Cryptographic Security Analysis of T-310

Nicolas T. Courtois[1], Jacques Patarin[2], Klaus Schmeh[3], Maria-Bristena Oprisanu[1], Matteo Scarlata[1,4], Om Bhallamudi[1]

[1]University College London, Gower Street, London, UK
[2] UVSQ, CNRS, Universite de Paris-Saclay, France
[3] cryptovision, Gelsenkirchen, Germany
[4] Computer Science, University of Pisa, Italy

Abstract. T-310/50 is an important Cold War cipher [76]. It was the principal encryption algorithm used to protect various state communication lines in Eastern Germany throughout the 1980s. The cipher seems to be quite robust, and until now, no cryptography researcher have proposed an attack on T-310. In this paper we provide a detailed analysis of T-310 in the context of modern cryptography research and other important or similar ciphers developed in the same period. We introduce new notations which show the peculiar internal structure of this cipher in a new light. We point out a number of significant strong and weak properties of this cipher. Finally we propose several new attacks on T-310.

Key Words: Cold War, block ciphers, T-310, unbalanced compressing Feistel ciphers, algebraic cryptanalysis, ElimLin, SAT Solvers, Differential Cryptanalysis, slide attacks, self-similarity attacks, ciphertext-only attacks.

Acknowledgments. We thank Marek Grajek, Philippe Guillot, Nathan Keller, Jean-Jacques Quisquater, Mate Soos and Bingsheng Zhang for their comments and suggestions.

Part of this work were done by UCL students doing project work for GA18 Cryptanalysis course taught at University College London in 2014-today. We thank the following UCL students who made numerous contributions to our work on T-310. These students are Om Bhallamudi, Mario D'Onghia, Mark Daniels, Maria-Bristena Oprisanu, Nikolai Rozanov, Matteo Scarlata, Varnavas Papaioannou, Moyu Wang.

Table of Contents

1 Basic Facts and History of T-310

T-310/50 is an important historical cipher designed and built by crypto engineers from East Germany in the 1970s. It is known to a larger English-speaking public since a paper published in Cryptologia in 2006 [76]. It was subsequently used to encrypt teletype communications during the last period of the Cold War. T-310 is known as being probably the "most important" cipher of that period and in 1989 there were some 3,800 cipher machines in active service across all sorts of government, party and internal security services [46, 76].

1.1 Chronology on T-310

Based on [45] we present here a short chronology on T-310 encryption machines:

1973 First specification of the tactical technical requirements for the T-310[...] Basic cryptological requirements: "Quasi-absolute security".
1974 Construction of a new cryptographic algorithm. Two mathematician cryptologists were commissioned for 1 year.
1976 T-310/50 teletype encryption device, T-310/80 data encryption device.
1980 Cryptological investigation of the security of the encryption process by cryptologists of the ZCO and the Soviet cryptologists. Finding that the DES has nothing in common with the encryption algorithm ALPHA.
1982 Put into serial production.
1984 The average monthly output of the T-310/50 amounted to 60 devices.
1986 Between 1984-86 there were 290 repairs, of approx. 1400 delivered devices.
1987 Presentation of the T-310/50, as a national encryption device, at the meeting of the ciphering services of the Warsaw Treaty.
07/89 Computer connection to the T-310/50 by telex card ATW in the BC 5120
11/89 There were in employment 3,835 machines of type T-310/50 and 46 machines of type T-310/51.
1990 Last change of the long-term key, use of the LZS-33.
1990 25.07.1990 Publication of the T-310 in the city halls of Berlin with politics and television.
1990 Analysis of the encryption algorithm by the BSI, unofficial statement: extremely secure. Official statement: not authorized to say anything about it.

2 A Block Cipher in A Stream Cipher Mode

T-310 is a synchronous stream cipher which derives its keystream from an iteration of a relatively complex block cipher.

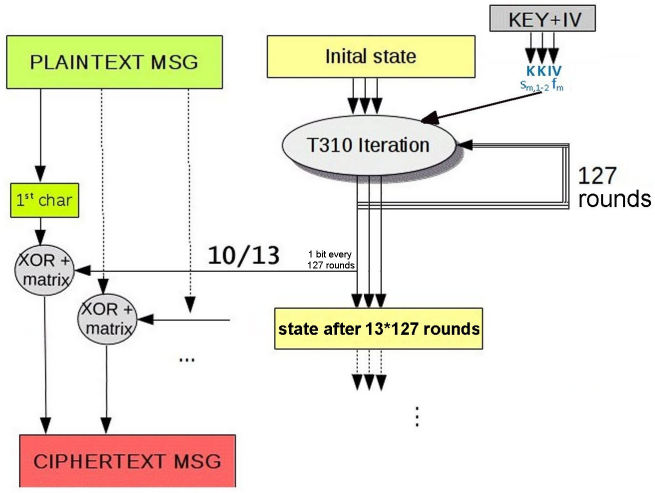


Fig. 2.1. T-310 Encryption Process.

T-310 contains an LFSR. However we cannot really hope to apply attacks on LFSR-based stream ciphers [24, 31], in T-310 the state of the LFSR is known to the attacker and is used to expand the IV into a longer sequence. In addition, in contrast to many LFSR-based stream ciphers, the main iterated component in T-310 is non-linear. It is simply an iterated block cipher with a relatively simple key schedule.

Overall, the main component to study in this paper is a keyed permutation which also takes an IV which we will call "the T-310 block cipher". The block size is only 36 bits, the secret key has 230 bits plus 10 parity bits, and the IV has 61 bits. The block cipher is not used directly to encrypt, but it is iterated a large number of times: Some $13 \cdot 127 = 1651$ rounds of the block cipher are performed[1] in order to extract as few as only 10 bits from the cipher internal state, which will be used to encrypt just one 5-bit character of the plaintext.

It appears that many techniques which have been traditionally developed in cryptanalysis of block ciphers cf. for example [9, 7, 34, 38, 32] should and will to some extent apply to T-310. For example, there exist techniques which break any cipher, if not too complex, cf. [9, 30, 31, 18, 79, 16]. Unhappily T-310 is quite complex.

[1] According to page 17 in [80], the round clocking frequency is 76.8 kHz which gives an encryption speed of about 46.5 characters per second.

3 Feistel and Generalized Feistel Ciphers vs. T-310

As a first approximation, and this is as we will see later, only a first vague (and inexact) classification, it appears that this block cipher belongs to the family of so called "Contracting Unbalanced Feistel ciphers" with 4 branches, cf. [65] and Fig. 4.3 below.

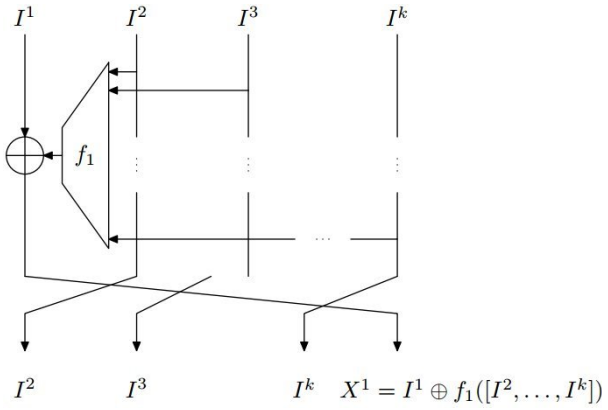


Fig. 3.2. A "Contracting Unbalanced Feistel" cipher with k Branches.

The construction of ciphers has a rich history which can be seen as developing many different ways to produce key-dependent permutations from composition of smaller building blocks which most of the time do not need to be permutations. The original Feistel cipher[2] had 2 branches and was invented around 1971 [48, 49]. Then East German cipher designers have already in 1970s [76] mandated a substantially more complex internal structure which can be seen as a very peculiar sort of "Contracting Unbalanced Feistel" which should be studied in the context of other similar ciphers known in crypto history and in academic literature[3].

3.1 T-310 vs. Other Contemporary Block Ciphers

An important historical example of exactly a "contracting" cipher with 4 branches and a near-contemporary of T-310 is the RC2 cipher by Rivest which was designed in 1989 cf. [61] with an (alleged) collaboration with the NSA. RC2 have been very widely used worldwide for real-life communications security, first in Lotus Notes software and later also in S/MIME encrypted email standard of 1997. Another more academic example of (exactly) a compressing cipher with 4 branches is McGuffin cipher proposed by Bruce Schneier and Matt Blaze at FSE'94 which cipher was immediately broken during the same conference [73]. The earlier RC2 has remained a trade secret for a longer time and only in

[2] Which is to date, probably the most popular block cipher construction ever invented.
[3] There exist countless generalizations and extensions of Feistel schemes, cf. [67, 65, 66, 57].

1997-1998 it was re-discovered and analysed (without great success) in crypto community [61]. Another important historical cipher with a very large real-life footprint which is still used today by millions of people is a block cipher which is used inside the SHA-1 hash function. SHA-1 is "Contracting Unbalanced" Feistel with 5 branches. It was developed by the US-government funded Capstone project which began in 1993 and which aimed at developing a full suite of long-term crypto algorithms with 80-bit security. The Capstone project has also produced the well-known Skipjack algorithm. Skipjack is unique type of cipher with 4 branches which are neither contracting nor expanding [66] sort and more like local application of the basic Feistel with only two branches at one time, with a lot of extra irregular structure [58]. A report from 2011 reveals that Skipjack has been designed by the NSA earlier in the 1980s with "building blocks and techniques that date back more than forty years" [5].

Skipjack and T-310 share the same characteristic of being so called "Type 1" ciphers, which are intended to protect classified information and government communications. The design and specification of such ciphers is expected to be confidential[4]. However eventually ciphers will be declassified, e.g. Skipjack, their spec leaks out, e.g. RC2, or they become obsolete and the spec can be found in government archives, e.g. T-310. The overall result is that these ciphers can eventually be studied by security researchers.

3.2 Weak or Strong - Cryptanalysis

The theory of "contracting" Feistel ciphers indicates that such ciphers update the internal state quite slowly and therefore require a larger number of rounds to be secure than Feistel ciphers with 2 branches [65]. With very strong round functions this theory would recommend at least 8 rounds for a cipher with 4 branches [65], which however is by far insufficient for any cipher build with more realistic (simpler/faster and substantially weaker) components. Most of the ciphers we have mentioned above have a very substantial number of rounds and to the best of our knowledge they achieve a very decent level of security. Even though Lotus Notes software has been an object of a number of controversies regarding deliberate weakening by the NSA, no convincing attack has been published to date against RC2 cipher [61]. Similarly, to this day there is no attack on the full Skipjack cipher cf. [58] and the SHA-1 when used in encryption is also quite robust [62]. Finally, until now, no attack of any sort whatsoever have been published on the T-310 cipher. In this paper we provide a first analysis of T-310.

[4] This means that they will be also subject to export restrictions, and also that publication of research articles or press reporting can be prohibited (e.g. with court orders or under former UK DA rules).

4 Feistel Ciphers and High-Level Structure of T-310

A classical unbalanced Feistel scheme in the contracting family [65] is as follows.

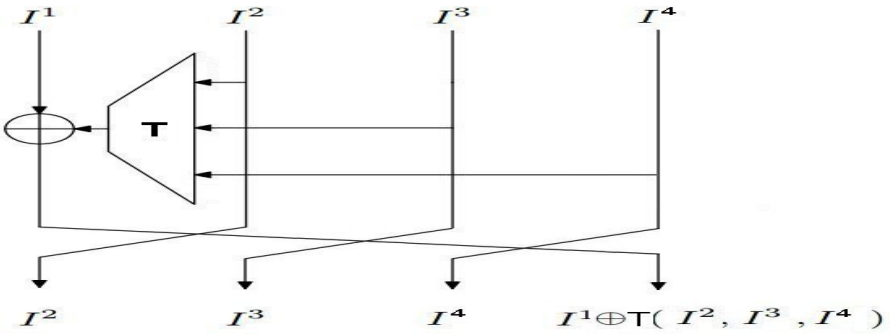


Fig. 4.3. A Contracting Unbalanced Feistel cipher with $k = 4$ branches.

Now T-310 is potentially a lot more complex. All depending on the so called "long-term key" or the internal wiring which takes a form of a plug-in card cf. [45]. The original term is LZS which is an abbreviation of *Langzeitschlüssel* as opposed to weekly/daily keys "ZS" of *Zeitschlüssel* which are perforated cards (with holes punched in them). The LZS would be changed roughly once per year, cf. [44] or only when "necessary" cf. [45].

The main part of an LZS (formal definition below) are two functions D and P which specify two sets of connections. These D, P could be compared to the P-box in DES (which however never changes) or to a Stecker in an Enigma machine (which however would be changed daily). Interestingly, the P-box in DES would be just an internal part at the output of a round function $T()$, while here D and P have extra powers: a possibility to alter the structure of Fig. 4.3 in a very substantial way 4.3. In T-310 D and P work on both inputs and outputs of $T()$. Depending on the exact values of D and P, we will be deviating more or less, or not at all, from a classical unbalanced contracting Feistel with four branches in Fig. 4.3.

4.1 Long-Term Keys - Notation

In this paper we denote an interval of type $\{1, \ldots, 9\}$ by a short notation $\{1-9\}$. Similar but different than the notation $\overline{1-9}$ used in [80].

Definition 4.1.1 (LZS). We call an LZS which is an abbreviation of German *Langzeitschlüssel* cf. Appendix A, a triple (D, P, α) where $D : \{1-9\} \rightarrow \{0-36\}$, $P : \{1-27\} \rightarrow \{1-36\}$ and $\alpha \in \{1-36\}$ which will be studied in Section 14.1.

In addition to basic functions D, P, T, in this paper we will also use notations $\underline{\mathbf{D}}$, $\underline{\mathbf{T}}$ and $\underline{\mathbf{P}}$ which are expect to be derived or constructed from the D, P, T respectively and the exact definition of which will be different in different parts of this paper. For example in the next Section we have $\underline{\mathbf{D}} : \mathbb{F}_2^9 \rightarrow \mathbb{F}_2^9$ cf. Fig. 4.4 which will be D and which re-arranges the order of wires as specified by D. In

most places in this paper we will actually have $\underline{\mathbf{D}} : \mathbb{F}_2^{9+1} \rightarrow \mathbb{F}_2^9$ as depicted on Fig. 5.5, but will also consider for example $\underline{\mathbf{D}} : \mathbb{F}_2^{9+3} \rightarrow \mathbb{F}_2^9$ cf. Section 5.5. The main point of these new notations is to rewrite the cipher description in a new way with new particularly compact notations due precisely to introduction of $\underline{\mathbf{D}}$, $\underline{\mathbf{T}}$ and $\underline{\mathbf{P}}$, and the full formal description of how this works to define a round of a block cipher in a typical setting will be done later in Section 7.1.

4.2 The Importance of Long-Term Keys

Different long-term key wiring functions D and P can make T-310 operate in many different ways. There exist several classes or types of LZS. Historical documents shows clearly that similar to DES [8, 6], T-310 is an extremely carefully designed cipher in terms of how the information propagates inside the cipher.

This is due to the LZS wiring precisely. The historical LZS literature contains tens of pages of detailed analysis and many strong mathematical and combinatorial properties are mandated or shown to hold for specific types of LZS. This has a very strong effect on the entropy of LZS. Initially if we assume that D, P should be injective[5] the number of possibilities for P is $36!/9!$ and for D it is $36!/27!$. Overall the entropy of an injective choice of (D, P) is about 164.6 bits. Interestingly, the designers have imposed so many very strong requirements on (D, P) cf. for example Appendix B, that they have reduced this space to at most 94 bits of entropy, cf. Section 8.5. There also exist a number of special anomalous keys listed in [44]. In this paper we also consider some special keys in Section F.2 and E.1.

4.3 Basic LZS Classification

In this paper many different parts are concerned with study of how the choice of LZS affects the T-310 block cipher and it security. We first look at theory and high-level structural questions here in Sections 4.4-5.5 below, then in Section 5. Then in Section 8 we discuss main historical key classes KT1/KT2 used in the real life with KT1 being the "main" historical version. There is strong evidence in that some 7 keys of type KT1 have been used in practice in the period of 1979-1990, cf. Section 8 and [44]. Then in Section 11, Section 18 and in Appendix C and D we study how the choice of LZS affects the properties of the round function. Finally we study also a number of anomalous special keys in Section 18 and in Appendix E and F.

[5] This is in principle mandatory in T-310, cf. page 115 of [80]. Not true for the so called "testing key" 17 from 1979 which has $P(25) = P(26)$, cf. [44].

4.4 Unbalanced Feistel Reinforced with a Permutation

We start our study of potential and real LZS with a simple example of how a Feistel cipher with 4 branches could be altered or re-wired. For example, we can imagine that we want to reinforce the construction of Fig. 4.3 by a permutation of wires $\underline{\mathbf{D}}$ applied to I^1.

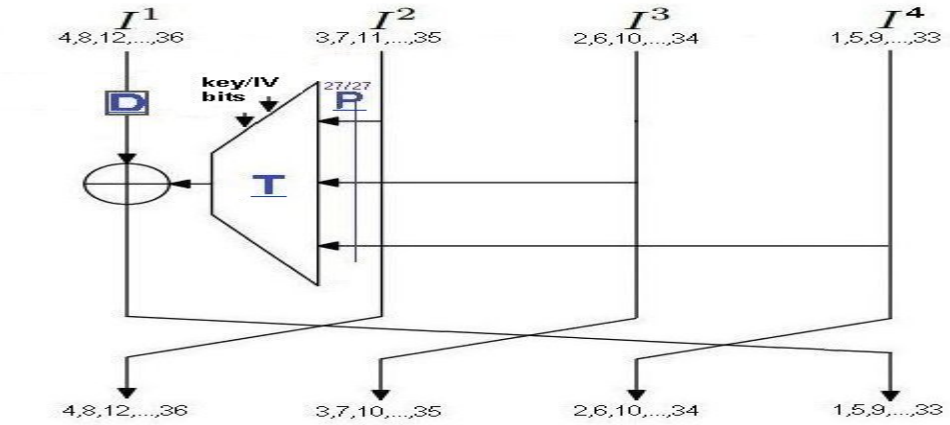


Fig. 4.4. T-310 variant which is like a classical unbalanced Feistel reinforced by D.

Then $\underline{\mathbf{P}}$ could be also a permutation on 27 wires and outputs of $\underline{\mathbf{D}}$ and $\underline{\mathbf{P}}$ will be disjoint (at least in this case).

