Fig. 4. It is possible to see our invariant attack with LZS 551 is of the form $\mathcal{P} = AC + BD$ with the interesting cycling structure shown here and also studied in [19].

and one solution is $Z = 1 + d + e + f + de + cde + def$. This attack is very different[24] compared to what we see in this paper: $\mathcal{P}$ is irreducible and not at all a product of linear terms like ABC. The invariant $\mathcal{P}$ as above has very interesting properties: it does indeed divide the space of possible cipher states into two unequal size sets and introduces strong biased on the various tuples of internal variables (for subsets of 8 variables $e-h$ and $m-p$ actually used in $\mathcal{P}$). We have checked that already with $N = 3$ variables we obtain some strong biases on N-tuples of variables which can be used in higher-order correlation attacks. For example when $\mathcal{P} = 0$ there are 40 events with $efg = 1$ but only 16 events with $ef(g+1) = 1$. Here the main idea is that correlation attacks involving $N = 1$ or $N = 2$ variables will not work, but starting from $N = 3$ we have plenty of interesting correlations which may be exploited by the attacker. This question is beyond the scope of the present paper and is not obvious in a well-designed cipher, and leads to further technical questions (different correlations need to be combined together to form an attack). Additional examples of biases induced by polynomial invariants can be found in [20].

[21] As opposed to the shorter-distance 1-bit correlation attacks of [34].

[22] Such as looking for invariants actually using some key bits in a very specific way rather then totally eliminating them as we do here and in [23].

[23] A strong example fully compliant with all the requirements of the KT1 class of keys approved to protect government communications cf. [32, 52, 51].

[24] Actually Section 10 in [23] shows further keys such as 558 or 550 with more sophisticated quartic invariants closer to what we see in present paper.

10 Further Developments and Open Problems

We have just started to explore an incredibly rich space of new attacks and the number of possibilities is overwhelming. We list here a few questions which we would like to find answers to in the future:

Question 1. It is possible to find a modified version of the current attack which also works[25] with the exact original Boolean[26] function? We conjecture that this should be possible and is just a matter of trying with a few more attacks of this type, possibly of degree higher[27] than 8. If so it can be used directly to backdoor the original historical encryption hardware by inserting a printed board with a malicious LZS.

Question 2. Is it possible to find strong keys against our attacks or guarantee in any way that the attacks will not work, cf. [7, 9]? For example:

Question 2a. Are there some specific requirements in the style of KT1 or KT2 classes cf. [32] which are able to avoid polynomial invariant attacks?

Question 2b. Is it possible to find some super-strong Boolean functions which can make T-310 secure against invariants attacks such as studied in this paper? This is related to Question 1 and we conjecture that this would be impossible.

Question 3. What about other block ciphers e.g. DES? Then what about ciphers such as AES? In theory, as in [23] our attack is applicable directly to many other ciphers. In practice, given the fact that most modern ciphers use way more key material in one round than T-310 cf. [35], at some moment they could become secure or maybe even provably secure against polynomial invariant attacks, see Section 5.6 in [23] and [7, 9]. However there is no reason to believe that this attack would not work for example for some other classical ciphers such as triple DES. We just need to try and adapt the attack to other ciphers which requires a bit of patience, see Section 11 below for DES.

[25] The probability it would is expected to be high very e.g. 5 % in our current attack.

[26] It is important to see that a Boolean function has typically more than one special or extremely simple annihilator, see Appendix I.19 in [31].

[27] An example of an attack of degree 7 can be found in [17] and another in Appendix A.2.

11 Application to DES

In this section we will show how the same product attack construction in this paper can be applied to DES[28] (or say 3DES) more or less immediately[29]. The number of inputs of each Boolean function is also the same, however there is a lot more[30] key bits to take into account, and different Boolean functions are independent[31]. The main research questions are, with the construction of this paper (essentially the same product attack) and some fixed $\mathcal{P}$, what is the success probability given the P-box and when the Boolean functions are chosen at random. A second (dual) question for the same construction is, given a fixed set of S-boxes and an invariant $\mathcal{P}$, what is the success probability when the P-box is selected at random. Finally, another important question is the Pbox actually used in DES a secure choice and what are weaker choices which can be attacked more easily. We consider the usual structure of DES with duplication of bits at boundaries of S-boxes:

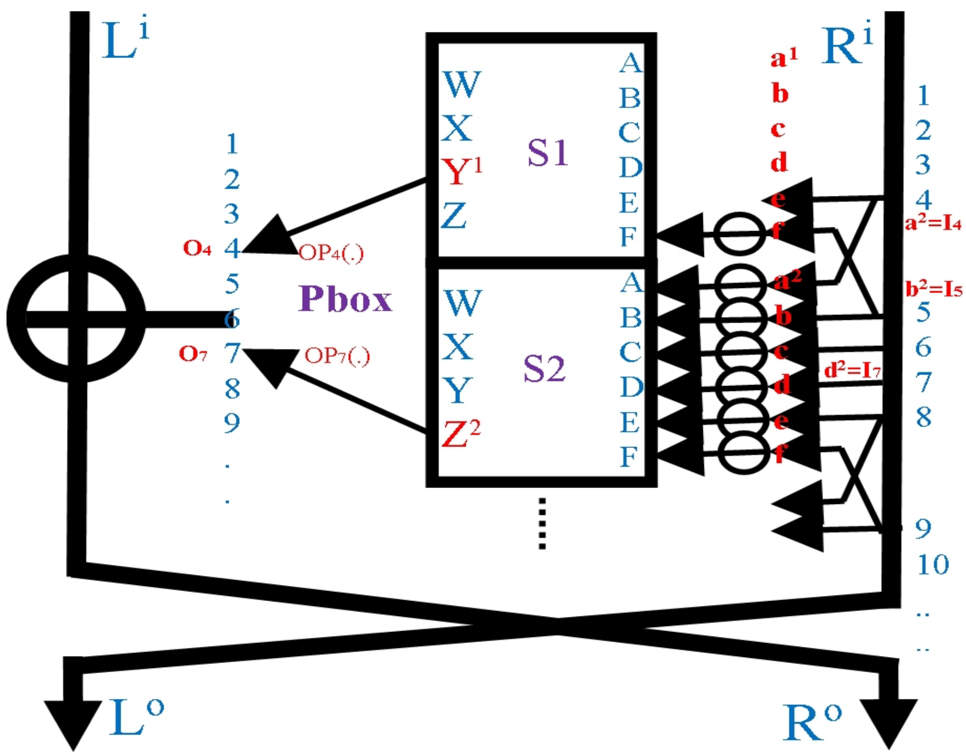


Fig. 5. One round of DES

[28] In Eastern Germany a modified DES was known under the name LAMBDA1 [35] and was implemented around 1990 inside a portable electronic cipher machine T-316.
[29] Many thanks to Jacques Patarin and Willi Meier for suggesting this idea to me.
[30] A considerable difficulty for the attacker, see early results and conclusion in [23].
[31] This is rather very good and helpful for the attacker.