In particular, if $\underline{\mathbf{D}}$ is just an identity permutation NOT erasing/replacing any bits, and we already specify only bits from I^{2-4} in P, we are back with an ordinary Unbalanced Feistel in Fig. 4.3.

4.5 Permutation D and Chosen Long-Term Key Attacks

There is no evidence the a simple bijective permutation of wires $\underline{\mathbf{D}}$ applied to I^1 would ever be used in a real-life cipher T-310. This is a degenerate special case which we have invented in order to show [later on] that T-310 designers have intentionally and deliberately excluded this case. However it is easy to see that this type of anomalous T-310 encryption as on Fig. 4.4 can be implemented with standard T-310 hardware and that using it would have some very interesting consequences.

A Weak LZS Attack

Imagine that we had D such that D corresponds[6] to bijection between the set $\{4 \cdot 1, \ldots, 4 \cdot 9\}$ and the set $\{4 \cdot 1, \ldots, 4 \cdot 9\}$. This would make encryption round

[6] The actual way to define D is slightly different in [80, 76], it is defined as an application $D : \{1, \ldots, 9\} \rightarrow \{4 \cdot 1, \ldots, 4 \cdot 9\}$ which is a perfectly equivalent definition and we can use the same letter D. In this paper we will also use the notation $\underline{\mathbf{D}}$ which is the corresponding application $\underline{\mathbf{D}} : \mathbb{F}_2^9 \rightarrow \mathbb{F}_2^9$ as on Fig. 4.4 which is induced by D and which re-arranges the order of wires as specified by D. As already explained in most parts of this paper we will have different sort of $\underline{\mathbf{D}}$ for example $\underline{\mathbf{D}} : \mathbb{F}_2^{9+1} \rightarrow \mathbb{F}_2^9$.

work exactly as on Fig. 4.4. Then we get an **unexpected result**. In such a configuration the bit called v_0 in [76] is not used. Consequently **half of the secret key**, which are all the $s_{m,1}$ will NEVER be used during the encryption. Therefore we have discovered a particularly weak class of long-term keys where the effective key size of a "daily" key would be reduced from 230 to[7] 115 bits.

Social Engineering Chosen-LZS Attack on T-310

It could be quite easy for an enemy to convince some employees of Eastern-German state to use an alternative key based on some rumors of compromise of the current key. If we allow such a key to be chosen be the enemy, we can reduce the cipher key size to 115 bits (half of the key is never used).

More Chosen-LZS Attacks

In Section 20 we present another substantially stronger weak-LZS attack, which is also in our opinion more realistic.

Divide And Conquer Attacks

More generally, the strict split of the key between $s1$ and $s2$ parts leads to many other consequences. In Section 7.1 we will show how the description of the cipher with our new notations $\underline{\mathbf{D}}, \underline{\mathbf{P}}$ leads to a functional separation between two halves of the secret key. Consequently, in Section 7.6 we will see that T-310 uses two halves of the key in such substantially different way, so that the attacker can hope to attack them separately.

[7] This observation is in fact a non-obvious result, due to the fact that 5 key parity equations concern $s1$ and only $s1$, and other 5 concern the other half of the key, cf. page 255 [76] or page 117 in [80].

5 Alterations to the Unbalanced Feistel Construction

In this section we look at those types of T-310 long-term setup which are relevant to the variants of T-310 which (according to the current knowledge) are known to be either recommended by the designers or actually used practice in encryption.

5.1 Mainstream T-310: Non-Bijective D

In the most common T-310 cipher versions known from the literature, D is NOT a bijection (however the round function will still be bijective cf. Section 11). Keys with bijective D such as considered in Section 4.4 just above would NOT be compliant with the two classes of long term keys KT1/KT2 described in [80].

More precisely, both types of recommended T-310 keys D according to [80]. will always have following Section 2.2 page 115 that:

$$\exists_{1 \leq i \leq 9} \; D(i) = 0$$

This requirement is mandatory for both standard types of T-310 keys known as KT1 and KT2 which are described in pages 58 and 59 in [80]. Moreover for KT1 keys we always have $D(1) = 0$ cf. page 256 in [76] and page 55 in [80], while for KT2 keys we always have $D(i) = 0$ for some $1 \leq i \leq 7$, cf. page 59 of [80].

5.2 Consequences of $D(i) = 0$

The fact that some $D(i) = 0$ has two important consequences. First, it excludes the attack of Section 4.5. Secondly, it makes that one or more of the $4 \cdot i$ values is **not** attained by D. This corresponds to a more peculiar $\underline{\mathbf{D}} : \mathbb{F}_2^{9+1} \rightarrow \mathbb{F}_2^9$ as on Fig. 5.5 which takes one additional input sometimes called v_0, cf. [76] for which $D(i) = 0$ is a place-holder. Then this one[8] extra input will be[9] substituted by some key bit $v_0 \leftarrow s_{m,1}$ which is constant different in each round according to one part of the secret key.

[8] We can also have $\underline{\mathbf{D}}$ with more than one additional inputs in the so called KT2 case cf. Section 5.5 below.

[9] As we will see below, or at least in all historical cases known to us which are complaint with [80], which could excludes some very special cases, cf. Section 8.

5.3 Unbalanced Feistel vs. KT1 Keys (Most Common Case)

In KT1 case, the function D induces a "quasi-permutation" on 9 bits which we will later call **D** most of which has the function to rearrange the order of bits in I^1, which would be just a way to improve the diffusion of bits inside each branch. However, quite importantly, it is not a permutation and in fact **removes** 1 bit out of 9 and adds one fresh bit which is a constant dependent on the key.

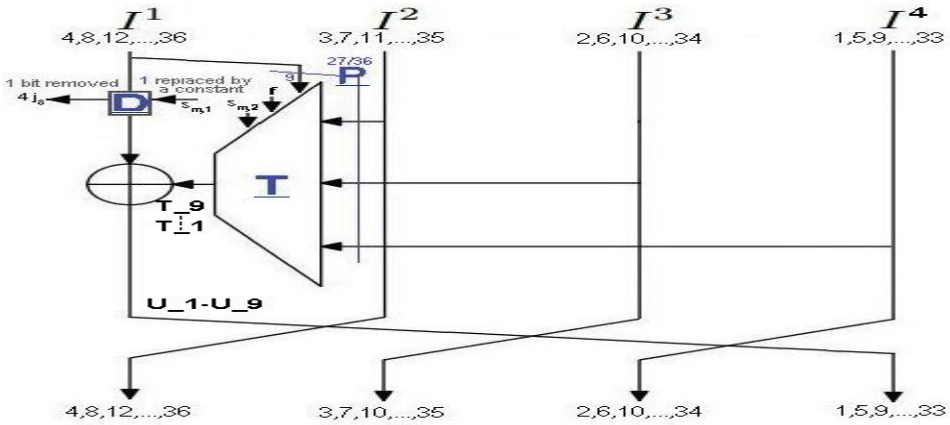


Fig. 5.5. T-310 is NOT exactly a simple unbalanced Feistel scheme. In the common KT1 case, the spec allows to use also bits from the leftmost branch I^1 under a number of highly technical conditions. It also disconnects ONE of the 9 bits in the left branch and replaces it by a key-dependent constant $s_{m,1}$ which is different in each round.

Here the second function is $P : \{1, \ldots, 27\} \to \{1, \ldots, 36\}$ will specify a subset of 27 bits from all the 36 bits from I^{1-4} to be used as inputs of T. As already explained P should be bijective. It appears that in KT1 case there will be always 8 values which are taken by both P and D.

5.4 The High-Level Structure of KT1 Keys

The long-term key D/P are not fully specified in [80, 76], instead a complex set of constraints which D and P must satisfy is given. cf. Appendix B. Now, for all KT1 keys of [80] we observe that just one[10] bit from leftmost branch I_1 gets disconnected and it is replaced by a constant[11] then it is possible to see that P must include at least one input from the leftmost branch or 1 bit would be lost and it would become impossible to build a bijection.

In addition, the criteria which the KT1 long term key D/P should satisfy described in [80, 76] guarantee that **all** the 9 bits from the leftmost branch will be outputs of $P()$ of type $4 \cdot l$ and therefore must and will be used as inputs to

[10] In [76] it is also exactly one bit, and precisely the one with index equal to $4j_8$, which number is part of one way to define a long-term key in [76].

[11] This constant will be later seen to be one of the key bits and it will be different in different rounds.

T. Therefore in a typical version of T-310 as per [76], T will have as many as 9 additional inputs from the leftmost branch J^1, which interestingly are those 9 bits which would be traditionally **forbidden** to use in traditional unbalanced Feistel ciphers, cf. Fig. 3.2 above. This means that there will be serious difficulties in making sure that our "tweaked" generalized Feistel is still going to be a bijection[12]. Depending on the choice of D, P in T-310 decryption[13] is a lot less trivial process enabled by additional properties. An additional internal "triangular" structure is now badly needed in order to enable these 9 bits of I^1 which "theoretically" now depend on themselves, to be computed - one by one - in a specific order, cf. Section C. A detailed example of how inversion can be performed is provided in Appendix C.9.

Overall these more or less important modifications which depart from a traditional Feistel structure impose a lot of strong constraints which are bound to have, very important consequences for the cryptanalyst and will heavily limit the complexity of T. This also makes the current Luby-Rackoff theory e.g. [65–67] not exactly relevant to the security of this cipher or requires a more adapted theory to be yet developed, and the cipher will rather require a substantially larger[14] number of rounds than other similar ciphers [e.g. RC2] to be provably secure or secure against "generic" attacks.

[12] This question is studied in Appendix C.

[13] Decryption in the sense of computing the previous states of the T-310 generalized Feistel variant is **not** needed in normal operation of the cipher. However for this cipher to be bijection, is needed as a structural property which was clearly imposed by designers of T-310, cf. Appendix C, and this for some very good (security) reasons such as preventing the entropy of the cipher state from being depleted by iteration.

[14] The opposite could also be argued: Though the strong internal structure of T in T-310 certainly leads to imperfect/poor diffusion, a well-chosen $\underline{\mathbf{D}}$ - NOT required by the theory such as [65] - could make it substantially stronger and avoid attacks such as splitting the cipher in 2 loosely connected parts, cf. Fig. 3-4 in [38, 39].

5.5 Alterations to the Unbalanced Feistel Construction with KT2 Keys

It seems rather obvious that replacing more than 1 bit by a constant would weaken the cipher. It is also worth noting that for KT2 keys as described in [80] more bits can be disconnected, but again, only one is replaced by a constant [at least for key 15 from [44]]. This type of keys is not yet well understood.

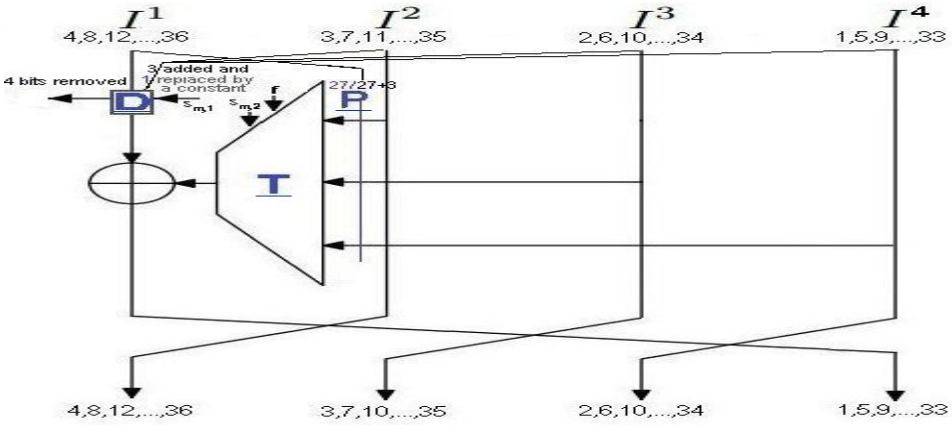


Fig. 5.6. Connections of T-310 when using the KT2 key 15 from [44].

Comparison KT1 vs KT2. Keys of type KT1 and KT2 differ very substantially. For example for KT1 keys the outputs of $\underline{\mathbf{D}}$ and $\underline{\mathbf{P}}$ will have 8 numbers in common. In contrast for KT2 keys the two sets of outputs of $\underline{\mathbf{D}}$ and $\underline{\mathbf{P}}$ will be always disjoint, cf. page 59 in [80].

An example of a real-life KT2 key is the key 15 from 1979 in [44] which we have verified to satisfy all the conditions from pages 59-60,114-115 and 117 in [80]. In this example only 3 bits of I^1 are used in T, while 9 are typically used for KT1 long-term keys (which are illustrated in Fig. 5.5). In KT1 we always have $\underline{\mathbf{D}} : \mathbb{F}_2^{9+1} \to \mathbb{F}_2^9$, while in KT2 we may have $\underline{\mathbf{D}} : \mathbb{F}_2^{9+3} \to \mathbb{F}_2^9$ but also for example we could have $\underline{\mathbf{D}} : \mathbb{F}_2^{9+2} \to \mathbb{F}_2^9$ cf. Appendix E.3.

6 Detailed Description of T-310

Given Fig. 6.7 and Fig. 5.5, in order to fully specify the cipher T-310 we need:

1. To specify the u_0 the initial 36-bit state I^{1-4} of the block cipher which is a constant equal to 0xC5A13E396, cf. [76].
2. To specify D, P fully, cf. Section 8 and how they affect the exact connections inside one round of the block cipher ϕ, cf. Sections 7.5, 9 and Section 11) which is further extended in Appendix C.
3. To specify the internals of the round function $T : \{0,1\}^3 \times \{0,1\}^{27} \rightarrow \{0,1\}^9$ cf. Section 9 below.
4. To specify how the 3 bits of key and IV $(f_m, s_{m,1}, s_{m,2})$ used by $\underline{\mathbf{D}}$ and T are generated, for each round $m \geq 1$, cf. Section 13.2 and 13.4.
5. To specify the encryption component: how bits from the state of our iterated block cipher are extracted, cf. Section 14.1 and used to encrypt the plaintext 5 bits a time, cf. Section 16.

On Fig. 6.7 below we show how all these things come together.

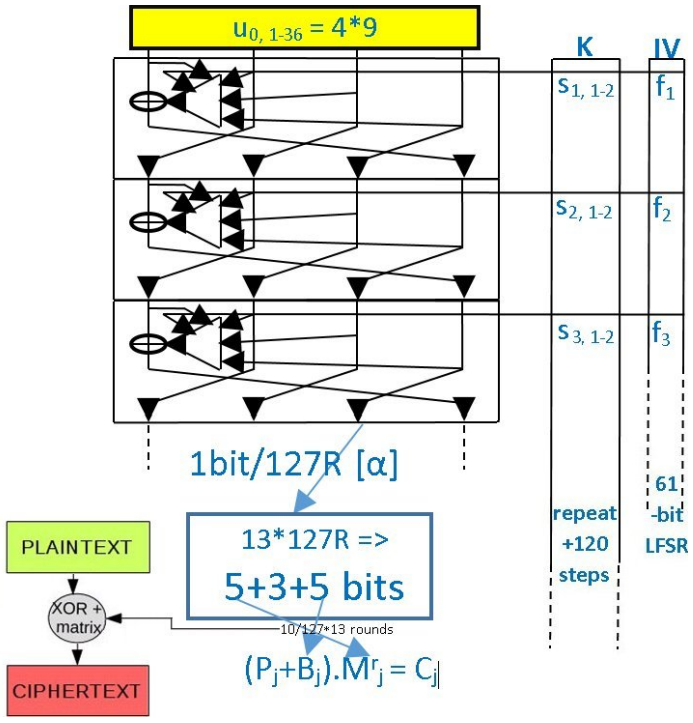


Fig. 6.7. T-310 Cipher.

7 Construction of One Encryption Round ϕ

We start by a high-level description where we introduce our new notations.

7.1 Compact High-Level Description of One Round ϕ

We denote by $u_{m,j}$ the state of the block cipher after m rounds of encryption $m = 0, 1, \ldots$ and $j = 1 \ldots 36$. Then each round is computed as:

$$(u_{m,1-36}) = \phi\left(s_{m,1}, s_{m,2}, f_m;\ \ u_{m-1,1-36}\right)$$

Here $\phi : \{0,1\}^3 \times \{0,1\}^{36} \to \{0,1\}^{36}$ is one full round of encryption with 3 bits of key+IV per round which is written here using our new compact notations as in Fig. 4.3 and Fig. 5.5:

$$\left(u_{i,I^{1-4}}\right) = \phi\left(s_{m+1,1}, s_{i,2}, f;\ \ u_{m,I^1},\ u_{m,I^2},\ u_{m,I^3},\ u_{m,I^4}\right) =$$

$$\left(\ u_{m,I^2};\ \ u_{m,I^3};\ \ u_{m,I^4};\ \ \underline{\mathbf{D}}(s_{m+1,1};\ \ u_{m,I^1}) \oplus \underline{\mathbf{T}}\left(f, s_{m+1,2},\ \ \underline{\mathbf{P}}(u_{m,I^{1-4}})\right)\ \right)$$

Below we explain our new notations $\underline{\mathbf{D}}$, $\underline{\mathbf{T}}$ and $\underline{\mathbf{P}}$ not previously used for T-310.

7.2 Definition of $\underline{\mathbf{P}}$

Then $\underline{\mathbf{P}} : \{0,1\}^{36} \to \{0,1\}^{27}$ is a permutation of wires which also defines which wires are not[15] used (depending on cases). It is defined as:

$$\underline{\mathbf{P}}_k\left(u_{m,1}, u_{m,2}, u_{m,3}, \ldots, u_{m,36}\right) = u_{m,P(k)} \quad \text{for any} \quad k = 1 \ldots 27$$

Therefore we have:

$$(v_1, v_2, v_3, \ldots, v_{27}) = \left(u_{m,P(1)}, u_{m,P(2)}, u_{m,P(3)}, \ldots, u_{m,P(27)}\right)$$

7.3 Definition of $\underline{\mathbf{T}}$

The definition of $\underline{\mathbf{T}} : \mathbb{F}_2^{29} \to \mathbb{F}_2^9$ is the same as T with order of outputs inversed, i.e. $\underline{\mathbf{T}}_i(f, s_2, v_{1-27}) \stackrel{def}{=} T_{10-i}(f, s_2, v_{1-27})$, which function $T : \mathbb{F}_2^{29} \to \mathbb{F}_2^9$ will be defined in Section 9.

7.4 Definition of $\underline{\mathbf{D}}$ in KT1 Case

We focus primarily on KT1 case represented in Fig. 5.5. In this case, $\underline{\mathbf{D}}$ is near-permutation of 9 wires with one additional bit of input $s_{i,1}$ which is the bit[16] which "replaces" the bit which[17] is "removed" in KT1 case.

In KT1 case (and not in KT2 case) we have a particularity that outputs of $D()$ are always expected to be multiples of 4, and are of the form $D(a) = 4 \cdot b$ with $b \in \{0, \ldots, 9\}$. Here we distinguish two types of inputs for $\underline{\mathbf{D}}$. First, the the case $b = 0$, which corresponds to replacing one bit by a constant which is not from I^1 but equal to $s_{i,1}$. Then we have all the other multiples $4 \cdot b$ with $b \neq 0$ which are exactly a subset of those 8 out of 9 bits of I^1 which are used.

Overall, our permutation D induces a function $\underline{\mathbf{D}} : \{0,1\}^1 \times \{0,1\}^9 \to \{0,1\}^9$ defined as follows, where we use a quite unusual numbering of inputs to keep it partly compatible with [76] and Fig. 5.5.

$$\underline{\mathbf{D}}_i\left(s_1;\ u_4, u_8, u_{12}, \ldots, u_{36}\right) = s_1 \quad \text{when} \quad D(i) = 0$$

$$\underline{\mathbf{D}}_i\left(s_1;\ u_4, v_8, u_{12}, \ldots, u_{36}\right) = u_{D(i)} \quad \text{when} \quad D(i) \neq 0$$

---

[15] Specific example of missing bits are listed in Table 1 page 26 which leads to some important differential properties, cf. Section 12.2.

[16] This bit was called v_0 in [76].

[17] It happens in fact at position equal to $4j_8$ following the notations used in [76].

7.5 Summary: Main Part of ϕ

Putting it all together, we have 9 new bits created at each round which we will call U_{1-9} as defined here below. We also recall that these 9 bits will be shifted to branch I^4 now, cf. Fig 5.5 and therefore we have:

$$(u_{m+1,1}, u_{m+1,5}, u_{m+1,9}, \ldots, u_{m+1,29}, u_{m+1,33}) \stackrel{def}{=}$$
$$(U_1, U_2, U_3, \ldots, U_8, U_9) \stackrel{def}{=}$$
$$\underline{\mathbf{D}}(s_1;\ \ u_{m,I^1}) \oplus \underline{\mathbf{T}}\left(f, s_2,\ \ \underline{\mathbf{P}}(u_{m,I^{1-4}})\right) =$$
$$\big(u_{m,D(1)} \oplus T_9(f, s_2, u_{m,P(1-27)}),$$
$$u_{m,D(2)} \oplus T_8(f, s_2, u_{m,P(1-27)}),\ \ u_{m,D(3)} \oplus T_7(f, s_2, u_{m,P(1-27)}), \ldots$$
$$\vdots$$
$$\ldots, u_{m,D(8)} \oplus T_2(f, s_2, u_{m,P(1-27)}),\ \ u_{m,D(9)} \oplus T_1(f, s_2, u_{m,P(1-27)})\big)$$
$$\text{where by convention} \quad u_{m,1} \stackrel{def}{=} s1$$

Moreover for most historical T-310 keys we have $D(1) = 0$ which gives:

$$(u_{m+1,1}, u_{m+1,5}, u_{m+1,9}, \ldots, u_{m+1,29}, u_{m+1,33}) =$$
$$\underline{\mathbf{D}}(s_1;\ \ u_{m,I^1}) \oplus \underline{\mathbf{T}}\left(f, s_2,\ \ \underline{\mathbf{P}}(u_{m,I^{1-4}})\right) =$$
$$\big(s_1 \oplus T_9(f, s_2, u_{m,P(1-27)}),$$
$$u_{m,D(2)} \oplus T_8(f, s_2, u_{m,P(1-27)}),\ \ \ldots$$
$$\ldots \qquad\qquad u_{m,D(9)} \oplus T_1(f, s_2, u_{m,P(1-27)})\big)$$

Notes on notation: We use the letter ϕ following [76] and we consider that $\phi : \mathbb{F}_2^{3+36} \to \mathbb{F}_2^{36}$. Similar but different notations are used in [80]: except that it uses a capital letter Φ which is written in handwriting and which looks like neither ϕ nor Φ which could lead to some confusion. Then this letter Φ and other similar notations are used at many places in a very mathematical style which privileges compact notations over trying to avoid any ambiguity[18]. In this paper we will also privilege compact notations and when some 3 bits are fixed in some particular encryption context we will consider that we have a function $\phi : \mathbb{F}_2^{36} \to \mathbb{F}_2^{36}$ which will typically be a permutation and which following the habit of [80] will still be denoted by ϕ.

7.6 A Potential Serious Vulnerability - Divide And Conquer Attacks on Key Space

There is something interesting which is revealed by our new notations $\underline{\mathbf{D}}$ and $\underline{\mathbf{P}}$. We observe that ONLY half of the key bits (120 bits) will ever be used to

[18] In [80] Φ will typically denote our permutation ϕ where the $(s1, s2, f)$ bits are fixed, OR when all possible 8 choices of $(s1, s2, f)$ are considered. Given a fixed (P, D) we have exactly 8 possible permutations which are sometimes denoted by $\Phi_0, \ldots, \Phi_7$. At other places the notation Φ_T is used to distinguish the permutation Φ of the round function of T-310 cipher from one defined for a different cipher, e.g. page 47 in [80].

form any of the $s_{i,1}$ used by $\underline{\mathbf{D}}$, and another disjoint half of key bits is used to form the $s_{i,2}$ used in $\underline{\mathbf{P}}$. This fact alone is potentially a serious **design flaw** in T-310 cipher and suggests there might be some divide-and-conquer or guess-then-determine attack, where initially the attacker would guesses only half of the key bits etc and confirm this guess without knowing the other half. For example we could just compile statistics on how frequently flipping a bit i flips also bit of the form $i + 4k$ for any number of the rounds of this cipher, and realize that this depends on $\underline{\mathbf{D}}$ primarily and if we guess this half of the key we should get a specific recognizable pattern.

We don't know if such attack will be efficient, and the attacker currently does NOT easily get access to see all the flipped bits, cf. Section 14.1-16, BUT it would be **extremely easy** for the designers to **avoid** any such attack on T-310 by mandating a sequence derived from both halves of the key for both $\underline{\mathbf{D}}$ and $\underline{\mathbf{P}}$.

8 Long Term Keys D, P

The long term key D/P are not fully specified in [80, 76], however some historical examples of D/P can be found in [44]. It appears that the "main" historical versions of T-310 were primarily using KT1 keys. KT1 keys are defined in [80]. There is strong evidence in that some 7 keys of type KT1 have been used in practice in the period of 1979-1990, cf. Section 8 and [44]. Then there exists another substantially less popular class of long-term keys KT2. The sources and [80, 44] list only 1 such key which is number 15 from 1979 and we are not sure if this was ever actually used to encrypt any substantial volume of communications.

8.1 Example of D, P of Popular Type KT1

No example of actual long term key D/P is given in [80, 76]. Instead a set of peculiar constraints on D/P are specified. In [76] only the so called KT1 class of keys of [80] is specified and it is not specified[19] exactly. In Appendix B we provide a complete specification of this class. In [80], another class of keys KT2 is specified. Both these classes of keys are clearly meant to make Fig. 5.5 have the desired properties such as invertibility and possibly other which need yet[20] to be elucidated.

Several real-life historical examples of keys D/P from 1977-1990 can be found in [44] which are given numbers[21] of type `Der Langzeitschlüssel 14: (1979).`

[19] We demonstrate this fact in Section F.1.

[20] At this moment we are far from being able to make the full assessment of the impact of these criteria on the strength of T-310, and apparently there may exist other alternative sets of rules, cf. [44].

[21] It appears that only keys 14,15,21,26,30,31,32,33 in [44] are for T-310. Inside these keys only some are long-term keys for T-310 and other are apparently keys for different other East German encryption machines different than T-310 which are also listed there. Circumstantial evidence seems to show that this numbering is consistent with earlier documents, for example in page 42 of [80] we read that key 14 are of type KT1 and key 15 are of type KT2, which is also true for keys found in [44].

In our research by default we will use the following real-life long-term key number 26 from [44]. We have carefully checked that key 26 belongs to the so called KT1 class which is fully described in [80] also [not completely] described in [76].

```
//Der Langzeitschl\"{u}ssel 26: (1981)
D=0,28,4,32,24,8,12,20,16 P=8,4,33,
16,31,20,5,35,9,3,19,18,12,7,21,13,23,25,28,36,24,15,26,29,27,32,11,
alpha=4
```

We have also tested all the other keys in [44] and we have verified that keys which belong to class KT1 are only and exactly those numbered 14,21,26,30,31,32,33.

8.2 Properties of KT1 Keys

KT1 keys mandate a sort of total ordering on the outputs of D: there exist 8 pairwise distinct exist integers $j_1, \ldots, j_8 \in \{2, \ldots, 9\}$ such that $D(j_1) = 4$ and $D(j_k) = 4j_{k-1}$ for any $k = 2 \ldots 8$, cf. Appendix B and [80, 76]. Other important properties of KT1 keys are studied in Section 5.3, Section 5.4 and in Appendix C.

8.3 KT2 Key Class

This type of keys is not yet well understood. The specification of class KT2 is substantially more complex than KT1, it is split in several parts which can be found on pages 59-60,114-115 and 117 in [80]. We have checked that the only known authentic key of type KT2 which [22] is key 15 from 1979 in [44] does indeed belong to KT2. This key is as follows:

```
D=0,4,17,12,35,32,2,24,20 P=15,13,33,
34,6,8,5,3,9,18,14,22,28,30,21,31,7,25,26,16,27,11,23,29,19,1,36
```

It seems rather obvious that replacing more than 1 bit by a constant would weaken the cipher. It is also worth noting that for KT2 keys as described in [80] more than one bit will be disconnected contrasting with KT1 keys, but in both KT1/KT2 cases **only one** is replaced by a constant (this is at least for key 15 from [44]).

The keys of type KT2 are also studied in Section 5.5, Appendix D and Appendix E.

8.4 Other Keys and Key Classes

We have tested all the keys which are indicated as keys for T-310 in [44] and some such keys do not belong to neither of two classes KT1/2. Some of these keys such as 27/28 are clearly indicated as "anomalous keys for testing", others such for example as key 29, look like other similar KT1 keys, yet do not satisfy all the KT1 conditions enumerated in page 256 of [76]. In Appendix F.1 we present another key which is also "almost" but not quite KT1. Similarly in Appendix E.2 we present several keys which are "almost" but not quite KT2.

[22] We also study this key in Fig. 5.6 of Section 5.5 page 15.

8.5 Key Sizes for the Long Term Keys

According to page 56 in [80], the entropy of (D, P) belonging to class KT1 (also studied in [76]) is between 78.1 and 79.7 bits. For class KT2 it would be between 76.1 and 89.2 bits. It appears that the designers have not attempted or had no capacity to evaluate the sizes of these sets more precisely in 1980s, for example due to a limited computing power. We need to add to this a third not yet studied part of long-term key which is called α and is simply a number between 1 and 36, cf. Section 14.1 below. However, the number of possibilities for α is reduced to about 30 cf. page 117 in [80]. Therefore the entropy of α is only about 4.9 bits.

Overall, the union of both classes of KT1/KT2 keys with a specification of α will have approximately between 2^{83} and most 2^{94} elements. Thus the effective key size for the long-term key for T-310 is between 83 and 94 bits.

Remark: This is **NOT very large** compared to other historical ciphers. For example the effective long-term key size for Enigma is hundreds of bits (88 bits per unknown rotor), and for GOST cipher it is about 354 bits, cf. [33]. This small key size suggests that a cryptanalyst could maybe be able to break T-310 also when the long-term setting are unknown.

8.6 Long Term Keys vs. Security

It is clear that the choice of D, P is crucial for the security of this cipher, in the same way as the choice of the bit permutation which occurs after the round function is crucial for the security of DES, cf. slide [10] and in the same way as a bad choice is what makes GOST weak, cf. [34], and leads to some very good attacks, cf. again Fig. 3-4 in [39] and all the attacks of [38–40].

9 Description of T

The standard method to define the round function of T-310 is to define $T : \mathbb{F}_2^{2+27} \to \mathbb{F}_2^9$ as follows:

$$\begin{aligned}
T_1\,(f;\ s_2;\ v_{1-27}) &= \mathbf{f}\\
T_2\,(f;\ s_2;\ v_{1-27}) &= T_1 \oplus Z(\mathbf{s_2}, v_{1-5})\\
T_3\,(f;\ s_2;\ v_{1-27}) &= T_2 \oplus v_6\\
T_4\,(f;\ s_2;\ v_{1-27}) &= T_3 \oplus Z(v_{7-11})\\
T_5\,(f;\ s_2;\ v_{1-27}) &= T_4 \oplus v_{13}\\
T_6\,(f;\ s_2;\ v_{1-27}) &= T_5 \oplus Z(v_{14-19}) \oplus \mathbf{s_2}\\
T_7\,(f;\ s_2;\ v_{1-27}) &= T_6 \oplus v_{20}\\
T_8\,(f;\ s_2;\ v_{1-27}) &= T_7 \oplus Z(v_{21-26})\\
T_9\,(f;\ s_2;\ v_{1-27}) &= T_8 \oplus v_{27}
\end{aligned}$$

Here $Z : \mathbb{F}_2^6 \to \mathbb{F}_2$ is a Boolean function defined in Section 10.1. We also recall the input naming of [76]: $e_0 = f$, then $e_1 = s_2$, and then $e_2 = v_1$ up to $e_{28} = v_{27}$.

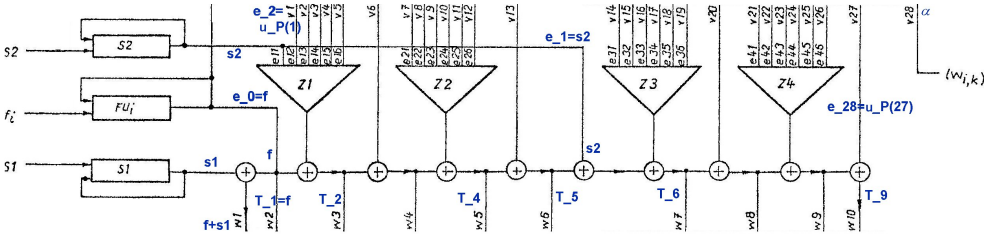


Fig. 9.8. A zoom inside the "complication unit" drawing cf. [46], which can be mapped to the description of $T()$ and e_i notations of [76].

One can also view ϕ and T as a stateful system which operates on 9 bits:

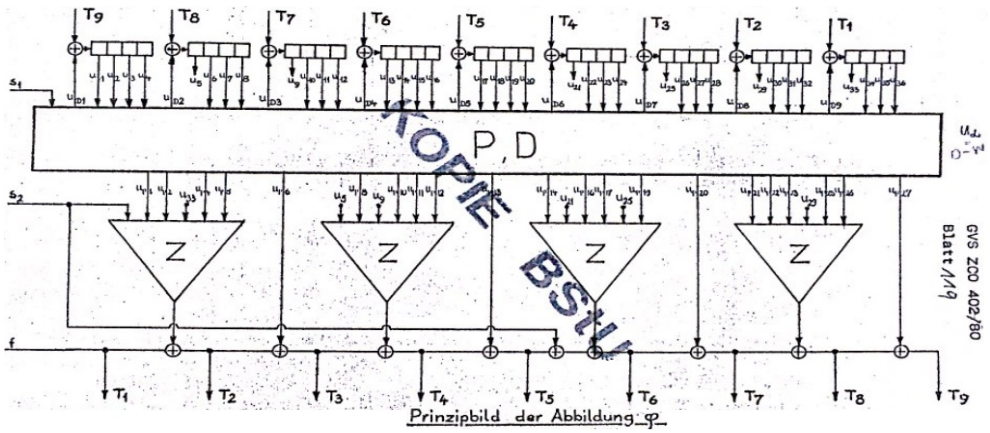


Fig. 9.9. T-310 round function seen as a stateful component $\mathbb{F}_2^{3+9} \to \mathbb{F}_2^9$ with 36 memory bits, cf. page 119 in [80] which also mandates a certain subset W of 6 special bits e.g. u_5 and u_{33} cf. page 117 in [80] and Fig. 2.11 in Appendix B.

9.1 Observations on $T()$

In general, in Luby-Rackoff theory, for a block cipher to be secure it is required for T to be very complex. What strikes us in T-310, is that the the round function T is extremely simple and highly structured, which again is clearly a sort of inevitable consequence of the fact that the bits of I^1 would need to be computed in a specific order.

It is also something that reveals a highly regular internal structure with weak to inexistent diffusion properties [the diffusion is the job of **D** and **P**] and an inherently **sequential** character of T-310 computations. In the very definition of $T()$ above, there is exactly one natural order of computing the output bits $T_1, \ldots, T_9$. Accordingly, T-310 can also be viewed as a complex stream cipher with non-linear feedback which generates one new bit a time in this order and where the amount of non-linearity or Multiplicative Complexity [19] which is less than just one application of Z per new $u_{i,j}$ state bit generated.

9.2 Observations on $T()$ Combined with Final XORs

We can now combine together this "straight-line order" structure inside T with the next step which is done after $T()$is computed: the XOR with bits of the left branch I^1 which leads to the creation [23] of 9 new bits denoted by U_i in Section 7.5. It is easy to see that these 9 bits are computed in the exact order $U_9, \ldots, U_1$ and that the following equations hold:

$$\begin{aligned}
U_1 \oplus s_1 &= U_2 \oplus u_{D(2)} \oplus u_{P(27)} \\
U_2 \oplus u_{D(2)} &= U_3 \oplus u_{D(3)} \oplus Z_4(u_{P(21-26)}) \\
U_3 \oplus u_{D(3)} &= U_4 \oplus u_{D(4)} \oplus u_{P(20)} \\
U_4 \oplus u_{D(4)} &= U_5 \oplus u_{D(5)} \oplus Z_3(u_{P(14-19)}) \oplus s_2 \\
U_5 \oplus u_{D(5)} &= U_6 \oplus u_{D(6)} \oplus u_{P(13)} \\
U_6 \oplus u_{D(6)} &= U_7 \oplus u_{D(7)} \oplus Z_2(u_{P(7-12)}) \\
U_7 \oplus u_{D(7)} &= U_8 \oplus u_{D(8)} \oplus u_{P(6)} \\
U_8 \oplus u_{D(8)} &= U_9 \oplus u_{D(9)} \oplus Z_1(s_2, u_{P(1-5)}) \\
U_9 \oplus u_{D(9)} &= f
\end{aligned}$$

Here we distinguish Z_1, Z_2, Z_3, Z_4, which by definition are 4 copies of the same Boolean function $Z()$ defined in Section 10.1, and which are computed in this exact order Z_{1-4}.