We are now going to adapt the product construction from Section 5 in order to build invariants for one round of DES. We first propose a basic construction which will work in some pathological cases in a very straightforward way. then we show how to make it work for the actual DES P-box and with modified S-boxes, which will be substantially harder.

11.1 Notation and Starting Point

We are going to first construct an attacks which works on 2 bits: a well-chosen pair of bits for example 4,7, corresponding to one S-box for example S2 (both bits are expected to connect to inputs of the same S-box). Let I_{1-32} be the input of the DES round function and let O_{1-32} be the output of the DES round function. In the first round we have $I_1 = R01$ where $R01^i$ denotes the first input on the right side which will be sometimes denoted also by $R01$ if there is no ambiguity. Similarly the notation $R02^o$ denotes the second bit on the right hand side on the ouptut side of the cipher. If our encryption is performed for 1 round only we have $R02^o = O_2 + L02^i$ where $+$ will always denote addition modulo 2 (when used for binary variables). Finally for one round of the DES Feistel scheme we have $L01^o = R01^i$ and the same applies for all 32 bits on the right side at the input.

We also introduce the following notation in order to simplify our polynomial expressions. We are going to denote by $OP_i(.)$ the output polynomial which is connected to output O_i. This polynomial is always one of the 4 outputs W, X, Y, Z for exactly one of the S-boxes, and has 6 input variables $a - f$ which are also consecutive 6 variables of type I_{1-32} (with wrap-around). As an example, on our picture in Fig. 5 focuses on a pair of bits for example 4,7, showing that potentially they could be connected to any pair of outputs, depending on the P-box. Our Fig. 5 suggests that could be they are connected to 3rd output Y^1 of the 1st S-box, which will be later called $Y1$, and to the last output of S2, here denoted by Z^2 and later denoted by $Z2$. Later we will write that $OP_4(R01, \ldots, R32) = Y2(a, b, c, d, e, f)$ and also that $OP_7(R01, \ldots, R32) = Z(a, b, c, d, e, f)$ this ignoring the key bits completely or considering that all key bits are (for the time being) at zero. This is not quite accurate if look at the actual P-box used in DES, see later Fig. 9, where we will see that output 4 and 7 are actually equal to Boolean functions $W6$ and $Z7$ however we do not yet assume that we use the original DES P-box, and at this stage we assume that the P-box can be modified by the attacker.

In addition we will write that $P(Z^2) = 7$ to explain that the last input Z of second S-box S2 is precisely the one connected to output O_7, which is the case on our figure, and then $OP_7(.)$ is actually a polynomial in 6 variables $a^2, \ldots, f^2$ which are also equal to $I_4, \ldots, I_9$. In addition this polynomial has 6 more input variables which are the 6 key bits involved and which we will sometimes ignore. We will in consider arbitrary S-boxes or arbitrary set of 32 Boolean functions which depend on the secret key of an arbitrary length in an arbitrary way. We denote inputs of each S-box by letters $A, \ldots, F$ in blue on our figure, which letters are not be used frequently, and the inputs before key whitening are denoted by more precise notations $a^k, \ldots, f^k$ for S-box k, $k = 1 - 8$ which are those in red colour in our figure, and when there is no ambiguity they will be denoted simply by letters $a - f$.

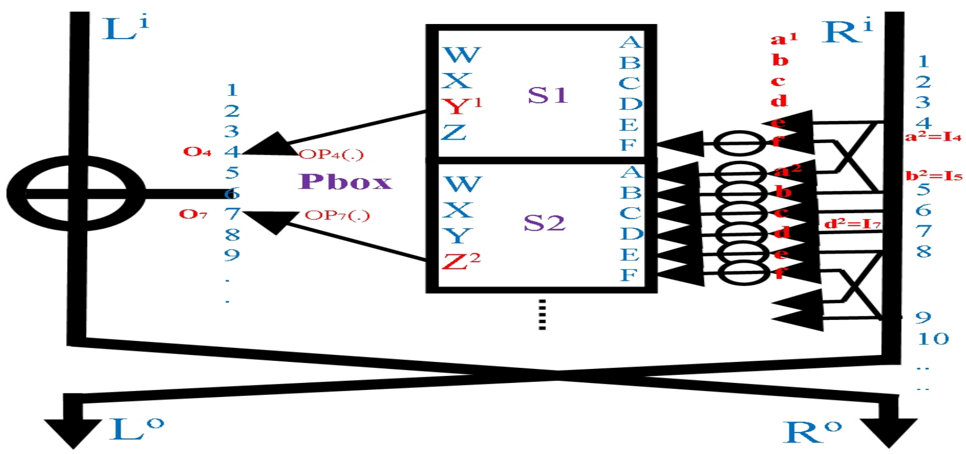
11.2 Basic Product Sub-Construction on 2 Bits

The goal will be at the end to obtain two polynomials which become equal i.e. if $\mathcal{P}(\text{Inputs}) = \mathcal{P}(\text{Output ANF})$ modulo a number of assumptions. First we work on just one pair of bits for one S-box for example S2. For example we consider inputs a, d of S2 in red on our picture which are also simply I_4 and I_7.

$$R_4^i = a^2$$
$$R_7^i = d^2$$

Now we define the following 2 polynomials:

$$\begin{cases} A \stackrel{def}{=} (R_4 + R_7) & \text{which is right bits } 4, 7 \\ B \stackrel{def}{=} (L_4 + L_7) & \text{which is left bits } 4, 7 \end{cases}$$



We then consider how these polynomials compare at both Input/Output sides denoted by 'exponent' indices i and o. We have:

$$\begin{cases} A^i = (R_4^i + R_7^i) = a + d \\ B^i = (L_4^i + L_7^i) \\ A^o = (L_4^i + L_7^i + OP_4(.) + OP_7(.)) \\ B^o = A^i = a + d \end{cases}$$

From here we have:

$$\begin{cases} A^i B^i = (a + d)(L_4^i + L_7^i) \\ A^o B^o = (a + d)(L_4^i + L_7^i + OP_4(.) + OP_7(.)) \end{cases}$$

Could these two polynomials be identical? Yes if the sum of polynomials $OP_4(.) + OP_7(.)$ can be annihilated by $(a + d)$. Interestingly this cancelation opportunity absolutely does NOT contain the inputs $L_4^i + L_7^i$, it is a property of the round function polynomials, and we have the following (local) Fundamental Equation for our choice of two variables $(i, j) = (4, 7)$:

$$FE_{4,7} = A^i B^i + A^o B^o = (a + d)(OP_4(.) + OP_7(.))$$

11.3 Is Our Attack Feasible with Just One FE_{ij}

It is easy to see that unless our DES Pbox is pathologically weak, and the actual Pbox in DES is known to be extremely strong, this attack is NOT possible as such. It is easy to see that the only way to make the attack above work, i.e. AB is an invariant polynomial after any number of rounds, is to either mandate that two outputs of one S-box such as $OP_4(.) + OP_7(.)$ are always at zero for all 2^6 inputs, or to use a Boolean function which is annihilated by $a + d$ which would be possible only for certain keys and only if the 6 inputs of BOTH $OP_i(.)$ and $OP_j(.)$ come from the same S-box. All this is quite special and improbable[32] and so far we have not found a convincing attack on DES.