9.3 Vulnerabilities of the Whole T Component

In Section 12.1 below we show that there is a serious problem in all versions of T-310: the round function will systematically omit to use some 9 bits regardless of the long-term key. This has serious consequences for the security of T-310 in particular against differential attacks, cf. Section 12 below.

[23] These bits will become $(u_{33}, u_{29}, \ldots, u_5, u_1) = (U_1, U_2, \ldots, U_9)$ at the input of the **next** round ϕ.

10 The Non-Linear Component of T-310

The only[24] non-linear component in T-310 block cipher and keystream generation process is a simple Boolean Function Z with 6 inputs.

10.1 Description of the Boolean Function Z

Following page 113 in [80] we have:

$$\begin{aligned} Z(e_1,e_2,e_3,e_4,e_5,e_6) = \\ 1 \oplus e_1 \oplus e_5 \oplus e_6 \oplus \\ e_1e_4 \oplus e_2e_3 \oplus e_2e_5 \oplus e_4e_5 \oplus e_5e_6 \oplus \\ e_1e_3e_4 \oplus e_1e_3e_6 \oplus e_1e_4e_5 \oplus e_2e_3e_6 \oplus e_2e_4e_6 \oplus e_3e_5e_6 \oplus \\ e_1e_2e_3e_4 \oplus e_1e_2e_3e_5 \oplus e_1e_2e_5e_6 \oplus e_2e_3e_4e_6 \oplus \\ e_1e_2e_3e_4e_5 \oplus e_1e_3e_4e_5e_6 \end{aligned}$$

which is the same as on page 256 in [76] except that the constant 1 is missing in [76]. In contrast, online sources [46] seem to provide incorrect[25], information.

Another Vulnerability. The fact that the same Boolean function is used everywhere is of course a potential vulnerability. For example, using the same Boolean function many times in an LFSR-based stream cipher combined with self-similarity properties which allow the whole inputs of these functions to repeat at a later time during the encryption process is known to lead to some extremely fast key recovery attacks on certain ciphers. An example of such attack can be found in [41] which attack was further improved in [51].

[24] Final "double" one-time pad character encryption module of T-310 is also non-linear, cf. Section 16, which fact however will be out of scope for most of the security analysis, as the main or final step in many attack will work starting from a pure block cipher property.

[25] The ANF published in [46] appears to be incorrect and more precisely two cubic terms $e_1e_3e_4$ and $e_1e_3e_6$ are missing in [46]. In fact all the three representations of $Z()$ provided in [46]: ANF, truth table and DNF, are pairwise inconsistent, so it is difficult to consider [46] to be a reliable source.

11 Properties of T-310 Round Function ϕ

The original documents on T-310 contain a great deal of claims about various mathematical and cryptographic properties of ϕ and various combinations of permutations derived from or based on ϕ. Here crucial properties to be studied will be vulnerability to Differential Cryptanalysis (DC) cf. Section 12 below, truncated differentials, linear cryptanalysis and bi-linear and multinear flavors which will be particularly interesting here cf. [12, 10] and many other, ElimLin and advanced variants thereof [9, 18, 79], etc. We plan to study **all** these in future revisions of this paper as all of these deserve a serious consideration for a serious government encryption systems such as T-310. For example on page 56 of [80] it is very clearly specified that ϕ should be a bijection, which question is related to some strong DC attacks as we will see below.

11.1 Is One Encryption Round ϕ a Permutation?

From a purely functional engineering perspective **nothing** forces the round function to be invertible, and this property is simply not required for the normal operation of the cipher. However it is possible to see that the security consequences of ϕ being not a permutation would be severe, and comparable to some spectacular so called "Vanishing Differential Attacks" which have been for example used a lot by hackers in the last 20 years to extract secret keys and clone mobile phone SIM cards, see Appendix C.1. It is also clear that the designers of T-310 and other East German ciphers in 1970s have done a great deal of effort to make sure that T-310 keys are always bijective therefore making this type of powerful differential attacks impossible. We refer to Section 20 to see that this sort of attack would also be truly devastating for T-310. In order to show that T-310 was designed to resist this attack, in Appendix C we provide complete and detailed mathematical proofs to show that all KT1/KT2 keys lead to a bijective round ϕ, which also therefore proves the security of all known historical versions of T-310 against "Vanishing Differential" attacks.

11.2 Another Result on ϕ

The following result is claimed to hold (apparently) for all long-term keys for T-310, cf. page 49 in [80].

Theorem 11.2.1 (Local injectivity result for ϕ^4). For four rounds ϕ^4 if we fix the block input u on 36 bits, and vary the 12 of the key and IV bits, we obtain 2^{12} pairwise distinct $\phi(u)$ values on 36 bits.

This result also holds for 1,2 and 3 rounds of T-310, cf. page 49 in [80].

12 Differential Attacks and Differential Vulnerabilities in T-310

Differential cryptanalysis is one of the oldest one of the most powerful and generally applicable attacks on block ciphers, cf. [11, 40, 8, 38]. We refer to [40] for a survey and pointers on the "confidential" history of development of differential cryptanalysis. In [76] we read that it is not clear if the designer knew about linear and differential cryptanalysis (LC/DC). In modern cryptanalysis it is very clear that Feistel schemes are naturally susceptible to Differential Cryptanalysis (DC) and there exist many differential attacks which are structural attacks primarily on the way in which the components are connected ignoring totally the exact content of the components cf. [67], or ignoring it in approximation, cf. [38–40]. However for other ciphers, the properties of the S-boxes will be very important for DC attacks. There exist also many advanced forms of differential cryptanalysis [29, 54, 59, 13] where it can be combined with other attacks such as software algebraic attacks [34, 32, 47]. In this Section we give some first results on DC attacks on T-310. First we are going to show there are good reasons to claim that all version of T-310 are inevitably quite vulnerable to such attacks.

12.1 Missing Bits - A Potential Vulnerability of T for any P

We observe that in the situation of T-310 in Fig. 5.5 there is no reason whatsoever why the number of inputs of T would be limited to 27. Interestingly, even though 9 extra inputs to T have been added, the total number of inputs was kept at 27, i.e. some of the 27 inputs from I^{2-4} will **not be used**. This is clearly not a good idea. Certain bits or differences on these bits have no effect on the output of T and therefore less bits in subsequent rounds will be affected.

Here are the exact bits which are not used for different historical keys:

Table 1. List of 9 bits which are NOT used by the round function T for different long-term keys

key nb	bits which are not $P(j)$
16	4,8,12,16,20,24,28,**31**,36
14	2,3,7,10,11,17,22,30,31
21	10,14,15,17,22,23,27,30,35
26	1,2,6,10,14,17,22,30,34
30	10,13,14,15,17,22,26,31,34
31	2,3,11,14,17,19,27,31,34
32	2,3,6,7,17,19,26,31,35
33	7,11,14,15,19,23,27,31,34
15	2,4,10,12,17,20,24,32,35

Conjecture 12.1.1 (Missing Bits in T-310). We conjecture that 9 bits are systematically missing, which is clear for KT1 keys due to criteria listed in [76], and which is less evident for KT2 keys such as key 15.

Remark: This property of 9 missing bits seems to be an artefact of the historical process which has lead to the development of T-310 such as backwards compatibility or/and the temptation to design a cryptosystem which is simple/elegant/ or about which some interesting properties can be shown to hold, cf. for example Section 1.5 in [80].

12.2 Missing Bits - Applications

This property of Table 1 and Conjecture 12.1.1 is very likely to weaken the cipher against various differential attacks such as [38, 39] and in general. If the reader doubts whether the fact of not using all available bits in each round T degrades the security against differential cryptanalysis, consider the following example.

Fact 12.2.1 (A 3R Property for key 30). For example it is easy to see that for key 30, if we flip bit 13, only one bit 16 is flipped after 3 rounds.

This is an extremely rare example of a differential where the number of active bits does not grow.

Remark: Our missing bits property can also have positive consequences, for example it can be used to prove the certain correlation attacks will not work, see for example Thm. 24.0.1 page 53.

12.3 Examples of Differential Attacks on T-310

Our student Matteo Scarlata (with help of another student Mario D'Onghia who also developed another fast parallel tool) has developed a software tool written in Python cf. Section L and [75] for discovery of differential attacks on T-310. Here are some preliminary results for key 26.

Table 2. Some "good" differential properties for T-310 with LZS-26

key nb	rounds	input → output	proba
26	4	[22] → [18]	$2^{-1.95}$
26	7	[1] → [1,12]	$2^{-3.19}$
26	8	[6] → [18]	$2^{-5.85}$
26	10	[30] → [36]	$2^{-6.8}$
26	13	[25] → [18]	$2^{-9.9}$
26	17	[26] → [11,23]	2^{-16}

12.4 Differential Vulnerabilities with Different IVs

More possibilities for stronger attacks exist if we allow two different IVs. We have for example discovered that:

Fact 12.4.1 (A chosen-IV differential property for T-310 block cipher). Consider 2 parallel encryptions with the T-310 block cipher and our example of a long-term key specified in Section 8. Consider two encryptions with the same key, same input and two different IV, one IV is composed of all 0s ($\forall_i f_i = 0$)

and the other IV is all 1s. Then the probability that the outputs difference has HW equal to 35 out of 36 bits (strong result, the output differential is almost fixed) for 4 rounds is as low as $2^{-8.1}$ instead of around $36 \cdot 2^{-36}$ expected for an ideal block cipher.

Related Research: The designers have clearly mandated that to flip 0 bits (have a collision) for 4 rounds and for 2 different IVs cannot happen, cf. Thm. 11.2.1. Now it seems also that 36 bit difference does not happen either.

13 Key and IV Scheduling Parts in T-310

13.1 Basic Facts About T-310 Keys

According to [45] the long-term keys LZS of T-310 take a form of plug-in cards and are changed only "when necessary", for example once per year. Daily keys "ZS" are implemented as punched cards and are changed weekly prior to 1982, then daily.

13.2 Key Scheduling and $s_{m,1-2}$

The key scheduling in T-310 is simply a periodic repetition every 120 rounds and following [76] we have:

$$s_{m+120,1-2} = s_{m,1-2}$$

The initial key is $s_{1-120,1-2}$ which is 240 bits, however 10 out of 240 bits are parity bits so the effective key size is 230 bits [76].

13.3 IV Generation and Transmission in T-310

According to [76] the IV is chosen at random in T-310 operation. It is then transmitted in cleartext in a form of a certain special sequence of characters called SYF (synchronization sequence) which has 25 characters, it is prefixed to the cipher message, and it is automatically recognized at the other end as a beginning of a transmission, cf. pages 15-17 in [80].

13.4 IV Expansion and f_m

The f_m sequence is obtained with an LFSR and it starts at $f_{-60}, \ldots, f_0$ which is the 61-bit IV which according to [76] is chosen at random. These bits are not used in encryption and the first bit used is f_1. The LFSR is defined by:

$$f_i = f_{i-61} \oplus f_{i-60} \oplus f_{i-59} \oplus f_{i-56}$$

Which corresponds to the polynomial $x^{61} = x^5 + x^2 + x^1 + 1$, cf. [46]. The period of this LFSR is $2^{61} - 1$ which is a prime.

14 T-310 Keystream Generation Process

T-310 is a cipher in which hundreds of rounds of a relatively complex block cipher are used to produce just a handful of bits of keystream. This keystream is produced and used in several stages: first some [extremely few] bits of the state $u_{i,j}$ are extracted and become bits of intermediate state a_i, which are further decimated a proportio of 10/13 of these bits will be used in actual encryption which we study later in Section 16.

14.1 Bit Selection For Encryption

T-310 has another part of long-term key called α which is simply a constant integer called $\alpha \in \{1, \ldots, 36\}$ here and in [76] (and called d in [44]) which governs the extraction of one bit every 127 rounds:

$$a_i \stackrel{def}{=} u_{127 \cdot i, \alpha}, \quad i = 1, 2, 3, \ldots$$

The for each $127 \cdot 13$ consecutive rounds we discard 3 bits out of 13 and we use 10 for encryption in a way specified in the next Section 16.

It is important to note that NOT every value α is permitted, some 6 values are excluded, $\alpha \notin W$ where $W = \{5, 9, 21, 25, 29, 33\}$, cf. Section B and D and page 117 in [80].

Remark. A basic observation is that a relatively large proportion of 10/13 of these bits will be actually used and conversely these bits are those which the attacker may have hopes to have some access to.

14.2 Discussion - Low-Rate Extraction

This selection of extremely few bits is rather (at least at first sight) where T-310 appears to be a **particularly strong**[26] cipher design. It seems that it is actually potentially stronger[27] than other ciphers we have compared it to, such as RC2, DES, or Skipjack, this is also what was intended by the designers in 1973 and what the BSI report from 1990 said, cf. Section 1.1.

The main point is that only one bit of the state of the cipher per 127 rounds of the block cipher is extracted for the actual encryption and could eventually be available to the attacker. This is an incredibly low quantity and cryptanalytic literature knows extremely few examples where the cipher would actually be broken under such difficult circumstances.

One major example is the so called "Dark Side Attack" on MiFare classic [29, 54], one of the most widely used security device on our planet, with approximately 2 billion RFID smart cards sold. In this attack the attacker obtains only 4 bits from each encryption [29, 54]. Here we can obtain only 1 bit for each 127 rounds of encryption, and though there is no limit on how many round we could have, the more rounds, the harder it becomes to develop any sort of cryptographic attack.

[26] Of course it could also easily be made yet a lot stronger, for example if a one-way function was used to format the outputs, or if we used a large size stateful filter/combiner such as on Fig. 2. in [24].

[27] Stronger, unless these ciphers would also be used in some specific "very careful" mode, with very few bits used for actual encryption, such as in T-310, cf. also [60].

15 Estimating Strength of T-310 Against Direct Software Algebraic Attacks

A natural question is how robust is T-310 against software algebraic attacks, techniques which as already explained do in a certain sense break any cipher, if not too complex, cf. [9, 30, 31, 18, 16, 34, 70]. Here the security of T-310 can be compared to KeeLoq, also a block cipher which locally looks like a stream cipher, and which has hundreds of rounds. General-purpose software key recovery attacks on KeeLoq with a SAT solver can recover the key for about 160 rounds only, cf. [25–27] for attacks running within hours/days on a PC, and having access to 32 bit of information per encryption. This would maybe scale up to 200 rounds for 1 CPU year. The complexity of KeeLoq is lower than T-310: in KeeLoq we have 1 Boolean function with 5 inputs per round, in T-310 we have 4 evaluations of a Boolean function with 6 inputs per round. In this respect T-310 remains more robust than KeeLoq[28] and is maybe comparable to Simon [18, 32, 16] which is a cipher of remarkable simplicity and extremely low multiplicative complexity [19]. Moreover here we dispose only of 1 bit of information per 127 rounds. Overall we do not expect that a SAT solver can break more than say 127 rounds of T-310 block cipher.

How Many Rounds do We Need to Attack? Initially it seems that the attacker has little choice other than work on the first character of the ciphertext C_0 and try to develop an attack on $11 \cdot 127 = 1397$ rounds. In this paper we are going to show that there exists non-trivial methods which allow the attacker to generate Plaintext/Ciphertext (P/C) pairs for less rounds, in some scenarios as little as 120 rounds only[29].

15.1 Computer Simulations

Our student Om Bhallamudi has developed an open source software solution for implementing software algebraic attacks on T-310. This software is described in Appendix K.

[28] KeeLoq is really a bad example, a particularly weak cipher which can broken with time complexity as low as 2^{28} [25] and even 2^{23} for 15 % of keys, cf. [26].

[29] In some cases yet fewer rounds, cf. attacks with faulty LZS in Section 20 where however the attacks such as described in this section would not be appropriate. An example of an attack scenario with extremely few rounds and also with faulty LZS where the software algebraic attack technique of this section would be applicable can be found in Appendix C.4.

16 Encryption in T-310 - Double One-Time Pad

As already explained, from the iterated block cipher we extract just 1 bit per 127 rounds: $u_{127,\alpha}, u_{254,\alpha}, u_{3\cdot 127}, \ldots, u_{1651,\alpha}$ and for every 13 bits we discard 3 and use 5+5 bits. More precisely we put:

$$C_j = (P_j \oplus B_j) \cdot M^{r_j}$$

where P_j/C_j is the plaintest/ciphertext character on 5 bits, respectively, then $B_j = (a_{7+13(j-1)}, \ldots, a_{11+13(j-1)})$ are 5 consecutive bits out of 13 previously discussed and r_j is a "stepping" output which is derived from the FIRST consecutive 5 bits out of 13 as follows:

$$r_j = \begin{cases} 0 & \text{if} \quad R_j = (0,0,0,0,0) \\ 0 & \text{if} \quad R_j = (1,1,1,1,1) \\ 32 - r & \text{if } R_j \cdot M^r = (1,1,1,1,1) \end{cases}$$

where $R_j \stackrel{def}{=} (a_{1+13(j-1)}, \ldots, a_{5+13(j-1)})$ and

$$M = \begin{pmatrix} 0\,0\,0\,0\,1 \\ 1\,0\,0\,0\,0 \\ 0\,1\,0\,0\,1 \\ 0\,0\,1\,0\,0 \\ 0\,0\,0\,1\,0 \end{pmatrix} \quad \text{which is such that} \quad M^{31} = Id.$$

17 Basic Observations and Basic Attacks on T-310 Encryption Process

17.1 Side Channel Attacks on T-310

In the formula $r_j = R_j \cdot M^r$ we see that T-310 implements essentially an LFSR with variable number of steps. This will be a serious vulnerability IF implemented incorrectly. If the timing of this operation is **NOT constant**, this will leak to the attacker one bit of information on the state **after ONLY**[30] **127 rounds** of encryption, and probably even less, because of the poor diffusion, most bits are created earlier than after 127 rounds. Then we can recover the key by some form[31] of a simple automated software algebraic attack [9, 30, 18, 16, 70]. However of course in historical teletype systems the timing was probably constant, so this remains a theoretical[32] attack.

Remark 1. There exists nowadays formal software methods for automatically synthesizing small size implementations for arbitrary small-size problems such as here, see [19, 82]. Such methods are used on both sides, for defensive [constant-time] optimizations, which would be needed here, and for improving/enhancing cryptanalytic attacks such as proposed above and elsewhere in this paper. Therefore evaluation the actual complexity of such an attack takes some serious work on the S-box representation side, see for example [9, 20, 19, 34]. Exact complexity of such attacks will be studied in future updates of this paper. Our first software solver was developed by our students cf. Section 15.1 and until now it provides all the basic functionality of a software algebraic attack except (not as of yet) these advanced optimizations in the line of [19, 71, 82].

Remark 2. Using extremely few bits of a state of an iterated block cipher in a protocol is a good practice in security engineering. It can also be used as a strong defence against other side-channel attacks such as DPA, and it is used a lot in the industry and subject to patents, see in particular [60].

[30] An attack on more than 127 rounds would be difficult, cf. Section 15

[31] We expect that a SAT solver attack will be suitable, and also an ElimLin-style attack [18, 79, 16] or/and also a correlation attack, cf. Section 15.

[32] It could be different if T-310 was re-implemented and used over more modern packet-switched networks.

17.2 The Zero Value Attacks on T-310

The Zero-Value attack is a well-known folklore[33] attack in side channel cryptanalysis [42, 55]. The key vulnerability is nicely summarized in the PhD thesis by Matthieu Rivain [74] where we read that "multiplicative masking has a serious drawback: it does not mask the zero value". We have exactly the same problem here with $\cdot M^r$ masking in T-310! We recall the encryption formula from Section 16:

$$C_j = (P_j \oplus B_j) \cdot M^{r_j}$$

Theorem 17.2.1 (Zero-Value Vulnerability in T-310 block cipher). If $C_j = 0^5$ on 5 bits, then $P_j = B_j$ regardless what the R_j/r_j values are.
The converse also holds: if $P_j = B_j$ on 5 bits, then we must have $C_j = 0$.

Note. This property shows that "double" one-time pad of T-310 has a security flaw, and shows it could become equivalent of a "single" one-time pad, if we restrict our attention to a subset of encrypted characters.

Bad News. Unhappily, the designers of T-310 have done well to make this sort of attack relatively unattractive: following Section 16 the first bit of B_j comes from a_7 which comes from round $7 \cdot 127 = 889$. However it is possible to see that if you invert the roles of r_j and B_j, the attacker would get access to bits at round 127. Breaking T-310 with bit output after 889 rounds seems ambitious.

Cube Attack? However, possibly, 889 relatively simple rounds would not be out of reach for cube attacks [83, 43]. Cube attacks are also perfectly suitable when the attack can access only one bit of a state inside the block cipher. Unhappily, here the attacker does not have access to encryptions with different plaintexts u_0 that he could control. Only the IV can be variable. So we could consider a cube attack where the IV bits are considered as plaintext, and u_0 is fixed. However even in this case, the attacker still cannot apply the attack: it is difficult to imagine that the attacker will dispose of encryptions with several expanded IVs 889 bits long each, such that they would form a cube.

ElimLin+ Attack. An attack which will be more suitable for T-310 will be an ElimLin+ attack, which is an ElimLin attack where the attacker generates additional equations which can be generated by interpolation for example when the plaintext+IV are fixed to some value [79]. Here the attacker will fix the plaintext to u_0 and generate equations for several IVs observed in the wild. What is expected that for every Nr there will exist a number K such that there exist linear equations which relate bits after Nr rounds from K different IV encryptions and for EVERY key. This will also work if we are allowed to use ONLY one bit per encryption, for example when $Nr = 127$ we would use $a_7 = u_{889,\alpha}$ only for many different IVs. This in the light of Zero-Value attack above we get an attack on T-310 for which it is easy to mathematically prove it will work using the ANF of a_7 seen as a Boolean function of the IV bits and key bits, cf. [78]. The "only" problem again, is that 889 is a large number.

[33] It is typically attributed to Golic and Tymen [55, 74, 68] however the attack was known to ourselves and other researchers before, and while Golic and Tymen have developed one specific solution to this problem [55], other very different solutions exist cf. [42, 68].

18 Preliminary Analysis for Correlation Attacks and Space Shrinking Properties

In this section we establish a number of basic facts useful for our later correlation attacks on T-310.

18.1 Useful Natural Language Statistics

In this paper we need some basic facts about the bias on individual bits for German language plaintexts encoded with Baudot code or ITA-2 which is used by T-310. For example we look at bit known as bit I in Baudot code, and ask the question what is the probability that this bit is equal to 0 for a long plaintext. In the table below we report these basic stats based on simulations on 750 Mbytes of German language corpus downloaded from the online archives of the Zeit magazine from 1980-2000, cf. `www.zeit.de`.

Table 3. Statistics for the bias on different bits which occur for German language with 5-bit Baudot code (upper table) and 8-bit Ascii coding (lower table).

$P(\text{bit I} = 0)$	$P(\text{bit II} = 0)$	$P(\text{bit III} = 0)$	$P(\text{bit IV} = 0)$	$P(\text{bit V} = 0)$
$1/2 + 2^{-2.89}$	$1/2 - 2^{-4.11}$	$1/2 - 2^{-3.23}$	$1/2 + 2^{-4.44}$	$1/2 + 2^{-3.74}$

$P(\text{b0} = 0)$	$P(\text{b1} = 0)$	$P(\text{b2} = 0)$	$P(\text{b3} = 0)$	$P(\text{b4} = 0)$	$P(\text{b5} = 0)$	$P(\text{b6} = 0)$	$P(\text{b7} = 0)$
$1/2 - 2^{-4.46}$	$1/2 + 2^{-3.59}$	$1/2 - 2^{-3.50}$	$1/2 + 2^{-2.83}$	$1/2 + 2^{-2.33}$	$1/2 - 2^{-1.19}$	$1/2 - 2^{-1.07}$	$1/2 + 2^{-1.06}$

These statistics are done for letters and numbers, with spaces and special characters removed. In addition in the Baudot case only, we have converted all letter to lowercase, and we have converted the "umluat" accented characters to plain equivalents, e.g. German ü becomes u. These statistics could be different in a real-life attack setting due to special rules used by T-310 operators.

In the ASCII case we keep the capital letters, and we have ignored special characters such as 0xC3 and only looked at statistics for the actual characters, for example ü can be encoded as 0xC3 0xBC and in this case we kept only the last character. An interesting remark is that for most bits and for German language, ASCII coding produces larger biases. However in this paper we need to use the Baudot ITA-2 code which is the one originally used with T-310.

18.2 Correlation Attack vs. Weak Keys in T-310

The question of whether an LZS will make ϕ bijective in T-310 is one of the central questions in this paper cf. Section 11.1 and Appendix C.6. These questions are closely related to the question of correlation attacks on T-310. There exists potentially many different correlation attacks in symmetric cryptanalysis. In this paper we study two sorts of such attacks. In this Section we study one type of 1-bit biases at the possibility of one single bit of the current state U_{alpha} being biased, aiming at a certain type of ciphertext-only attacks which are later studied in Section 20. In Section 21 and in Appendix G we will study some very different sorts of 2-bit correlations between two bits U_{alpha} inside the T-310 block cipher.

18.3 A Specific Reason Why Correlations Exist

In this paper we are going to show that 1-bit biases are bound to happen for one specific reason: when the space shrinks when ϕ is not bijective. Moreover, the bias which we are going to obtain can be predicted, and depends essentially on the entropy of the output distribution for ϕ imposed by the long-term key LZS. In the following pages we will show that for example, if we consider some keys such as defined in Appendix E.2, the bias will be weaker, and we generate keys which obey to a substantially smaller subset of the conditions, cf. later Section 18.8, the bias will be yet stronger, and moreover it appears that the bias will be at least as good as expected from the shrinking properties, sometimes better, and we will see that bias will happen for **any** α for all the weak keys we consider.

18.4 A Method For Fast Estimation of Output Space

In this paper we will use a fast and inexact method for estimation of the output space size which is based on birthday paradox [84] and which is closely related to the notions of entropy and collision entropy. It is a well known result that collision entropy is at most equal to the Shannon's entropy and it cannot be too small, cf. Table 1 page 3 in [77] for a precise result. The key question is how to measure the entropy of the output distribution of ϕ **in approximation** without doing 2^{36} encryptions. This is needed in order to be able to quickly evaluate the comparative strength of different long-term keys LZS against correlation attacks which will study later.

We are facing the problem of efficiently approximating the entropy from observation using an oracle access, which problem is studied in detail in [1]. In this paper we need a fast method for approximating the result which will allow us to check many different long-term keys in a short time. In order to simplify the problem, we can for example assume that we sample the output space more or less uniformly with some M frequent values obtained by ϕ, and that all the other values occur less frequently and we neglect their existence. This question of estimating the collision probability from $\Omega(\sqrt{M})$ samples is mentioned on page 1 in [1] and the basic idea is that estimating Shannon's entropy is possible from

the the collision probability each time the Min-Entropy is large, i.e. we do not have any [34] events occurring with probability substantially higher than $1/M$.

In this paper we apply the birthday paradox to estimate the size of M from observation of the collision probability exactly. More precisely, we make an important simplifying assumption that if the entropy is equal to $\log_2(M)$, we assume that our output distribution behaves as a uniform distribution for M events, and that other events other than the M most frequent events, do not happen very frequently. Then we can draw the outputs at random in a way similar as in Thm. 21 in [1], until a collision occurs. We stop at 1 collision, and we measure the average expected time $\overline{n}$ for a first collision to occur. Then, under our simplifications assuming that M events are nearly equi-probable, and following [84] the average expected time $\overline{n}$ for a first collision to occur for a population of M is governed by the following approximation where we neglect negligible quantities in $1/M$ or smaller, and which is due to Ramanujan:

$$\overline{n} \approx 1 + \sqrt{\frac{\pi M}{2}} - \frac{1}{3}$$

Accordingly we can obtain $\overline{n}$ by running a few hundred simulations stop at the first collision and restart, and we can then estimate the size of output space M quite precisely as follows:

$$M \approx \frac{2(3\overline{n} - 2)^2}{9\pi}$$

This method will be used below for different LZS.

[34] This is what we expect here or we would have a different sort of attack on T-310 with guessing the full state on 36 bits

18.5 Space Shrinking - Original Keys vs. Special Keys

The main idea in our later correlation attacks is that correlations are going to occur because the output space shrinks for many (weaker) long-term keys. In the pages which will follow, we study how much exactly the image of type $\phi^k(\{1,\ldots,36\})$ shrinks for certain weak long-term keys (weak LZS). We start by looking at some original long-term keys found in [44]. We compare it to a special key we 208 have generated in Appendix E.2 as a counter-example in our proof that KT2 are bijective, it satisfies **all** the conditions of KT2 of Appendix D.1 except the "Matrix" which has $rank = 8$ instead of 9, cf. Appendix D.4.

Table 4. Space shrinking properties: comparison of a regular KT2 key 15 with bijective ϕ, some anomalous keys from [44] and our "Rank Deficient" key 208.

key nb	D	P	$\lvert\phi_0(\{0,1\}^{36})\rvert$	$\lvert\phi_0^4(\{0,1\}^{36})\rvert$	$\lvert\phi_0^{16}(\{0,1\}^{36})\rvert$	$\lvert\phi_0^{24}(\{0,1\}^{36})\rvert$
15	0,4,17,12,35,32,2,24,20	15,13,33,34,6,8,5,3,9,18, 14,22,28,30,21,31,7,25,26, 16,27,11,23,29,19,1,36	$2^{36.0}$	$2^{36.0}$	$2^{36.0}$	$2^{36.0}$
29	0,36,28,20,24,16,4,12,8	28,8,33,23,11,12,5,10,9,30, 19,18,4,31,21,24,13,25,22, 32,20,36,27,29,7,16,15	$2^{36.0}$	$2^{36.0}$	$2^{36.0}$	$2^{36.0}$
16	0,35,19,23,27,11,3,15,31	14,19,33,18,23,15,5,6,9,2, 34,1,30,11,21,3,22,25,17, 7,32,10,27,29,26,35,13	$2^{27.4}$	$2^{27.0}$	$2^{27.0}$	$2^{27.0}$
17	0,4,8,12,16,20,24,28,32	22,23,33,11,26,12,5,4,9,3, 2,1,19,10,21,8,7,25,6,35, 32,31,30,29,17,17,34	$2^{35.1}$	$2^{32.6}$	$2^{31.2}$	$2^{30.3}$
27	8,3,5,2,4,6,7,9,1	10,21,18,4,5,8,16,12,6,24, 2,7,3,25,17,26,9,14,22,1, 20,11,19,15,13,23,27	$2^{30.3}$	$2^{19.9}$	$2^{16.1}$	$2^{15.2}$
208	17,0,2,32,35,4,12,20,24	13,15,33,10,18,8,5,30,9,6, 3,14,16,22,21,31,7,25,26, 28,27,11,23,29,19,1,36	$2^{34.8}$	$2^{33.6}$	$2^{32.1}$	$2^{31.6}$

On Notation ϕ_0. The meaning of ϕ_0 is this table is that all the 3 key/IV bits in each round are fixed to 0 in each round. More precisely we recall from Section C.5 that each ϕ depends on has 3 key/IV bits s_1, s_2, f which makes that T-310 operates with non-commutative combinations of exactly 8 fixed permutations on 36 bits which are called $\phi_0, \ldots, \phi_7$ in Section 1.5 in [80]. For example we can have $\phi_3 \circ \phi_2 \circ \phi_7 \circ \phi_4$ with four rounds. The document also calls $G(P, D)$ the group generated by these 8 permutations and contains a number of results about composition of these permutations. The question of how much the shrinking results depend of which ϕ_s we will compose with each other is studied below.

18.6 Shrinking vs. Choice of Key and IV Bits - Key 208

For example potentially if we just compose ϕ_0^K for some K with all bits at 0. It is then important to see that the case ϕ_0^K will not[35] occur in a real life attack. Nevertheless we can study how the space shrinks also in this[36] case.

In Table 5 below we present some results for ϕ_0^K space size and key 208 as defined in Table 10 and using the fast method of Section 18.4. On the left in Table 5 we have all bits at 0, and on the right we look at random sequences

[35] If it does occur we would probably be able to exploit its periodic structure in various slide or self-similarity attacks [25, 34].

[36] This case does **not** differ from the general case in practice as we will see later

of type $\phi_3 \circ \phi_2 \circ \phi_7 \circ \phi_4$ or similar, which is more realistic compared to how a real attack would operate. In fact no difference of practical importance was ever observed. The results are very similar for every key and for every IV.

Table 5. Output space size for key 208 with all key/IV bits at 0 with ϕ_0^K (left) and for ϕ_s^K (tight) with a randomly chosen sequence $s \in \{0-7\}^K$

key nb	K	output space
208	0	$2^{36.0}$
208	1	$2^{35.0}$
208	2	$2^{34.4}$
208	4	$2^{33.6}$
208	8	$2^{33.0}$
208	16	$2^{32.1}$
208	32	$2^{31.2}$

key nb	K	output space
208	0	$2^{36.0}$
208	1	$2^{35.0}$
208	2	$2^{34.5}$
208	4	$2^{33.8}$
208	8	$2^{33.0}$
208	16	$2^{32.2}$
208	32	$2^{31.3}$

Overall we see that for key 208, the shrinking property is not very strong, and it is not true that we can shrink the space more substantially by increasing the number of iterations.

18.7 Weaker Rank-Deficient Keys in KT2b Style

An interesting question is whether we can generate some keys weaker than 208. In this sub-section we present one method to to this, which is not yet very good, another method will be studied in Section 18.8 below. For example we can try to generated weaker keys starting from KT2b conditions which are already potentially SUFFICIENT for T-310 to be totally secure, cf. Thm. D.6.1, except that we allow the rank to be deficient and lower than 9.

Table 6. New key 308 based on class KT2b except for matrix rank condition M_9.

key nb	D	P	rank of B
308	0,16,2,8,24,20,11,32,4	6,35,33,17,26,13,5,27,9,10,19,18,12,30,21,15,34,25,23,36,31,14,22,29,3,1,28	8
15	0,4,17,12,35,32,2,24,20	15,13,33,34,6,8,5,3,9,18,14,22,28,30,21,31,7,25,26,16,27,11,23,29,19,1,36	9

We can now compare how the space shrinks with key 308 compared to 208:

Table 7. Space shrinking comparison of keys 208 and 308 with ϕ_s^K, random s.

key nb	K	output space
208	0	$2^{36.0}$
208	4	$2^{33.8}$
208	16	$2^{32.2}$
208	64	$2^{30.2}$

key nb	K	output space
308	0	$2^{36.0}$
308	4	$2^{33.7}$
308	16	$2^{31.9}$
308	64	$2^{29.8}$

We see that key 308 is only slightly weaker than key 208. The idea that we need to give up on to KT2 and used a greatly reduced set of conditions KT2b in

order to generate weak keys for T-310 does not seem to work well. Alternatively we need to remove even more conditions, cf. Section 18.8 below. More precisely, in the following sub-sections we will see that we can find substantially weaker keys than 308 also if we follow absolutely all of some 40 rules mandated for KT2 keys, except (again) for the rank condition, cf. Section 18.9 below. We will also see that if we really want to produce the weakest possible keys we should rather try indeed random keys which satisfy a really minimal set of conditions KT3d, cf. Section 18.8 below.

18.8 Class KT3d - More Weak LZS Keys Generated At Random

An interesting question is: if we generate D, P at random with a really minimal number of conditions, such as and we are still avoiding any sort of "anomalous" situation such as such as key 17 which has $P(25) = P(26)$, cf. [44], how secure this would be? For this we are going to define our own class of keys called KT3d with a set of conditions which we consider a strict minimum, we define:

$$(P, D) \in KT3d \Leftrightarrow \text{ all the following hold:}$$

$$\begin{cases} D \text{ and } P \text{ are injective} \\ \forall (i,j) \in \{1,\ldots,27\} \times \{1,\ldots,9\} : P_i \neq D_j \\ \exists j_1 \in \{1,\ldots,7\} : D_{j_1} = 0 \end{cases}$$

Remark. KT2 and KT2b are included in class KT3d, but KT1 are not, because they have repeated entries of type $P(13) = D(7)$.