11.4 More Complex Attacks and the Original Pbox

The next question is more generally how to construct polynomials $\mathcal{P}$ such that they could be invariants for say 2 rounds with the original DES Pbox but with modified S-boxes. Potentially the success probability of such attacks (over the choice of the S-boxes but also over the choice of the key) is quite low HOWEVER is expected to increase as we progress with more complex polynomials and at higher degree. Moreover the number of possible polynomials is extremely large which also helps[33] the attacker.

[32] A probabilistic version of this attack with biased polynomial would be easier to make work, for example it is possible to see that if $Pbox(32) = 31$ and $Pbox(29) = 28$ DES can be distinguished from a random permutation using only about 2^8 Plaintext/Ciphertext or P/C pairs for a fraction of 2^{-16} weak keys this using the original exact NSA-made S-boxes [27] and with $A = R_{28} + R_{31}$ and $B = L_{28} + L_{31}$.

[33] We have studied such polynomials for 15 years, since [24]. However what previous research failed to realise is that our polynomial invariant attack has the ability to attack arbitrarily strong S-boxes when we increase the degree, see Section 2.4. If so, the fact that there are many key bits in each round does not matter that much anymore. This is because the key is considered to be a part of the S-box and our attack is able to work with a potentially quite large success probability, for any S-box including a variable or key-dependent S-box. Our attacks simply have some incredible features never seen before in block cipher cryptanalysis.

11.5 Closed Loop Configurations or Key Property Which Makes Ciphers Vulnerable to Non-Linear Invariant Attacks

Our experience shows that the primary problem in finding "interesting" non-linear invariant attacks is to find a configuration where some set of bits and S-boxes are primarily connected to each other in closed loop, and the name of "closed-loop invariants" is used in a very recent paper [54]. This idea is not new, for T-310 it was also studied in Sections 9.1. and 9.2. and Fig. 8 and Fig. 9 in [20] and for GOST it was already studied in [29]. We reproduce here Fig. 4 from [29]:

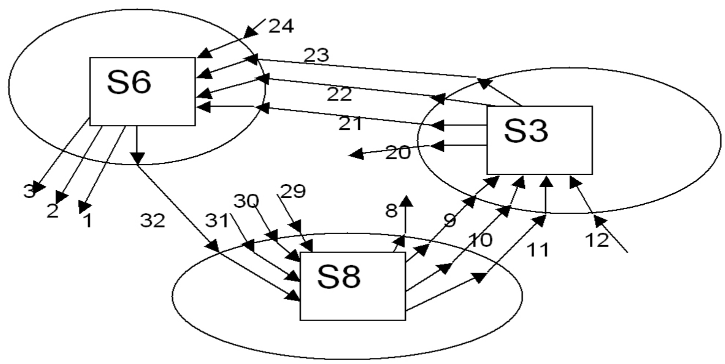


Fig. 6. Closed-loop connection between S-boxes S3,S6,S8 in GOST cf. Fig. 4 in [29].

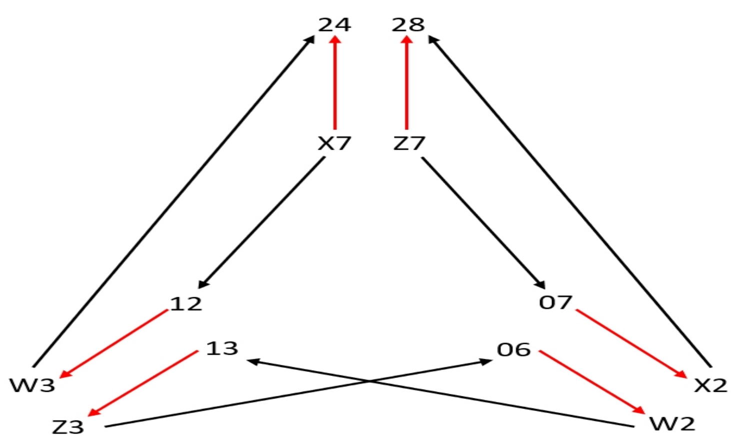


Fig. 7. Closed-loop connection between S-boxes S2,S3,S7 in DES.

The sets of S-boxes involved in all our non-linear attacks on DES found to date, cf. Section 11.6 are also of this type: they have been constructed precisely and deliberately starting from such sets. Similar properties are expected to exist for other ciphers, for example PP-1 cf. [36], and to a certain extent for all block ciphers. In particular we conjecture that they apply quite well to ciphers which are known to be vulnerable to truncated differential attacks, cf. [36, 29].

11.6 A Proof of Concept of the Applicability of Our Product Construction Attack to DES

We show here one full example showing that DES can be weak with a simple product of linear polynomials in the same way as in the current paper. Detailed explanations on how such examples can be generated will be published elsewhere. Our proof of concept is based on the following invariant polynomial:

$$\mathcal{P} = (1+L06+L07)*(1+R06+R07)*L12*R12*L13*R13*L24*R24*L28*R28$$

which is a non-zero polynomial of degree 10.

Theorem 11.7 (A Simple Degree 10 Attack On DES). Let $\mathcal{P}$ be as above, and assume that the S-boxes [including the key] satisfy the following 6 conditions:

```
(1+c+d)*W2==0
(1+c+d)*X2==0
ef*W3==0
ef*Z3==0
ae*X7==0
ae*Z7==0
```

Then $\mathcal{P}$ is a one-round invariant for this round.

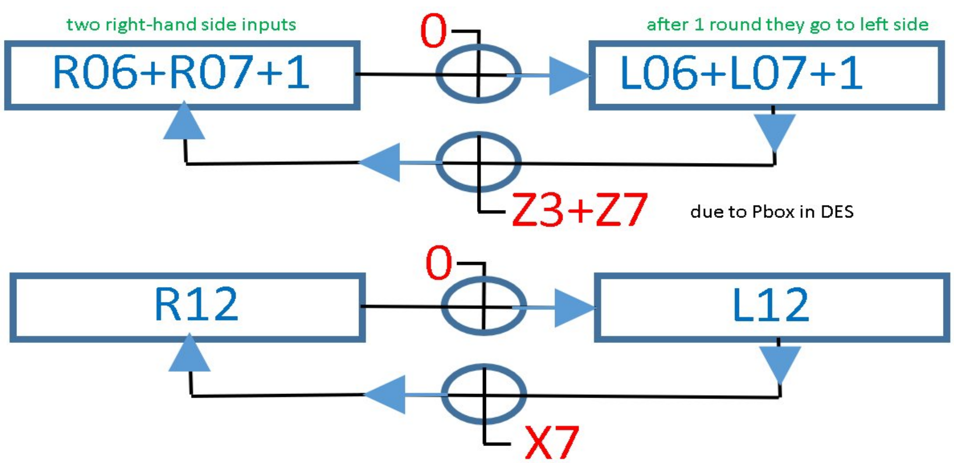


Fig. 8. Two cycles of length 2 relevant to our attack. Similar cycles are also obtained with $R13, L13, W2$ and then $L24, R24, W3$ and $L28, R28, X2$.

Furthermore, if the attack works for some key on 48 bits, the attack also works for a fraction of keys which is 2^{-5} of all possible round keys on 48 bits.

Notation: Here for example

$$(1 + c + d) * W2 == 0$$

means that the polynomial $(1 + c + d)$ annihilates[34] the 1st output W of the second S-box $S2$.

Remark 1: The content of the remaining five S-boxes can be arbitrary.

Remark 2: Here the secret key determines the Boolean function. One way to interpret this attack is that the effective key size of one round of DES against this invariant is only 5 bits.

Remark 3: This attack actually uses exactly the bits the S-boxes and the connections shown on Fig. 7 on the previous page.

Proof of Thm. 11.7:

We need to show that $\mathcal{P}^o$ is equal (as a polynomial) to $\mathcal{P}^i = \mathcal{P}$. We observe that:

$$\mathcal{P}^o = (1+L06^o+L07^o)*(1+R06^o+R07^o)*L12^o*R12^o*L13^o*R13^o*L24^o*R24^o*L28^o*R28^o =$$

which is equal to, where we show which exact Boolean functions are added in one round at the correct places with the standard DES P-box which was shown on Fig. 7 above, we obtain the following formula using input side variables only:

$$(1 + R06 + R07) * (1 + L06 + L07 + Z7 + Z3) * R12 * (L12 + X7) * R13*$$

$$(L13 + W2) * R24 * (L24 + W3) * R28 * (L28 + X2) =$$

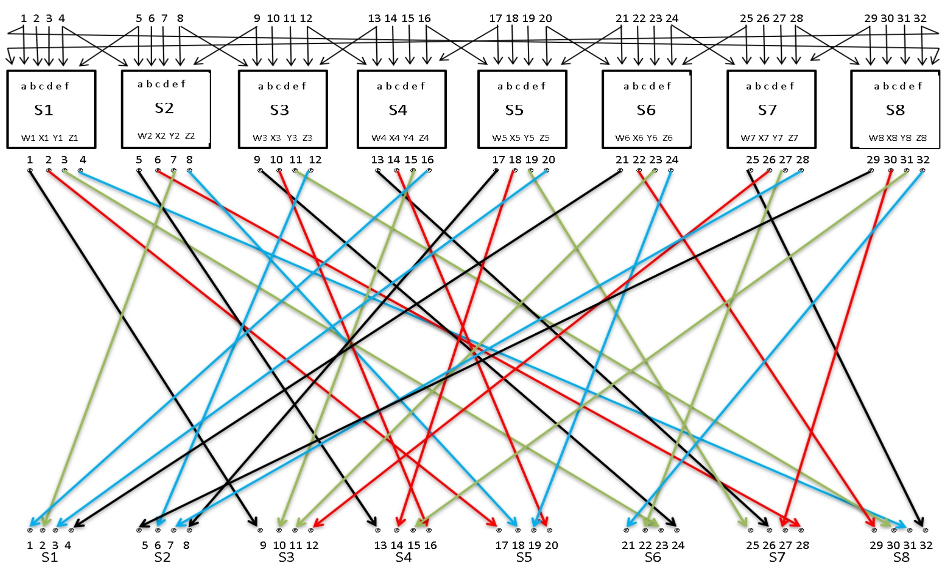


Fig. 9. Full round function of DES showing connections with S-boxes in the next round.

Now we are going to re-write our 6 assumptions knowing that inputs abcdef of S1 are in order R32 up to R5, this is for 1 round of DES and assuming that

[34] The sign == is used here to denote formal equality of polynomials (at other places denoted by ≡ or just = if no ambiguity).

the key is part of the S-box and therefore we don't need to worry (or not yet) what happens when the key changes. We obtain:

```
(1+c+d)*W2==0 becomes (1+R6+R7)*W2==0
(1+c+d)*X2==0 becomes (1+R6+R7)*X2==0
ef*W3==0 becomes R12*R13*W3==0
ef*Z3==0 becomes R12*R13*Z3==0
ae*X7==0 becomes R24*R28*X7==0
ae*Z7==0 becomes R24*R28*Z7==0
```

We see that $W3$ can be simply erased from our complex product because both $R12$ and $R13$ are factors in the whole product and therefore the difference is a multiple of $R12 * R13 * W3$ which polynomial is zero (and is zero for any input). In the same way we can simply erase $Z3$. Then we can also just erase $W2$ and $X2$ because $(1 + R6 + R7)$ is a factor of our product. We get:

$$(1+R06+R07)*(1+L06+L07+Z7)*R12*(L12+X7)*R13*L13*R24*L24*R28*L28 =$$

Finally we can erase $X7$ and $Z7$ because $R24 * R28$ is a factor of the whole product. Thus finally we obtain that:

$$\mathcal{P}^o = (1+R06+R07)*(1+L06+L07)*R12*L12*R13*L13*R24*L24*R28*L28$$

which is exactly equal to our original polynomial $\mathcal{P}^i = \mathcal{P}$ on the input side, and thus we have proven (under our 6 annihilation assumptions) the formal equality of two polynomials, the original and the transformed one:

$$\mathcal{P}^o = \mathcal{P}^i$$

which shows that our invariant attack works for 1 round of DES (if S-boxes with the key satisfy our 6 assumptions).

Finally we need to show what happens if the key changes. It is easy to see that if W is an S-box such that $ef * W3 = 0$ and $ef * Z3 = 0$ then if we add a 6-bit at the input of S3, we can notice that the polynomial ef is invariant thought the same input translation by the key with high probability of 2^{-2}. Therefore if the attack works for one key it will frequently work for another key. Moreover if the attack does not work for the original DES S-boxes for one key, it may still work for the same original DES S-boxes for another key.

Example: Here is an example of S-boxes where our invariant works:

```
S2:0,2,13,9,12,7,3,1,1,3,2,10,3,10,3,2,2,1,13,1,6,1,2,2,2,1,8,13,13,7,3,0,2,
2,14,2,5,6,3,3,1,3,8,11,7,7,1,1,2,2,9,11,0,15,0,1,0,3,8,12,14,4,2,2
S3:11,15,7,4,12,10,12,14,15,13,9,14,15,11,1,2,15,0,7,0,14,0,5,0,8,0,7,0,14,0,9
,0,10,7,13,3,15,7,4,15,5,15,6,15,8,7,13,14,15,0,11,0,10,0,7,0,13,0,11,0,1,0,11,0
S7:11,15,7,4,12,10,12,14,15,13,9,14,15,11,1,2,15,7,12,13,11,7,14,13,10,5,7,5,15,
7,4,15,1,0,15,0,2,0,7,0,13,0,14,0,13,0,14,0,15,0,11,0,10,0,7,0,10,0,11,0,1,0,11,0
```

Further Work: We expect that better results and a higher success probability over the choice of the round keys can be obtained when the polynomial $\mathcal{P}$ is of higher degree and when more than 3 S-boxes are used simultaneously.

Remark: This can be seen as a proof of concept of how to backdoor DES by modifying certain S-boxes and nothing else – everything else is like in the original FIPS Data Encryption Standard.