Now the question is how secure are these keys w.r.t. to the space shrinking properties such as in Table 5. In Table 8 we provide several examples of keys of type KT3d. For comparison purposes we also include key 16 of [44] which according to [80] is a special key which emulates a permutation on 27 bits of the so called SKS cipher. Finally, we include several previously studied "Rank-Deficient" keys in KT2 or KT2b style, and regular key 15 of [44] which is of type KT2.

Table 8. Examples of keys of type KT3d and their space shrinking properties

key nb	D	P	$\lvert\phi_0(\{0,1\}^{36})\rvert$	$\lvert\phi_0^4(\{0,1\}^{36})\rvert$	$\lvert\phi_0^{16}(\{0,1\}^{36})\rvert$	$\lvert\phi_0^{24}(\{0,1\}^{36})\rvert$
934	0,4,20,12,14,9,19,7,10	21,3,16,25,28,30,26,11,1, 5,6,32,36,29,24,2,23,33, 27,34,8,18,17,31,35,13,22	$2^{32.8}$	$2^{27.7}$	$2^{24.3}$	$2^{23.5}$
930	18,19,0,23,21,10,25,20	33,34,28,31,32,35,6,24,9, 16,15,30,29,3,14,26,11, 27,5,2,4,8,36,22,7,12,17	$2^{30.3}$	$2^{23.6}$	$2^{21.0}$	$2^{20.0}$
914	30,33,14,21,31,0,36,3,35	26,29,27,23,17,10,20,6,7, 16,32,25,2,22,15,9,11,24, 1,18,5,13,8,12,4,28,34	$2^{30.4}$	$2^{22.3}$	$2^{20.0}$	$2^{19.2}$
913	9,11,34,0,2,3,26,7,33	5,17,28,32,29,30,13,25,10, 23,36,31,21,14,15,22,18, 27,35,12,16,20,6,19,8,4,1	$2^{29.7}$	$2^{22.8}$	$2^{18.8}$	$2^{17.9}$
912	11,34,2,0,9,26,3,7,33	31,17,28,25,29,30,13,5,10, 24,14,23,36,21,15,22,18, 27,35,12,16,20,6,19,8,4,1	$2^{31.2}$	$2^{24.0}$	$2^{18.2}$	$2^{17.2}$
911	34,11,2,9,0,26,3,7,33	25,17,28,32,29,30,13,5,10, 23,14,24,21,36,15,22,18, 27,35,12,16,20,6,19,8,4,1	$2^{30.3}$	$2^{23.1}$	$2^{18.4}$	$2^{17.3}$
206	4,0,32,2,35,17,12,20,24	15,13,33,18,34,8,5,6,9,30, 22,14,16,3,21,31,7,25,26, 28,27,11,23,29,19,1,36	$2^{35.1}$	$2^{34.0}$	$2^{32.9}$	$2^{32.3}$
407	0,24,20,8,16,2,11,32,4	17,7,33,6,10,13,5,27,9,26, 22,18,12,30,21,15,34,25,23, 36,31,14,19,29,3,1,28	$2^{34.0}$	$2^{32.4}$	$2^{30.1}$	$2^{29.5}$
207	0,24,20,8,16,2,11,32,4	7,6,33,26,17,13,5,19,9,10, 27,18,12,30,21,15,34,25,23, 36,31,14,22,29,3,1,28	$2^{34.0}$	$2^{32.2}$	$2^{30.3}$	$2^{29.3}$
15	0,4,17,12,35,32,2,24,20	15,13,33,34,6,8,5,3,9,18, 14,22,28,30,21,31,7,25,26, 16,27,11,23,29,19,1,36	$2^{36.0}$	$2^{36.0}$	$2^{36.0}$	$2^{36.0}$

Observations. We see that the vulnerability of keys in class KT3d against space shrinking attacks varies very substantially for different keys in KT3d. The image space size of a typical KT3d key is less than 2^{35} and by trial and error, we have NOT been able to generate a single key of type KT3d with image size of 2^{35}, even though we know that such keys exist, for example 208 except that these subclasses do NOT occur at random with a sufficiently large probability.

18.9 How Output Space Reduction Produces Bias

In this paper we apply the following heuristic:

Conjecture 18.9.1 (Bias As A Result of Output Space Reduction). If for every sequence s of IV bits and key bits, ϕ_s^k does reduces the size of the output space to M frequent elements of $\mathbb{F}_2^{36}$, the we expect that for very α the output U_α will be biased with the same probability distribution as for a choice of M random elements of $\mathbb{F}_2^{36}$.

Justification: This is unlikely to be true in general, for example if ϕ is a mapping $\mathbb{F}_2^{36} \to \mathbb{F}_2^{36}$ which copies 25 bits and fixes the last bit to 0, for all $\alpha \neq 36$, and we have a very strong bias for the last bit. However we expect that this should be true in practice, and this will be the basis to estimate the bias as a function of M which we expect to be approximately $\mathcal{O}(\sqrt{1/M})$.

Below we present some experimental results for one key and IV sequence chosen at random for several weak long-term keys we have generated. This sign of the bias changes for another pair of IV,key, and the number of secret key bits used is limited to $2k$ for ϕ^k therefore it is realistic to expect that the attacker can guess these bits. In this table we average the bias for several different keys,

while keeping the same fixed IV. The keys used here defined in Table 8 and in Table 10.

Table 9. Simulations for ϕ^{16} which shows the average bias in absolute value for the U_α bit of cipher state after 16 rounds, for one well-chosen α_{best}, and averaged for any $\alpha \in \{1-36\}$ and over many keys.

LZS	$M_{\phi^{16}}$	α_{best}	$\lvert P(U_{\alpha_{best}}=0)-1/2\rvert$	average(1-36,keys)
206	$2^{32.9}$	16	$2^{-14.6}$	$2^{-15.5}$
208	$2^{32.2}$		2^{-}	$2^{-15.6}$
407	$2^{30.5}$		2^{-}	$2^{-15.2}$
207	$2^{30.2}$		2^{-}	$2^{-15.3}$
16	$2^{27.0}$	15	$2^{-13.1}$	$2^{-14.0}$
17	$2^{31.2}$		2^{-}	2^{-}
27	$2^{16.1}$	1	$2^{-9.0}$	$2^{-8.0}$
28	$2^{18.7}$		2^{-}	$2^{-10.8}$
934	$2^{24.3}$	23	$2^{-10.5}$	$2^{-12.9}$
930	$2^{21.0}$		2^{-}	2^{-}
914	$2^{20.0}$	21	$2^{-10.3}$	$2^{-11.7}$
913	$2^{18.8}$	1	$2^{-11.3}$	$2^{-10.9}$
912	$2^{18.2}$	1	$2^{-10.6}$	$2^{-10.7}$
911	$2^{18.4}$	29	$2^{-9.8}$	$2^{-11.0}$

We observed that the bias is quite substantial for **any** value of α and for any weak key studied, and it seems to follow a simple law $\mathcal{O}(\sqrt{1/M})$ which is what we would expect for a random function with M possible outputs.

19 On Chosen LZS Attacks

In this section we look at the question what kind of attacks are possible given all the properties studied in the previous section. An actual attack will be described in the next section.

19.1 A Problematic LZS Question

The key question is as follows: the function ϕ in T-310 is meant to be bijective. This question was not considered in [76] because this property is NOT required in normal operation of the cipher, see Section 11.1. Yet it is more or less clearly stated inside page 56 [80]. Now there are two crucial questions:

1. Is there a plausible scenario for a real-life attack where the LZS would not be bijective? Could it for example be quite difficult or cumbersome for German security services employees to detect that some long-term key is faulty, and therefore it could be used for some time without anyone noticing?
2. What are the consequences of an LZS being non-bijective? Is there a really fast attack significantly faster than 2^{230}?
3. Is there an attack faster than say 2^{50} feasible to execute in practice on a PC?
4. Is the attack scenario realistic: or for example all that we would get would be some sort of attack with repeated IV such as one previously outlined in Section C.4? Therefore just another[37] attack with repeated IVs would not be a game changer.
5. Could we have something like a ciphertext only attack? This is a rare thing in cryptanalysis research, cf. [53, 69].

In what follows we are going to see that the answer is yes to all these questions.

19.2 On Rank Deficiency of Some Otherwise Well-Formed Keys

So of the keys such as 207 we have studied above have some interesting properties. We define a "Rank-Deficient" KT2 long-term key as follows:

Definition 19.2.1 (Rank-Deficient KT2 key).
It will be any key which satisfies all of the some 40 technical conditions specified in pages 59-60,114-115 and 117 in [80] and in Section D except the very last rank condition about the matrix B of page 60.

Remark. If a key is "Rank-Deficient KT2" it is likely that this would be unnoticed. The condition which the KT2 keys must satisfy are numerous and very tedious to check. Some of them at random could be checked by a person which is charge with approval of such key, and the key would be the approved trusting that the employee which has generated it has done a good job. The original documents do NOT mention if KT2 have been proven to be bijective

[37] For sure the designers of T-310 knew about such attacks cf. [80, 46] have re-engineered the process to avoid them, and would not agree that this is a realistic attack. Moreover we already have developed several attacks with repeated IVs, for example in Section 21 and another in Appendix G.

[80]. The employee could have some doubt whether it is useful at all to perform any check on the keys and what properties are really required and the cipher would not be secure with.

Moreover out of all the conditions, this last condition could be the one which employees could systematically omit to verify. The reason for that that the matrix is NOT fully specified in [80]. The statement is highly ambiguous and does not meet the standard of a routine check people should run frequently. This matrix statement is poorly written with a high degree of ambiguity, and a reader could initially be puzzled by this condition, rather than accept it and just check it. This is because it is NOT at all obvious that such a matrix should exist in the first place, which we show in Lemma D.4.1 page 74. Moreover this condition requires a computer simulation and just **cannot** [38] **be checked manually** by a sort of person which would have the skills and understanding to check the other conditions which are written in elementary maths language. Overall, we believe that this check is very likely to be disregarded[39] in a real-life situation.

[38] To check this condition require slightly different skills and set of mind. It could only be checked through a complex computation which is prone to errors if done by hand, where the object of the study is not described in a readable way, and which could only plausibly be done with a computer algebra software such as NTL/Maple/SAGE we have used. Yet most East-German security personnel in 1980s would not have access neither to any computers of any sort, nor to computer algebra software we take for granted today.

[39] We have ourselves skipped this check for a long time until we discovered keys with bad properties actually exist, cf. Section E.5.

20 A Ciphertext-Only Faulty LZS Correlation Attack

In this section we describe an interesting new attack on T-310. This attack has potentially a very low complexity and we believe that this a practical[40] attack which very significantly undermines a confidence in T-310 algorithm. It is a non-standard form of attack, not one which appears frequently in crypto literature. Yet is also an attack which is likely to have a significant impact on the real-life security of this government encryption system which will be shown potentially highly vulnerable[41]. It combines four major vulnerabilities of T-310 we have previously uncovered: the Zero-Value attack of Section 17.2, the plausibility of a weak key being used in the real life studied in Section 19, the correlations of Table 9, and the plaintext statistics of Table 3. In the light of these vulnerabilities, another property of T-310 comes to light as a serious vulnerability the importance of which has been heavily underestimated so far.

20.1 On Key Scheduling in T-310

More precisely T-310 has an extremely weak[42] key schedule, and it should not be used, because there is a significant risk of a serious attack. To be honest, for a long time we did not think that anything was really wrong with T-310 key scheduling. In the same way, nobody thought for more than 20 years that the highly-regular key scheduling in GOST could lead to any significant attacks and until 2010 there was simply no attack on GOST, which is clearly stated in [72]. Then attacks on GOST have literally exploded, cf. [34, 38] for pointers to abundant 200+ pages long recent research on this topic. Initially we thought nothing could go wrong with a perfectly periodic key schedule, because the strongly a-periodic character of the IV handing in T-310. We were wrong as we are going to see below. This is due to a new attack scenario which we have not anticipated. Moreover we are going to show that such an attack could pose a significant threat to on an encryption system used in the real life.

[40] It could lead to decryption of communications encrypted by T-310 in the real life and in the ciphertext-only scenario as we will show later.

[41] It does not really matter whether this attack could have happened or if it has actually happened. The fact alone that this sort of attack is possible at all shows that T-310 is not a good cipher. Even though very clearly, in theory T-310 has been designed to avoid also this type of attack, cf. two theorems about KT1/KT2 in Appendix C.10 and D.6, there is serious problem. The mathematical foundations which make the cipher resistant to this attack, do **not** make it resistant to it in practice. An enemy could exploit the excessive complexity of how LZS are specified, or play on their over-confidence about the security of their cipher machines, and try to convince people to use a faulty key and it will be hard to check if it is deficient.

[42] Such as many other ciphers which were badly broken in the past cf. for example [7, 34, 38, 25–27] and this would a certainly be a good reason for a cipher to be rejected as a candidate for an encryption standard, cf. [7, 34, 38].

20.2 A Ciphertext-Only Correlation Attack on T-310

In this section we show how to combine the biases of ϕ^k output in Table 9 and biases on the plaintext due to Table 3 and Thm. 17.2.1 in order to decrypt T-310 communications in the ciphertext-only scenario. Our correlation attack works as follows:

1. We apply the Zero-Value attack and we will exploit a proportion of 2^{-5} of available ciphertext data.
2. We recall from Section 17.2 that if $C_j = 0^5$ we have $P_j = B_j$.
3. We can now approximate **certain not all** bits of the plaintext by $P_{j,I-V} = B_{j,0-5}$ which holds for **all** ciphertext characters $C_j = 0$ we selected.
4. by approximating the c and approximate the 5 bits of B_j knowing that $B_{j,0-5} = (a_{7+13(j-1)}, \ldots, a_{11+13(j-1)})$ and all these bits are biased using Table 9.
5. We know the expected average value of the bias but we do not know the sign of the bias. The sign of the bias depends on the values of key and IV bits preceding any of the $(a_{7+13(j-1)}, \ldots, a_{11+13(j-1)})$ which by definition are equal to $u_{127(7+13(j-1)),\alpha}, \ldots, u_{127(11+13(j-1)),\alpha}$. We know the IV bits at any location, we just need to guess key bits at certain locations.
6. In our attack we are going to guess a window of say 48 keys bits for a window of 24 consecutive rounds. The same window of 48 bits is repeated every 120 rounds, (with different IVs which are known to the attacker).
7. We will work on individual bits, and if we want to be able to know the sign of a bias reported in Table 9, we need to know the 32 key bits preceding the actual bits extracted which are $u_{m,\alpha}$ with $m = 127(B + 13(j-1))$ with five possible $B = 7 - 11$.
8. We assume that the attacker disposes of a pre-computed table which indicates the sign $\sigma_{K,IV} = +1$ or -1 for the bias for any 32 bits key and any 16 bit IV for ϕ_s^{16}. This table requires only 1 Terabyte of storage (2^{48} bits).
9. The probability that any $B_{j,0-5}$ we want to compute, can be approximated as a biased bit of type say $1/2 - 2^{-5.8}$ with the sign know to the attacker, is equal to $(48-32)/120 \approx 2^{-2.9}$.
10. In order to simplify our attack, we will only work on plaintext bits I and III in Table 3 which both have a bias of approximately $\pm 2^{-3.0}$. We need to pay attention to the signs, let $\sigma_I = +1$ and $\sigma_{III} = -1$ for these two bits.
11. The attacker will now compute many biased bits which are all more likely to be 0 than 1, and which combine the biases due to the plaintext and due to ϕ^{16}. Then he will count 0s and 1s and if the bias is sufficiently large he will be able to confirm if his choice of 48 was correct.
12. The attacker assumes that $B_{j,0} = (1 + \sigma_{K,IV})/2$ which is true with probability of about $0.5 + \beta$ where β is a positive value from Table 9, for example for LZS-27 we have $\beta \approx 2^{-8}$. Similarly we have $B_{j,2} = (1 + \sigma_{K,IV})/2$ for a different choice of 32 key bits and 16 IV bits which pertain to this position.
13. We know that $B_{j,0} = P_{j,I}$ and $B_{j,2} = P_{j,III}$ for all ciphertext positions with $C_j = 0^5$ selected.
    The sequence of bits the attack produces will be simply all the $(1+\sigma_I\sigma_{K,IV})/2$ or $(1+\sigma_{III}\sigma_{K,IV})/2$ for all the cases considered. We call these bits available to the attacker "the $B - I$ set".

14. We apply the Matsui's piling-up lemma [64] and we see that the overall bias for our bits which are $(1 + \sigma_I \sigma_{K,IV})/2$ or $(1 + \sigma_{III} \sigma_{K,IV})/2$ is going to be equal to $\gamma = 2^{-3.0}\beta$.
15. In order to distinguish these biased distributions and have results which is stronger than 8 standard deviations we need to generate about $8^2\gamma^{-2} \approx 2^{16+6.0}\beta^{-2}$ of biased bits in "the $B - I$ set".
16. We need to work with 8 standard deviations exactly: we apply the Gauss Error function cf. [39] which leads to probability of $2^{-49.5}$ of a false positive which is sufficient to confirm if our 48-bit key is correct.
17. We get 2 bits for our "the $B-I$ set" when we have ciphertext character with $C_j = 0^5$ which happens with probability 2^{-5} AND when simultaneously the window of 32 bits needed is contained within our window of 48 bits which happens with probability $2^{-2.9}$.
18. Therefore we need overall $2^{16+6.0+2.9+5}\beta^{-2}/2 \approx 2^{31.9}\beta^{-2}$ of encrypted characters in order to recover 48 bits of the key in time which is approximately $2^{48+31.9-5-2.9}\beta^{-2} \approx 2^{72}\beta^{-2}$.

    Here $-5-2.9$ comes from the fact the we can pre-select ciphertext bytes for the attack independently of the key depending on the window position.
19. Once we have a plausible candidate for 48 key bits, we can re-do the whole attack with a different and preferably overlapping interval of 23 consecutive rounds and 48 key bits. Making these intervals overlap with those where key bits are already known makes that these extra steps will be substantially faster and easier and their cost can be neglected.

Example of Application - Key 27: With key 27, we have $\beta = 2^{-8.0}$ typically, and the attacker can recover the full 230-bit encryption key in time of 2^{88} given about 2^{48} characters of encrypted data in the ciphertext-only scenario.