12 A Generic Multiple Cycle [Multi-]Product Linearization Attack on Arbitrary Block Ciphers

In a companion paper we show that the attack presented in this paper is a special case of a general construction which generalizes all our product attacks[35]. It is also is an attack on block ciphers (primarily applicable to Feistel ciphers) which is not only an attack on T-310, cf. Thm. 5.5 combined with Thm. 6.4 and Section 7. It also is an attack on DES, cf. Thm. 11.7. More importantly, this attack can and should be applied to also to many other block ciphers. In fact in a companion paper we re-formulate this attack in such a way that it does NOT make reference to ANY block cipher in particular. It will become a generic attack of the type that, IF certain equations about the internal connections and S-boxes of our cipher are true, with a substantial freedom in the choice of these equations, then we get a non-linear invariant which is guaranteed to work for any number of rounds[36]. Our new general attack is constructive and generalises previous attacks inside this paper and elsewhere and brings them to a new level. It is based on a simple idea that for a given cipher it may be possible to create a certain closed cycle (or several disjoint cycles) with certain properties, cf. for example Fig. 3. We will then multiply all our polynomials or several subsets for several sub-cycles creating several products which will then be added. Then we show that, depending on some annihilation events for the Boolean functions and the key bits used one specific round of our cipher, our polynomial $\mathcal{P}$ is going to be an invariant attack propagating for any number of rounds.

In some sense this new attack is already contained inside this paper, and all our attacks on T-310 in this paper can be derived from multiplying all the polynomials on the cycle shown on Fig. 3. Likewise our attack on DES can be obtained by multiplying all the polynomials on Fig. 8. A similar attack also studied in [19, 23] can be constructed from Fig. 4. The most general attack is however out of the scope of this paper and will be studied elsewhere. Our new attack could be called "Product Linearization Attack" on block ciphers and is vaguely related to other works which use the word "Linearization" cf. [12]. The main idea with linearization (in all cases) is to add new variables so that everything becomes linear and then try to eliminate these new variables. In

[35] In particular all the attacks in this paper including numerous already known attacks with 2 products cf. for example Appendix A.2 or in Appendix B.1. of [23] or in [19].

[36] The question of how this sort of attack applied to any particular cipher and how it behaves when the secret key is modified can then be studied separately for each cipher as an application of this general attack.

general “Linearization” in narrow the sense of [12] is not a very good attack on block ciphers and [13] is not a very good attack either. Our new attack gives a different meaning to “Linearization”. We will use a “Linearization” steps as a stepping stone aiming at constructing a polynomial invariant attack. An essential insight is that the process of “Linearization” eliminates the only thing which potentially makes the cipher secure, up to all non-linear functions which occur in a given attack. An important improvement compared to earlier algebraic attacks on block ciphers [12, 13] is that we do no longer need to eliminate or deal with and apply “Linearization” to all non-linear components inside a cipher. We only need to apply “Linearization” and in the form of annihilation, only for a tiny subset of them, or rather to some well chosen sums of linear and non-linear polynomials. This explains why this attack is in theory (and in practice) quite powerful.

13 Conclusion

This paper proposes a new way of attacking block ciphers. We construct a non-linear invariant attack with a product of polynomials and some strong and unique features. Given a non-linear component not chosen by the attacker, we show step by step how one can construct a polynomial invariant property of degree 8 which has the ability to work in a large number of cases. It becomes then easy[37] to make our invariant work for an arbitrarily large number of rounds and for any key and any IV. Two additional structural invariants attacks of degree 8 are given in App. A.1. We impose very few constraints on internal cipher wiring (the LZS) and on the Boolean function so that this or similar attack may also work accidentally.

In the most recent paper on this topic [23] the Boolean functions were extremely weak or pathological in order to make the attack work. Our new attack is able to adapt to (or work directly with) arbitrary Boolean functions. The success probability of our attack is surprisingly high: for just one invariant polynomial $\mathcal{P}$ we constructed it is about 5 %, this modulo some adaptations in the cipher wiring (the LZS), cf. Thm. 6.3. This is due to the fact that our invariant polynomial properties operate at a higher[37] degree than previously [23, 53], and also due to surprising results on the existence of annihilators of degree 3 of a special form for a random Boolean function with 6 variables. With all our 3 variants we get a cumulative success probability of about 15 % for any fixed Boolean function chosen at random, cf. Section 8.1.

In this paper we show that a block cipher can be insecure for **almost entirely structural reasons** and that a good choice of complex highly-nonlinear Boolean functions or S-boxes does not really help. A general attack (rich in possibilities)

[37] Compared to earlier attacks which work only in very few cases, cf. [23] we achieved here a phase transition from ‘hard” to “easy”, see also Section 2.4.

with these characteristics was not seen before in block[38,39] cipher cryptanalysis. It is easy to see that the exact attacks described in this paper will work also when the Boolean function is secret and unknown[40] to the attacker. In Section 11 we also show that similar "product" attacks exist for DES (with modified S-boxes).

13.1 Further Remarks

The principal attack proposed here assumes that the attacker knows the Boolean function and we adapt[41] to it. An interesting question is what the success probability when the cipher wiring is completely fixed and the Boolean function is chosen uniformly at random. We have tried this with key 265. In a combined attack using all the 1+2 invariants we have constructed, the success probability is then lower, about 0.8 % experimentally. This still seems incredibly high[42] for an attack where the internal wiring (LZS 265) is completely fixed and the Boolean function can be very strong (it is chosen at random).

13.2 Application to DES

We show that our attack also applied to Data Encryption Standard. In this case a serious difficulty occurs: it only works for a fraction of keys. For DES the attack is not as strong as for T-310. However the effective key size for 1 round in this attack is nevertheless quite small.

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. Martin Albrecht: *Algebraic Attacks against the Courtois Toy Cipher,* In Cryptologia, Volume 32, Issue 3 July 2008 , pp. 220 - 276, 2008.
3. Frederik Armknecht, Matthias Krause: *Algebraic Atacks on Combiners with Memory,* Crypto 2003, LNCS 2729, pp. 162-176, Springer.
4. Arnaud Bannier, Nicolas Bodin, and Eric Filiol: *Partition-Based Trapdoor Ciphers,* `https://ia.cr/2016/493`.
5. C. Berbain, H. Gilbert, and J. Patarin: *QUAD: A Practical Stream Cipher with Provable Security,* In Eurocrypt 2005, LCNS, Springer, 2005.

[38] It may seem weaker than so called "generic attacks" on Feistel ciphers which work typically for only say up to 7 rounds, cf. [49, 50]. In contrast our attack is able to work for an unlimited number of rounds.

[39] In stream cipher cryptanalysis we have worst-case attacks mathematically proven to work for 100 % or all Boolean functions for certain parameter choices, cf. [21, 3, 22].

[40] It is also able to work for several distinct Boolean functions e.g. imagine that $Z1 \neq Z2$, or if we make the Boolean functions key-dependent, cf. [46] and also in DES.

[41] The difference lies in accidental vs. deliberate wiring of the 12 inputs for W and Y.