Note: This attack can be further optimized to be more flexible. If we dispose of more data we can start with the number of key bits guessed smaller than 48 bits. If we dispose of more data, we need to guess more bits.

21 Decryption Oracle Attacks and Keystream Recovery

A plausible attack scenario is that the attacker would have access to a decryption oracle. The attacker can send any IV and the ciphertext and can recover the plaintext. In this case we are going to show (over the next few pages) that the cipher is not secure.

For example the attacker can send several messages with the same IV with:

$$C_0 = P_0 \cdot M^{r_0} \oplus B_0 \cdot M^{r_0}$$

Then in all these encryptions r_0 and B_0 will be the same. So for two encryptions with the same IV we have:

$$C_0 \oplus C'_0 = (P_0 \oplus P'_0) \cdot M^{r_0}$$

and more generally if ciphertexts submitted to the oracle have length k characters

$$C_j \oplus C'_j = (P_j \oplus P'_j) \cdot M^{r_j} \quad \text{for all } \ 0 \le j < k.$$

This allows to recover M^{r_j} uniquely in a proportion of 1-1/32 of cases where $C_j \neq C'_j$. In addition the attacker could chose ciphertexts such that $C_j \neq C'_j$. Moreover we recall following Section 16. that M^{r_j} does almost always [but not always] allow to determine R_j:

$$r_0 = \begin{cases} 0 & \text{if} \quad R_j = (0,0,0,0,0) \\ 0 & \text{if} \quad R_j = (1,1,1,1,1) \\ 32 - r & \text{if } R_0 \cdot M^r = (1,1,1,1,1) \end{cases}$$

Therefore we need to the proportion of 1/32 cases where r_j is 0 modulo 31, which cases create an ambiguity on the bits $R_0 = a_{1-5}$. One of the two problematic events happens with overall probability less than 2/32. Overall in at least 30/32 cases over all possible pairs $P/C, P'/C'$, where the 5 bits of $R_0 = a_{1-5}$ are uniquely determined and $r_j \neq 0$. We obtain the following overall result:

Theorem 21.0.1 (Decryption Oracle Attack on u_{127}). For every IV chosen by the attacker, given at most about $2 \cdot 32/30 \approx 2.13$ "Chosen IV and Random Ciphertext" (CIVRC) queries on average, the attacker can obtain $a_1 = u_{127,\alpha}$ with a negligible computation effort. We will assume that α is known or we guess it (it has low entropy and many choices are substantially weaker). Moreover, with "Chosen IV and Chosen Ciphertext" (CIVCC) queries, we only need about $2 \cdot 32/31 \approx 2.06$ CIVCC queries where the attacker can make sure that $C_0 \neq C'_0$ for any pair.

Proof: The result is straightforward and all these steps were already give above. In the CIVCC case the attacker can make sure that $C_0 \neq C'_0$ for any pair by selecting up to 32 chosen ciphertexts with different C_0. This cannot be done for more than 32 cases, however the probability that the attacker would ever need more than 32 calls to the decryption oracle (due to the fact that for each pair he would get $r_j = 0$ is extremely small, of the order of $(1/31)^{\binom{32}{2}} \approx 2^{-2457}$.

Key Recovery Step. Access to many values of $a_1 = u_{127,\alpha}$ for different IVs, should be sufficient to recover the T-310 key with a SAT solver [9] or ElimLin+ attack [79] in a similar way as for 160 of KeeLoq cf. [25, 26]. The exact time and data complexity of this attack will be studied in a future update of this paper.

21.1 General Black-Box Decryption Oracle Attack

We generalize the same attack for $a_1, a_2, a_3, \ldots, a_k$. It is possible to see that the data complexity grows quite slowly with k. We will now consider only the (stronger) CIVCC attack, as we find it hard to imagine a scenario where an attacker could do CIVRC and not CIVCC.

Theorem 21.1.1 (General Decryption Oracle Attack). For every IV chosen by the attacker, and for every $k \geq 1$, the attacker can obtain all of a_{1-k} with a computation effort linear in k and with about $K \approx \sqrt{(\log_2(k) + 10)/2}$ decryption CIVCC queries on average with one fixed chosen IV and random ciphertexts, and with ciphertexts length of k characters.

Proof: We assume that we have K decryption CIVCC oracle queries (with chosen IV and chosen ciphertexts) with ciphertexts of length k characters. This gives us about $K^2/2$ pairs where we could try to apply the formula:

$$C_j \oplus C'_j = (P_j \oplus P'_j) \cdot M^{r_j} \quad \text{for all } \ 0 \leq j < k.$$

and following the analysis in the previous Section 21 exactly and only two things can go wrong in each of these cases. Either we have $C_j = C'_j$ or $r_j = 0$. This again happens with probability less than $2/32$. The probability that we need at least K CIVCC is the probability that for every j all the $K^2/2$ pairs fail to work, i.e. less than $k \cdot (2/32)^{K^2/2}$. If we want for example a 0.1 % failure rate we need $k \cdot (2/32)^{K^2/2} \approx 2^{-10}$ which leads to $\log_2(k) = 4K^2/2 - 10$ and therefore we get $K \approx \sqrt{(\log_2(k) + 10)/2}$. This quite is small in practice, for example for all $k \leq 4$ million, we need not more than $K = 4$ decryption queries. and for $k \leq 2^{40}$ we would use $K = 5$.

22 A Decryption Oracle with a Slide Attack

In the previous section Thm. 21.0.1 we see that in the decryption oracle scenario, it is relatively easy to recover the keystream components R_j, B_j by asking for several decryptions with the same IV and we need about $K = 2$ sometimes $K = 3$ decryption queries per IV. This gives access to a_1 which however still depends on 230 key bits. In addition we get access to further $a_2, a_3, \ldots$ very cheaply: we don't expect that we will ever need to do more than $K = 4$ decryption queries per IV for up to 4 million, cf. Thm. 21.1.1. This is what we are going to exploit now. the starting point is that we have **not yet** used the full power of the chosen IV attack scenario: the capacity to select many arbitrary IVs and therefore for example well-chosen related IVs.

Now we are going to design our slide attack. There exist many different slide attacks, e.g. [56, 25, 34]. We want to exploit self-similarity of the T-310 block cipher: the key bits repeat every 120 rounds, and we need to adjust the IV bits in order to obtain identical permutation. Then the question will be whether these identical permutations can have identical inputs. Here is our first basic attack.

First we consider some integer s such that:

1) $120s$ is small mod 127
2) $120s$ is not too large in absolute value (or we will need to decrypt long messages)

For example $s = 18$ has $120{\cdot}18 = 127{\cdot}17{+}1$. Of $s = 1$ given $120{\cdot}1 = 127{\cdot}1{-}7$. The first example, as we will see later, is one we have designed for "stronger" long-term key settings of T-310. The second example is meant to work more easily for "weaker" long-term key settings of T-310.

Now the main idea in the attack is that **if** by some sort of "happy" accident for some encryption with some IV, we have

$$u_{120s+0} = u_0 = 0xC5A13E396,$$

then the attacker can detect this fact efficiently.

It should be noted that this equality on 36 bits normally happens with probability 2^{-36} except in some special cases such as few steps after $0xC5A13E396$, in which cases this probability is lower. However we do not see a method for the attacker to obtain a better probability than 2^{-36}, the attacker needs to try many cases where this property can happen accidentally and eventually he will succeed.

23 Slide Property Detection With Decryption Oracle and Internal Correlations

Here is how the attacker can detect/confirm of his guess is correct:

Theorem 23.0.2 (Sliding Property Detection with a Decryption Oracle). For every IV chosen by the attacker, and for every $s \geq 1$, the attacker can detect if $u_{120s+0} = u_0$ with near-certainty with decryption of $K \approx \sqrt{\log_2(k)}$ texts of length $120s + \mathcal{O}(1)$ and time complexity about in $\sqrt{\log_2(k)}$ as in Thm. 21.1.1 where k is fixed integer for every s, and in general it grows exponentially with[43] d.

Proof: We describe how the distinguisher works in four Steps 1-4.

Step 1. We select two IVs which are distant by $120s$ steps of our 61-bit LFSR, called IV, IV'. We recall that $120s$ mod 127 is small. We recall that the key is repeated after every multiple of 120 rounds, but the keystream is extracted every 127 rounds. Then IF in some two encryptions have the same state

$$u_0 = u'_{120s} \quad \text{[Sliding Assumption]}$$

which occurs with probability 2^{-36} THEN we have

$$u_i = u'_{120s+i}$$

for any number of steps $i \geq 0$.

Step 2. Then it is easy to see that for both encryptions the attacker can recover the keystream with roughly at most 4 decryption queries per IV, cf. Thm. 21.1.1.

Step 3. Now we have $120s = 127t{+}d$, where d is small. For example $(s, t, d) = 18, 17, 1$. This means that IF again $u_0 = u'_{120s}$ the keystream extracted from the second encryption is shifted by $127t + 1$, i.e. it is extracted at t "big" a_i-scale steps later with 127 rounds each, and with d-round iteration of ϕ offset. So we

[43] Here d is as described earlier. It could also be defined as the integer which minimizes $d = 120s - 127t$ in the absolute value.

cannot hope that these bits will be identical BUT we can hope they will be somewhat correlated to each other. We have

$$a_j = u_{127j,\alpha}$$

and

$$a'_j = u'_{127j,\alpha} = u'_{127(j-t)-d+120s,\alpha} = u_{127(j-t)-d,\alpha}$$

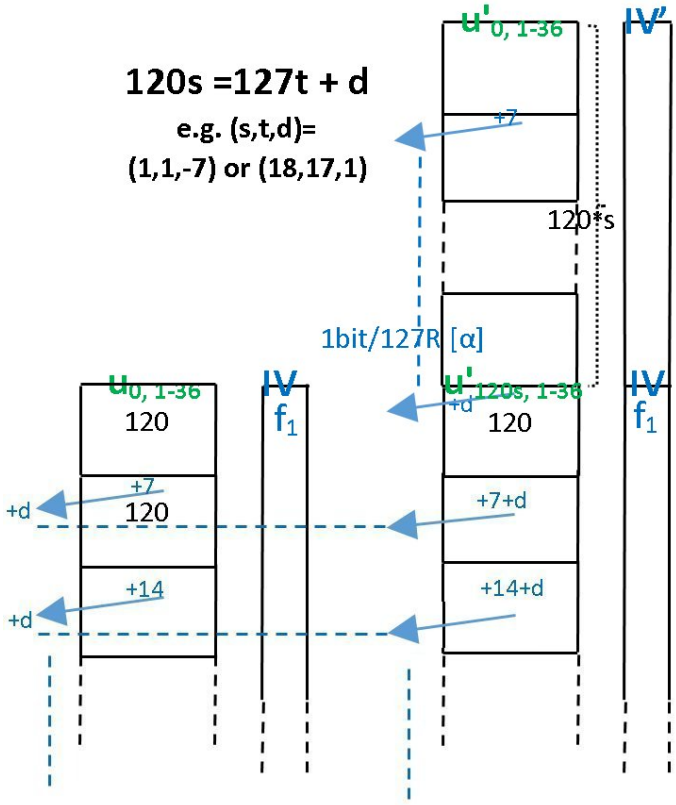


Fig. 23.10. Slide Attacks on T-310 - the IVs are identical at positions which are also distant by a multiple of 120, the keystream is shifted by ϕ^d, where d can be positive or negative.

23.1 Step 4 - Simplified Correlation Analysis

Now as a first approximation, we see that the attacker has access to sequences $u_{127j,\alpha}$ and $u_{127j'-d,\alpha}$ for any j, j' which are shifted by $d = 1$ encryption round ϕ. The question now is if there is a **correlation** between these 2 bits which makes that the slide assumption $u_0 = u'_{120s}$ will be detected.

This depends on the values of α, d, the particular correlation characteristics of the Boolean function Z used (such as correlation immunity), and on the long-term key D, P. We conjecture that for every α there exists one or several d

such that our attack works. For example we can use our example of a long-term key described in Section 8. Here for $d = 1$ we have observed that we have $u_{127j-1,\alpha} = u_{127j,\alpha}$ with probability $0.5 - \varepsilon$ with $\varepsilon = 2^{-3}$ which means that the attacker can easily detect if our sliding condition on 36 bits is true for $\alpha = 17$ [similar results can be obtained for other values of α].

Then for $d = -7$ we have observed that we have $u_{127j+7,\alpha} = u_{127j,\alpha}$ with probability $0.5 - \varepsilon$ with $\varepsilon = 2^{-11}$ which means that the attacker can easily detect if our sliding condition on 36 bits is true for $\alpha = 21$ [similar results can be obtained for other values of α].

23.2 Step 4' - Actual Correlation Analysis

Things are less simple than in the section above. It is NOT true that the attacker has access to sequences $u_{127j,\alpha}$ and $u'_{127j,\alpha}$ for every j. More precisely, following Section 16, only 10 out of every 13 bits from Thm. 21.1 can be recovered on each side. This makes that only some pairs $u_{127j,\alpha}, u'_{127(j-t),\alpha}$ will be actually available, actually a proportion of $(10/13)^2 \approx 0.59$.

This is of course sufficient to detect the correlation with about twice higher k than expected previously. Quite happily we can easily afford up to $k = 4$ million with just $K = 4$ decryptions with the same chosen IV, cf. Thm. 21.1.1. This is of course sufficient to detect the correlation with about twice higher k than expected previously. Again for k up to 2^{22} we need just $K = 4$ decryptions per IV, and for k up to 2^{40} we need $K = 5$. This ends the proof of Thm. 23.0.2.

23.3 Sliding Step - Summary

We see that the attacker can obtain P/C pairs on 36+36 bits for the T-310 block cipher for $120s$ rounds away and with arbitrarily chosen IVs, and where the second IV is obtained by closing by clocking the LFSR $120s$ steps backwards.

More precisely, following Thm. 23.0.2 the attacker can **detect** if the internal states on the 36 bits are identical. He can know with near-certitude that

$$u_0 = u'_{120s} \quad \text{[Sliding Assumption]}$$

is true for some pairs IV, s and for the current secret key.

This condition is true with probability 2^{-36} in general and when it occurs the attacker will detect it.

23.4 A Basic Full Sliding Key Recovery Attack with $d = -7$

The question is now HOW to break this block cipher knowing that the attacker can identify P/C pairs for 120 rounds with $s = 1$. We can thew follow the whole process described above more than once and obtain several P/C conditions on 36 bits (one is not sufficient to uniquely determine a key on 230 bits).

1. We have $d = -7$ and $s = 1$.
2. The attacker try some $8 \cdot 2^{36} = 2^{39}$ IVs on 61 bits, to discover some $2^{39-36} \approx 8$ "good" IVs where he has $u_{120s} = u_0 = 0xC5A13E396$. At this moment he does not know which IVs are the "good" ones.
3. For each of IV_i, $i = 1 \ldots 2^{39}$ the attacker will step the IV exactly $120s$ step backwards to obtain IV'_i.

4. The pairs IV, IV' are always shifted by a multiple of 120 rounds so that they key bits $s_{i,1-2}$ are also aligned.
5. Memory requirements are very small.
6. Then we apply Thm. 23.0.2 cf. also Fig. 23. The attacker can see if $u_{120s} = u_0 = 0xC5A13E396$ by aligning 2 sequences a_j and a'_{j+t}, where 10/13 bits only are known to the attacker, discarding all the pairs where either of a_j, a'_{j+t} is not known, and counting how many times we have $a_j = a'_{j+t}$.
   In order to estimate the data complexity needed to distinguish beyond reasonable doubt if we observe a correlation, we could use the Gauss error function. Here the rule of thumb is the same as in linear cryptanalysis: if we want to make sure that we are beyond 10 standard deviations, we need approximately $k = 120s + 10^2\varepsilon^2$ of encrypted data.
7. Here for $d = -7$ we have $\varepsilon = 2^{-11}$ so we need about $k \approx 2^{29}$ of encrypted data which will give $K = 5$ in Thm. 23.0.2.
8. The data complexity is about $8 \cdot 2^{39} \approx 2^{42}$ chosen IV chosen ciphertext decryption queries with which are $2^{25.5}$ bytes each in length.
9. The time complexity is about $2^{39} \cdot 2^{29} \approx 2^{69.5}$ CPU clocks spent in examining correlations plus the time to recover the key from 8 P/C pairs for 120 rounds by a SAT solver attack. As long as this step takes less[44] than say 2^{69} CPU clocks, this does NOT change the complexity of our attack.

Overall we see that we can recover the 230-bit key of T-310 with about 2^{39} chosen IV chosen ciphertext decryption queries with messages of less than 2^{29} characters each. The time required is about 2^{65} CPU clocks and memory required is small.

23.5 An Advanced Sliding Key Recovery Attack with $d = 1$

In the case of some (stronger) T-310 keys, or if the SAT solver attack does not work as well as expected, we provide another attack with $d = 1$, cf. Appendix G.

24 On Correlation Immunity in T-310

Correlation immunity, has been an important goal in the design of encryption machines for many decades, cf. [81]. It is possible to see that cases where our Slide-Correlation attack of Section will work with small $s = 1, 2, \ldots$ are very rare. In a cipher such as T-310 there are many methods to insure that for many choices of D, P, α our attacks following Section 22 and will not work. In this section we focus of correlations with $s = 1$ used in our later attack of Appendix G. Similar analysis for $s = 7$ could be very complex to handle. Here are some reasons why such correlations will not exist.

[44] For example, in Table 1, Section 9, page 25 in [34], the time complexity with as number of P/C pairs grows. We expect a similar result here and arguably 120 rounds of T-310 are the equivalent of 8 rounds of GOST in terms of complexity and key usage.

One example would be a consequence of a "good" choice of the Boolean function Z which has a certain level of correlation immunity. In many cases we don't even need to study $Z()$ because a bit at α at input of ϕ^s simply does not affect the same bit α at the output of ϕ^s for a small s and therefore a correlation is impossible.

In another example, a lack of correlation could be deduced from a detailed analysis of Fig. 3.12 showing that even if bit 29 was one of the inputs of $Z1$, which it isn't, the bit u_{20} still prevents any correlation with $s = 1$ and $\alpha = 29$ form existing.

Another example is that lack of correlation can be a result of some bits being not used in T. For example we have the following two easy results:

Theorem 24.0.1. If α is not a multiple of 4 and it is one of the bits not used by T in Table 1 page 26, it is easy to see that there will not be any correlation for $s = 1$ rounds.