[42] Due to multiple symmetries inside the expression $(a+b)(c+d)(e+f)$ numerous modification for the W/Y input wiring and/or the Boolean function will also work.

6. Joan Boyar, Magnus Find, René Peralta: *Four Measures of Nonlinearity,* In Algorithms and Complexity, CIAC 2013, LNCS 7878, pp. 61-72, Springer.
7. 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.
8. Tim Beyne: *Block Cipher Invariants as Eigenvectors of Correlation Matrices,* best paper award at Asiacrypt 2018, pp. 3-31. One version is also avail. at `https://eprint.iacr.org/2018/763.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. Claude Carlet: *The complexity of Boolean functions from cryptographic viewpoint,* In Complexity of Boolean Functions (Dagstuhl, Germany), Dagstuhl Seminar Proceedings, no. 06111, 2006.
11. Peter Clote, Evangelos Kranakis: *Boolean functions, invariance groups, and parallel complexity,* SIAM J. Comput. 20 (3) pp. 553-590, 1991, an earlier extended abstract appeared in the Proceedings of Structure in Complexity Theory, pp. 55–65, 1989.
12. Nicolas Courtois, Adi Shamir, Jacques Patarin, Alexander Klimov, *Efficient Algorithms for solving Overdefined Systems of Multivariate Polynomial Equations*, In Advances in Cryptology, Eurocrypt'2000, LNCS 1807, Springer, pp. 392-407.
13. Nicolas Courtois and Josef Pieprzyk: *Cryptanalysis of Block Ciphers with Overdefined Systems of Equations,* Asiacrypt 2002, LNCS 2501, pp.267-287, Springer.
14. 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.
15. 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`.
16. Nicolas Courtois: *Two Philosophies For Solving Non-Linear Equations in Algebraic Cryptanalysis,* `http://www.nicolascourtois.com/papers/Igamma-Mycrypt2016.pdf`, in Paradigms in Cryptology, Mycrypt 2016. Malicious and Exploratory Cryptology, Raphael C. W. Phan and Moti Yung editors, pp. 506-520, LNCS 10311, Springer 2017.
17. Nicolas T. Courtois, Aidan Patrick: *Lack of Unique Factorization as a Tool in Block Cipher Cryptanalysis,* Preprint, `https://arxiv.org/abs/1905.04684` submitted 12 May 2019.
18. Nicolas Courtois, Gregory V. Bard, David Wagner: *Algebraic and Slide Attacks on KeeLoq,* In FSE 2008, pp. 97-115, LNCS 5086, Springer, 2008.
19. 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.
20. Nicolas T. Courtois, Marios Georgiou: *Constructive Non-Linear Polynomial Cryptanalysis of a Historical Block Cipher,* At `http://arxiv.org/abs/1902.02748`.
21. Nicolas Courtois and Willi Meier: *Algebraic Attacks on Stream Ciphers with Linear Feedback,* Eurocrypt 2003, Warsaw, Poland, LNCS 2656, pp. 345–359, Springer. Extended version: `www.nicolascourtois.com/toyolili.pdf`.
22. Nicolas Courtois: *Algebraic Attacks on Combiners with Memory and Several Outputs,* ICISC 2004, LNCS 3506, pp. 3–20, Springer 2005. Extended version available on `https://ia.cr/2003/125/`.
23. Nicolas T. Courtois: *On the Existence of Non-Linear Invariants and Algebraic Polynomial Constructive Approach to Backdoors in Block Ciphers,* `https://ia.cr/2018/807`, last revised 27 Mar 2019.

24. Nicolas Courtois: *Feistel Schemes and Bi-Linear Cryptanalysis,* in Crypto 2004, LNCS 3152, pp. 23–40, Springer, 2004.
25. 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.
26. 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 [25].
27. Nicolas Courtois, Guilhem Castagnos and Louis Goubin: *What do DES S-boxes Say to Each Other ?* Available on `https://ia.cr/2003/184/`.
28. Nicolas Courtois: *An Improved Differential Attack on Full GOST,* in "The New Codebreakers – a Festschrift for David Kahn", LNCS 9100, pp. 278-299, Springer, 2016.
29. Nicolas Courtois: *An Improved Differential Attack on Full GOST,* In Cryptology ePrint Archive, Report 2012/138. 15 March 2012, updated December 2015, `https://ia.cr/2012/138`.
30. 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.
31. 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`
32. 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`.
33. Nicolas Courtois, Maria-Bristena Oprisanu and Klaus Schmeh: *Linear cryptanalysis and block cipher design in East Germany in the 1970s,* in Cryptologia, 05 Dec 2018, `https://www.tandfonline.com/doi/abs/10.1080/01611194.2018.1483981`
34. Nicolas Courtois, Marios Georgiou and Matteo Scarlata: *Slide attacks and LC-weak keys in T-310,* in Cryptologia, vol 43, Iss. 3, 2019, pp. 175–189.
35. 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.
36. Nicolas Courtois, Michał Misztal: *Aggregated Differentials and Cryptanalysis of PP-1 and GOST,* In CECC 2011, 11th Central European Conference on Cryptology. In Periodica Mathematica Hungarica Vol. 65 (2), 2012, pp. 11-26, Springer.
37. Nicolas Courtois: *Algebraic Complexity Reduction and Cryptanalysis of GOST,* Monograph study on GOST cipher, 2010-2014, 224 pages, available at `https://ia.cr/2011/626`.
38. Tony Crilly: *The rise of Cayley's invariant theory (1841-1862),* In Historia Mathematica, Vol. 13, Iss. 3, August 1986, pp. 241–254
39. Nicolas Courtois: *On Multiple Symmetric Fixed Points in GOST,* in Cryptologia, Iss. 4, vol 39, 2015, pp. 322-334.
40. Orr Dunkelman and Nathan Keller: *Linear Cryptanalysis of CTC,* Available at `https://ia.cr/2006/250/`.
41. Ian P. Gent and Toby Walsh: *The sat phase transition, in Proc. of ECAI-94, pp. 105–109, Wiley, 1994.* `http://www.princeton.edu/ chaff/papers/gent94sat.pdf`

42. *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`
43. E. K. Grossman, B. Tuckerman: *Analysis of a Weakened Feistel-like Cipher,* 1978 International Conference on Communications, pp. 46.3.1-46.3.5, Alger Press Limited, 1978.
44. 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.
45. C. Harpes, J. L. Massey: *Partitioning cryptanalysis,* In FSE 97, LNCS 1267, pp. 13–27, 1997.
46. Sandy Harris, Carlisle Adams: *Key-Dependent SBox Manipulations,* In SAC'98, LNCS 1556, pp. 15–26, Springer, 1999.
47. Lars R. Knudsen, Matthew J. B. Robshaw: *Non-Linear Characteristics in Linear Cryptoanalysis,* Eurocrypt'96, LNCS 1070, Springer, pp. 224–236, 1996.
48. Pawel Morawiecki: *Malicious Keccak,* `https://ia.cr/2015/1085`
49. *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.
50. Valerie Nachef, Jacques Patarin, Emmanuel Volte: *Feistel Ciphers, Security Proofs and Cryptanalysis,* hard cover book, Springer, 2017, ISBN 978-3-319-49530-9.
51. 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.*
52. *Klaus Schmeh: The East German Encryption Machine T-310 and the Algorithm It Used,* In Cryptologia, vol. 30, iss. 3, pp. 251–257, 2006.
53. 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.
54. Yongzhuang Wei, Tao Ye, Wenling Wu, Enes Pasalic: *Generalized Nonlinear Invariant Attack and a New Design Criterion for Round Constants,* https://pdfs.semanticscholar.org/bd9f/fb5ae863ee15ce07c71eab8fb84ee9e810d1.pdf In IACR Tr. on Symm. Crypt. Vol. 2018, No. 4, pp. 62-79.

A A Generalized Attack and Generalized Thm. 5.5

In Section 5.4 we asked when assuming our 4 assumptions on $D(2), D(3), D(6), D(7)$, we could have $P = ABCDEFGH$ to be an invariant and we provided a very precise answer which was Thm. 5.5. Here we generalize this result in order to essentially triple the chances of success. The main idea is that instead of working on

$$H \to G \to F \to E \to ?\ D \to C \to B \to A \to ?\ H$$

and later multiplying all the polynomials involved, maybe we could also try to multiply all the polynomials involved in for example:

$$H \to G \to F \to E \to ?\ D+1 \to C+1 \to B+1 \to A+1 \to ?\ H.$$

This leads a stronger and more general attack as follows:

Theorem A.1 (A Generalized Degree 8 Attack). Let I and J be two constants in $\{0,1\}$ and let

$$\mathcal{P} = (A+I)(B+I)(C+I)(D+I)\cdot(E+J)(F+J)(G+J)(H+J)$$

then $\mathcal{P}$ is always a non-zero polynomial of degree 8 for any $I, J \in \{0,1\}^2$, and it is a one-round invariant (for any input of the cipher and any F, K, L), if and only if the FE is equal zero, where the FE is exactly:

$$\begin{array}{c}(B+I)(C+I)(D+I)*(F+J)(G+J)(H+J)*\\((E+J)(W+I+J)+(A+W+J)(Y+I+J)+JI)\end{array}$$

Proof: We use the same notations as in Thm. 5.5 and initially we distinguish input and output-side variables and polynomials by A^o vs. A^i. We recall our assumption:

$$\begin{cases}\{D(2),D(3)\} = \{6\cdot 4, 7\cdot 4\}\\ \{D(6),D(7)\} = \{2\cdot 4, 3\cdot 4\}\end{cases}$$

and in Section 5.1 we have already established that

$$\begin{array}{c}H^o = y_9 + y_5 = x_{D(3)} + W(.) + x_{D(2)} = W(.) + A^i\\ D^o = y_{25} + y_{21} = x_{D(7)} + Y(.) + x_{D(6)} = Y(.) + E^i\end{array}$$

which are again seen as showing how the polynomials D and H on the output side can be rewritten as expressions using only input-side variables (where all bits FKL are already eliminated). The output polynomial is:

$$(A^o+I)(B^o+I)(C^o+I)(D^o+I)\cdot(E^o+J)(F^o+J)(G^o+J)(H^o+J) =$$

$$(B^i+I)(C^i+I)(D^i+I)(Y(.)+E^i+I)\cdot(F^i+J)(G^i+J)(H^i+J)(W(.)+A^i+J) =$$

at this moment we have only inputs left and we can use shorter notations:

$$(B+I)(C+I)(D+I)(F+J)(G+J)(H+J)(Y(.)+E+I)(W(.)+A+J) =$$

Finally we add the last expression to the input polynomial $(A+I)(B+I)(C+I)(D+I)(E+J)(F+J)(G+J)(H+J)$ and obtain that the FE is equal to:

$$(B+I)(C+I)(D+I)(F+J)(G+J)(H+J)\,((A+I)(E+J)+(Y(.)+E+I)(W(.)+A+J))$$

which is the exact result we need:

$$\begin{array}{c}(B+I)(C+I)(D+I)*(F+J)(G+J)(H+J)*\\((E+J)(W+I+J)+(A+W+J)(Y+I+J)+JI)\end{array}$$

Application Notes. At this moment we have only been able to use this result in cryptanalysis when $IJ = 0$, i.e. in 3 cases out of 4, and our earlier attack is a special case when $I = J = 0$. When $IJ = 0$ we proceed in the same way as in Section 5.8 each of the two terms is going to be annihilated with a large probability. We need then to also consider annihilators of the form say $Z(a+b+1)(c+d)(e+f+1) = 0$ which is exactly the same if we consider flipping some inputs of Z, and does not change any of our success probabilities such as in Thm. 6.4.

A.2 A Generalized Attack When $IJ = 1$ and Degree 7 Invariants

It is possible to see that case where $IJ = 1$ can also be attacked in some way. We are not sure however if

$$\mathcal{P} = (A+1)(B+1)(C+1)(D+1)(E+1)(F+1)(G+1)(H+1)$$

is a good attack. Our current preliminary results seem to indicate that it does not work well. We do not know a single example where this invariant would actually work. However we observed that it is very common that

$$\mathcal{P} = (A+1)(B+1)(C+1)(D+1)(E+1)(F+1)(G+1)(H+1) + ABCDEFGH$$

is an invariant for 1 round. Here the degree is now 7 not 8, which type of invariant has never been seen before in block cipher cryptanalysis. Moreover this polynomial can also be factored as a product of 7 linear factors, for example (not unique) as:

$$\mathcal{P} = (1+A+H)*(1+B+H)*(1+C+H)*(1+D+H)*(1+E+H)*(1+F+H)*(1+G+H)$$

In a companion paper submitted elsewhere (and also in [17] - another different example) we show two other less trivial constructions of invariants of degree 7.

B Another Proof of Thm. 6.3

Here we propose another proof of Thm. 6.3. We recall our result to prove:

Thm. 6.3: Let Z be a Boolean function with $n = 6$ variables $abcdef$ chosen uniformly at random. We look at cases where

$$Z(a + b)(c + d)(e + f) \equiv 0$$

where the equality holds in the ring B_n, i.e. for any input. This annihilation happens with probability equal to 2^{-8} over the choice of $Z \in B_6$.

Second Proof of Thm. 6.3: Here below is an alternative second and third proof based on methods and formulas suggested by Matteo Abbondati who was our collaborator in 2019 helping running various student cryptanalysis projects. Both methods are quite general and both allow to derive similar results in other cases. Our proof will be done in several steps: first we prove some useful results and lemmas then we complete our second proof in Appendix B.6 (and also our third proof cf. Appendix B.7 which is completed in Appendix B.13).

B.1 Basic Facts on Annihilators

We denote by $\text{Ann}(Z)$ the set of annihilators of a Boolean function Z which is a linear space which includes also the polynomial equal to 0, and let $\text{DimA}(Z)$ be the dimension of this space. Let B_n be the ring of Boolean polynomials in n variables (polynomials in their ANF without powers or with $x^2 = x$ cancellations done when multiplying the polynomials). For T-310 cipher we have $n = 6$. We have:

Theorem B.2 (Annihilators and HW). Let $Z \in B_n$.

$$|\text{Ann}(Z)| = 2^{\text{DimA}(Z)} = \frac{|\mathbf{B}_n|}{2^{wt(Z)}}$$

where $wt(Z)$ denotes the Hamming weight of the Boolean function Z, meaning the number of $1's$ in its truth table or, in other words, the cardinality of the support of Z.

Proof of Thm. B.2. We start by observing that:

$$\begin{aligned} T \in \text{Ann}(Z) \Leftrightarrow TZ = 0 \Leftrightarrow (T = 0\ \vee Z = 0) \Leftrightarrow (T = 1 \to Z = 0) \\ \Leftrightarrow \text{supp}(T) \subseteq \text{supp}(1 + Z) \end{aligned} \tag{10}$$

Then we consider the following bijection for any $K \subseteq \mathbb{F}_2^n$:

$$\begin{aligned} \Lambda : \{T : \mathbb{F}_2^n \to \mathbb{F}_2 : \text{supp}(T) \subseteq K\} \to \{\chi : K \to \{0, 1\}\} \\ g \longmapsto \chi_{\text{supp}(T)} \end{aligned}$$

Finally let $K = \text{supp}(1 + Z)$. The set on the left has cardinality equal to $|\text{Ann}(Z)|$ thanks to (10) above, while the set on the right has of course cardinality $2^{|\text{supp}(1+Z)|}$. Therefore we obtain the result claimed:

$$|\text{Ann}(Z)| = 2^{|\text{supp}(1+Z)|} = 2^{2^n - |\text{supp}(Z)|} = \frac{|\mathbf{B}_n|}{2^{wt(Z)}}$$

B.3 Probabilities Associated with Annihilator Spaces

Definition B.4. Given a Boolean function Z, we define the probability value associated to it as:

$$\mathrm{AP}_Z^n \stackrel{def}{=} \mathbb{P}\left(T \text{ Random in } \mathbf{B}_n,\, T \in \mathrm{Ann}(Z)\right)$$

Lemma B.5. Now due to Thm. B.2 we have:

$$\mathrm{AP}_Z^n = \frac{|\mathrm{Ann}(Z)|}{|\mathbf{B}_n|} = 2^{-wt(Z)}$$

where again $wt(f)$ denotes the number of 1's in the truth table of f.

B.6 A Quick Second Proof of Thm. 6.3

We can rephrase our theorem using the new notation:

Thm. 6.3: We have

$$\mathrm{AP}^6_{(a+b)(c+d)(e+f)} = 2^{-8}$$

Second Proof of Thm. 6.3: This result become now completely obvious in light of Thm. B.2 and Lemma B.5 above. All we need is to check that the set of points where $(a+b)(c+d)(e+f) = 1$ has $8 = 2^3$ elements which is true because for any $a, c, e \in \{0,1\}^3$ we can adjust b, d, f as follows:

$$\begin{cases} b = 1 + a \\ d = 1 + c \\ f = 1 + e \end{cases}$$

This ends our second proof of Thm. 6.3. $\square$

B.7 A Third Proof of Thm. 6.3

We are now also going to show another third way to derive the same result which allows also to derive more similar results.

B.8 Multiplicative Composition Formula

We need a tool which can be used to compute AP_Z from AP_{f_i} for simpler functions f_i. We will use the following result:

An interesting question is what happens when Z is factorized as a product of two Boolean functions like $Z = f_1 f_2$, which decomposition is not unique.

Theorem B.9 (Combination Formula). Given any two Boolean functions f_1 and f_2 we have:

$$\mathrm{AP}_{f_1 f_2} = \sqrt{\frac{\mathrm{AP}_{f_1}\mathrm{AP}_{f_2}}{\mathrm{AP}_{f_1+f_2}}} \tag{11}$$

This formula will later be used to simplify our problem by lowering the degree.
Proof of Thm. B.9. This formula follows from the following formula:

Lemma B.10. Given any two Boolean functions $X, Y \in \mathbf{B}_n$, we have the following relation between the Hamming weights:

$$wt(X+Y) = wt(X) + wt(Y) - 2 \cdot wt(XY) \tag{12}$$

Proof of Lemma B.5 is done by elementary logic by observing that $+$ is equivalent to logical XOR which corresponds to the symmetric difference of sets for the supports, and have to remove the intersection twice because it was counted twice inside $wt(f)$ and $wt(g)$. Rearranging the formula (12) we can write:

$$wt(f_1 f_2) = \frac{1}{2}\left(wt(f_1) + wt(f_2) - wt(f_1 + f_2)\right) \tag{13}$$

which yields the formula (11) in the following way:

$$\mathrm{AP}_{f_1 f_2} = \left(\frac{1}{2}\right)^{wt(f_1 f_2)} = \left(\frac{1}{2}\right)^{\frac{1}{2}(wt(f_1)+wt(f_2)-wt(f_1+f_2))} = \sqrt{\frac{\mathrm{AP}_{f_1}\mathrm{AP}_{f_2}}{\mathrm{AP}_{f_1+f_2}}}$$

B.11 Triple Composition Formula

We will also need the following result which can be obtained in the same way or by multiple application of Thm. B.9:

Theorem B.12 (Triple Combination Formula). Given any two Boolean functions f_1, f_2, f_3 we have:

$$\mathrm{AP}_{f_1 f_2 f_3} = \sqrt[4]{\frac{\mathrm{AP}_{f_1}\mathrm{AP}_{f_2}\mathrm{AP}_{f_3}\mathrm{AP}_{f_1+f_2+f_3}}{\mathrm{AP}_{f_1+f_2}\mathrm{AP}_{f_1+f_3}\mathrm{AP}_{f_2+f_3}}} \tag{14}$$

we also recall Lemma B.5 which says that:

$$\mathrm{AP}^n_Z = \frac{|\mathrm{Ann}(Z)|}{|\mathbf{B}_n|} = 2^{-wt(Z)}$$

B.13 Finalizing Our Third Proof of Thm. 6.3

Now it becomes easy to compute the final result using Thm. B.12. We have $n = 6$ and three sets of disjoints pairs of variables. Let

$$\begin{cases} f_1 \stackrel{def}{=} (a+b) \\ f_2 \stackrel{def}{=} (c+d) \\ f_3 \stackrel{def}{=} (e+f) \end{cases}$$

It is easy to see that for each possible subset from f_1 through $f_2 + f_3$ until $f_1 + f_2 + f_3$ the Boolean function is balanced and we have $wt(sum) = 2^5$ and $\mathrm{AP}_{sum} = 2^{-2^5}$ in all these cases. This means that we have simply:

$$\mathrm{AP}^6_{f_1 f_2 f_3} = \sqrt[4]{\frac{2^{-4\cdot 2^5}}{2^{-3\cdot 2^5}}} = 2^{-2^5/4} = 2^{-2^3} = 2^{-8}$$

This ends the third proof of Thm. 6.3. $\square$