Proof: If $\alpha \neq 4k$ it will belong to a branch other than I^1 in Fig. 4.4 page 10. Then after 1 round the perturbation will not affect T and move to another branch. The output at bit α after $s = 1$ rounds is therefore totally independent form the input flip at α and these two bits are therefore not correlated.

We can also obtain a stronger result:

Theorem 24.0.2. If $\alpha, \alpha + 1$ are both present in the list of bits not used by T in Table 1 and if $\alpha = 4k+1$ or $\alpha = 4k+2$ than there will not be any correlation for any of $s = 1, 2$ rounds.

Proof: If $\alpha = 4k{+}1$ or $+2$ it will belong to branch I^4 or I^3 in Fig. 4.4 after 1 round the perturbation will not affect T and move to the branch I^3 or I^2 respectively and become $\alpha + 1$, and after 1 more round it will still not flip anything in T and move to branch I^2 or I^1. A bit flip has just moved to another location different than α. Again, the output at bit α after $s = 1, 2$ rounds is independent form the input flip at α and these two bits cannot be correlated.

Remark. Once our perturbation arrives to branch I^1 it is guaranteed to flip one of the inputs of T, for all KT1 keys, this happens for reasons of $P()$ taking all the possible $4 \cdot k$ values, cf. [76] and Section 5.4.

25 Summary of Strong and Weak Points in T-310

In this paper we study the peculiar internal structure of the T-310 cipher.

25.1 On Per-Round Weakness vs. Number of Rounds

Our key observation is that there is a strong internal triangular structure and that nearly all these properties can be traced back to one single assumption where we disconnect/replace just one bit in $\underline{\mathbf{D}}$ in an unbalanced Feistel scheme. These peculiarities make one round of this cipher clearly weaker than comparable historical block ciphers such as DES or RC2. This is exacerbated by low "per-round" gate/hardware/multiplicative complexity, which is relevant in software algebraic and/or SAT solver attacks, cf. Section 15. Finally we show that T-310 is not very strong against differential cryptanalysis, cf. Section 9. The **question is** now[45] is this "per round weakness" compensated by a larger number of rounds, AND by the fact[46] that[47] extremely few[48] bits extracted from the internal state (1 bit every 127 rounds) are actually used for encryption. Overall we expect that the cryptanalyst will have hard time breaking T-310 and probably the per-round weakness of T-310 is most probably NOT a weakness "per se" and can be mitigated by the fact that T-310 consumes as many as 1651 rounds of the permutation ϕ per each encrypted character.

25.2 Definite Vulnerabilities of T-310

We can however already say that we have identified **several serious vulnerabilities** in the design of T-310. These properties are able to very substantially degrade the security of T-310 **for no apparent reason**. It appears that they simply cannot be defended[49] by any engineering or practical reasons known to us, such as the cost or speed of encryption. Several examples of such properties of T-310 with variable levels of severity are given in Sections 4.5, 7.6, 8.5, 9, 17.2, 20.1, 18,18.9,19, 20.1 and in Section 22. Most of these should be considered as either engineering mistakes or at least as definite areas where the cipher is weaker than it could otherwise be.

[45] This is **not unusual**, the same problem occurs in for example in [34, 18, 32], and also for about half of block ciphers submitted to the NIST AES competition in the late 1990s.

[46] This is **quite unusual**, and here T-310 appears to be substantially more robust than nearly any other cipher known in crypto literature.

[47] However overall this situation is not unusual if we look at broader context in which one cipher could be used, for example to obtain a realistic card-only attacks MiFare classic cipher, the extremely low quantity of data which the attacker can dispose of is due to an extremely well engineered protocol in which the cipher is used, so that the reader must authenticate first, and very little data can be obtained by the attacker, at a price of exploiting an additional bug and a weak RNG, cf. [29].

[48] Needless to say smart people in the industry have known this for years, see for example [29, 60].

[49] This sort of properties are usually due to the to the inclination by the designers to mandate some simple and elegant internal structure and the popularity of certain ideas among cipher designers.

Some of these open interesting possibilities to potentially misuse the cipher and make it weaker on purpose just by selecting a weak long-term key, see for example Section 4.5, or in Section 19.

In a future version of this paper we are going to present a more detailed evaluation of the complexity of various attacks on T-310.

26 Conclusion and Summary of Our Attacks on T-310

T-310 is an important Cold War cipher. In this paper we study the peculiar internal structure of the T-310 cipher and show that it has several serious vulnerabilities, but also that it is very strong in the sense that it extracts extremely few bits for the actual encryption and a very large number of rounds will be used to encrypt just one character of the plaintext. This property makes that T-310 seems substantially stronger than other ciphers form the same historical period such as RC2, DES, or Skipjack. Cryptanalytic literature knows extremely few examples where the cipher would actually be broken under such difficult circumstances. In one such example the attacker obtains only 4 bits from each larger encryption [29]. In T-310 bits from round as high as 1397 are used to encrypt just the first character. In spite of this difficulty in this paper we propose several attacks on T-310.

For example our sliding attack on T-310 in Section 23.4, allows one to recover the 230-bit key of T-310 with about 2^{39} chosen IV / chosen ciphertext decryption queries, which need to be 2^{26} characters long. The time required is about 2^{65} CPU clocks to recover a 230-bit key and memory required is small. This attack requires some correlations to exist and will work **only** for some keys D, P, α, and will not work for any of the actual historical keys. Then in Appendix G we present another more complex sliding attack which uses another type of correlations with $s = 1$. Other similar slide attacks will appear in a future revision of this paper. Our preliminary conclusion here is that the designers of T-310 have made it quite immune to correlations required by such attacks, cf. Section 24.

Another very important attack on T-310 is given in Section 20. We see that the combination of regular periodic structure, deficient KT2 or other keys, can lead to very strong attacks in spite of the fact that the IV expansion destroys the perfectly periodic structure. We present a ciphertext-only correlation attack which seems to work for **every** single weak key known to us, cf. Table 9. For example with $\beta = 2^{-8.8}$ for key 207 the attacker can recover the full 230-bit encryption key in time of 2^{90} given about 2^{50} characters of encrypted data in the ciphertext-only scenario. Or with key 27, we have $\beta = 2^{-5.4}$ and time complexity will be 2^{83} and we need 2^{43} characters of ciphertext. It is extremely rare to see a ciphertext-only attack on a real-life government cipher[50].

This result shows that there are serious possibilities for degrading the security of T-310 by the choice of LZS, and we stress the fact that it quite difficult to check all of some forty conditions which the LZS of type KT2 must satisfy, and it would not be practical to check them routinely in the 1980s. At the same time in this paper we show that both KT1 class, cf. Thm. C.10.1 page 69 and KT2 class of long-term keys, cf. Thm. D.6.1 page 75, are mathematically proven secure against this sort of potentially devastating ciphertext-only attacks.

[50] This for example was not the case for Enigma during WW2 and the first ciphertext-only attack on Enigma was found only in 1995, cf. [53, 69].

References

1. Tugkan Batu, Sanjoy Dasgupta, Ravi Kumar, Ronitt Rubinfeld: *The complexity of approximating the entropy,* extended version of a paper which appeared in the 34th ACM Symposium on Theory of Computing, pp. 678687, 2002, revised April 2005, `http://cseweb.ucsd.edu/ dasgupta/papers/jent0409.pdf`
2. Om Bhallamudi: *Tool for Software Algebraic attacks on T-310 block cipher,* `http://www.nicolascourtois.com/software/codegen_last.zip`.
3. Mark Briceno, Ian Goldberg, and David Wagner. *GSM Cloning* online paper and web page. `http://www.isaac.cs.berkeley.edu/isaac/gsm-faq.html`, 1998.
4. Billy Brumley: *A3/A8 & COMP128,* slides from T-79.514 Special Course on Cryptology, Helsinki University of Technology, November 2014, `http://www.tcs.hut.fi/Studies/T-79.514/slides/S5.Brumley-comp128.pdf`
5. E.F. Brickell, D.E. Denning, S.T. Kent, D.P. Maher, W. Tuchman, *SKIPJACK Review Interim Report The SKIPJACK Algorithm* , 8 June 2011, archived at `https://web.archive.org/web/20110608020227/ http://www.cs.georgetown.edu/ denning/crypto/clipper/SKIPJACK.txt`
6. Don Coppersmith, *The development of DES, Invited Talk, Crypto'2000, August 2000.*
7. *Nicolas Courtois: Security Evaluation of GOST 28147-89 In View Of International Standardisation,* in Cryptologia, Volume 36, Issue 1, pp. 2-13, 2012. `http://www.tandfonline.com/toc/ucry20/36/1` An earlier version which was officially submitted to ISO in May 2011 can be found at `http://eprint.iacr.org/2011/211/`.
8. Nicolas Courtois, Guilhem Castagnos and Louis Goubin: *What do DES S-boxes Say to Each Other ?* Available on `eprint.iacr.org/2003/184/`.
9. 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.
10. Nicolas Courtois: *Data Encryption Standard (DES),* slides used in GA03 Introduction to Cryptography and later in GA18 course Cryptanalysis taught at University College London, 2006-2016, `http://www.nicolascourtois.com/papers/ des_course_6.pdf`
11. Nicolas Courtois: *The Best Differential Characteristics and Subtleties of the Biham-Shamir Attacks on DES,* On `eprint.iacr.org/2005/202`.
12. Nicolas Courtois: *Feistel Schemes and Bi-Linear Cryptanalysis,* in Crypto 2004, LNCS 3152, pp. 23-40, Springer, 2004.
13. Nicolas Courtois, Theodosis Mourouzis, Anna Grocholewska-Czurylo and Jean-Jacques Quisquater: *On Optimal Size in Truncated Differential Attacks,* In postproceeding of CECC 2014 conference, Studia Scientiarum Mathematicarum Hungarica.
14. 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`. Also explained in Appendix B of the extended version of [15].
15. 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.

16. Nicolas Courtois: *Software and Algebraic Cryptanalysis Lab, a lab used in GA18 course Cryptanalysis taught at University College London, 17 March 2016,* `http://www.nicolascourtois.com/papers/ga18/AC_Lab1_ElimLin_Simon_CTC2.pdf`
17. *Nicolas Courtois: Some algebraic cryptanalysis software by Nicolas T. Courtois,* `http://www.cryptosystem.net/aes/tools.html`.
18. *Nicolas T. Courtois, Iason Papapanagiotakis-Bousy, Pouyan Sepehrdad and Guangyan Song: Predicting Outcomes of ElimLin Attack on Lightweight Block Cipher Simon,* In Secrypt 2016 proceedings.
19. Nicolas T. Courtois, Daniel Hulme and Theodosis Mourouzis: *Multiplicative Complexity and Solving Generalized Brent Equations With SAT Solvers,* In COMPUTATION TOOLS 2012, ISBN 978-1-61208-222-6, pp. 22-27, 22 July 2012, best paper award.
20. Nicolas T. Courtois, Daniel Hulme and Theodosis Mourouzis: *Solving Circuit Optimisation Problems in Cryptography and Cryptanalysis,* In proceedings of SHARCS 2012 workshop, pp. 179-191, `http://2012.sharcs.org/record.pdf`. An abridged version appears in the electronic proceedings of the 2nd IMA conference Mathematics in Defence 2011, UK.
21. Nicolas T. Courtois: *Low-Complexity Key Recovery Attacks on GOST Block Cipher,* In Cryptologia, Volume 37, Issue 1, pp. 1-10, 2013.
22. Nicolas Courtois, Jerzy A. Gawinecki, Guangyan Song: *Contradiction Immunity and Guess-Then-Determine Attacks On GOST,* In Tatra Mountains Mathematic Publications, Vol. 53 no. 3 (2012), pp. 65-79. At `http://www.sav.sk/journals/uploads/0114113604CuGaSo.pdf`.
23. Nicolas Courtois and Willi Meier: *Algebraic Attacks on Stream Ciphers with Linear Feedback,* Eurocrypt 2003, LNCS 2656, pp. 345-359, Springer, 2003. A long, extended version of this paper is available from `www.nicolascourtois.com`.
24. Nicolas Courtois: *Algebraic Attacks on Combiners with Memory and Several Outputs,* ICISC 2004, LNCS 3506, pp. 3-20, Springer 2005.
25. Nicolas Courtois, Gregory V. Bard, David Wagner: *Algebraic and Slide Attacks on KeeLoq,* In FSE 2008, pp. 97-115, LNCS 5086, Springer, 2008.
26. Nicolas T. Courtois and Gregory V. Bard: *Random Permutation Statistics and An Improved Slide-Determine Attack on KeeLoq,* In LNCS 6805, pp. 35-54, Springer, 2012.
27. Gregory V. Bard, Shaun V. Ault and Nicolas T. Courtois: *Statistics of Random Permutations and the Cryptanalysis Of Periodic Block Ciphers,* In Cryptologia, Vol. 36, Issue 03, pp. 240-262, July 2012.
28. Nicolas Courtois: *La Carte à Puce,* 293 slides in English, overview of smart card technology, part of COMPGA12 course taught at University College London in 2007-2017, `http://www.nicolascourtois.com/papers/smartc.pdf`
29. Nicolas T. Courtois: *The Dark Side of Security by Obscurity and Cloning MiFare Classic Rail and Building Passes Anywhere, Anytime, In SECRYPT 2009 International Conference on Security and Cryptography: pp. 331-338. INSTICC Press 2009, ISBN 978-989-674-005-4.*
30. *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`.
31. Nicolas Courtois: *General Principles of Algebraic Attacks and New Design Criteria for Components of Symmetric Ciphers,* in AES 4 conference proceedings, Bonn May 10-12 2004, in LNCS 3373, pp. 67-83, Springer, 2005.

32. Nicolas Courtois, Theodosis Mourouzis, Guangyan Song, Pouyan Sepehrdad and Petr Susil: *Combined Algebraic and Truncated Differential Cryptanalysis on Reduced-round Simon,* in post-proceedings of SECRYPT 2014, 28-30 August 2014, Vienna, Austria.
33. Nicolas Courtois: *100 years of Cryptanalysis: Compositions of Permutations* slides about cryptanalysis of Engima and block cipher cryptanalysis, used teaching GA18 Cryptanalysis course at University College London 2014-2016, `http://www.nicolascourtois.com/papers/code_breakers_enigma_block_teach.pdf`.
34. Nicolas Courtois: *Algebraic Complexity Reduction and Cryptanalysis of GOST,* Monograph study on GOST cipher, 2010-2014, 224 pages, available at `http://eprint.iacr.org/2011/626`.
35. Nicolas Courtois: *Cryptanalysis of GOST,* a very long extended set of slides about the cryptanalysis of GOST, 2010-2014, `http://www.nicolascourtois.com/papers/GOST.pdf`. An earlier and shorter version was presented at 29C3, see [36].
36. Nicolas Courtois: *Cryptanalysis of GOST, (Security Evaluation of Russian GOST Cipher; Survey of All Known Attacks on Russian Government Encryption Standard.)* Presentation at 29th Chaos Communication Congress (29C3), 27-30 Dec 2012, Hamburg, Germany, `http://events.ccc.de/congress/2012/Fahrplan/attachments/2243_GOST_29C3_long.pdf` A video is available on: `www.youtube.com/watch?v=o_sP0qJam-4`
37. Nicolas Courtois: *On Multiple Symmetric Fixed Points in GOST,* in Cryptologia, Iss. 4, vol 39, 2015, pp. 322-334, `http://www.tandfonline.com/doi/full/10.1080/01611194.2014.988362`
38. 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.
39. Nicolas Courtois: *An Improved Differential Attack on Full GOST,* In Cryptology ePrint Archive, Report 2012/138. 15 March 2012, updated December 2015, `http://eprint.iacr.org/2012/138`.
40. 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.
41. Nicolas Courtois, Karsten Nohl, Sean O'Neil: *Algebraic Attacks on MiFare RFID Chips,* `http://www.nicolascourtois.com/papers/mifar_rump_ec08.pdf`.
42. Nicolas T. Courtois and Louis Goubin: *An Algebraic Masking Method to Protect AES Against Power Attacks,* `https://eprint.iacr.org/2005/204.pdf`
43. Itai Dinur, Adi Shamir: *Cube Attacks on Tweakable Black Box Polynomials,* In Eurocrypt 2009, LNCS 5479, pp. 278 - 299, Springer.
44. Jörg Drobick: *T-310 Schlüsselunterlagen,* a web site about T-310 and siblings of T-310, which enumerates several different known long-term keys for T-310 consulted 21 January 2017, `http://scz.bplaced.net/t310-schluessel.html`
45. Jörg Drobick: *T-310/50 ARGON,* a web page about T-310 cipher machines consulted 19 March 2017, `http://scz.bplaced.net/t310.html`
46. Jörg Drobick: *Analyseergebnisse des Chiffrieralgorithmus ARGON,* a web page about security analysis of T-310 and other encryption algorithms consulted 28 January 2017, `http://scz.bplaced.net/ke-sks-t310.html`
47. Jean-Charles Faugère, Ludovic Perret, Pierre-Jean Spaenlehauer: *Algebraic-Differential Cryptanalysis of DES,* in Proc of WEWoRC, pp 1-5, Springer, slides `http://www.pjspaenlehauer.net/data/slides/slidesC2DES.pdf`

48. H. Feistel, W.A. Notz, J.L. Smith, *Cryptographic Techniques for Machine to Machine Data Communications,* Dec. 27, 1971, Communications, IBM T.J.Watson Research.
49. Horst Feistel: *Cryptography and computer privacy;* Scientific American, vol. 228, No. 5, pp. 15-23, May 1973.
50. William F. Friedman, *The index of coincidence and its applications in cryptology,* , Department of Ciphers. Publ 22. Geneva, Illinois, USA: Riverbank Laboratories, 1922.
51. Flavio D. Garcia, Gerhard de Koning Gans, Ruben Muijrers, Peter van Rossum, Roel Verdult, Ronny Wichers Schreur, Bart Jacobs: *Dismantling MIFARE Classic,* Radboud University Nijmegen, Netherlands. In Esorics 2008, 13th European Symposium on Research in Computer Security, Malaga, Spain, 6-8 October, 2008
52. Gemplus Combats SIM Card Cloning with Strong Key Security Solution, Press release, Paris, 5 November 2002, see `http://www.gemalto.com/press/gemplus/2002/r_d/strong_key_05112002.htm`.
53. J. J. Gillogly, *Ciphertext-only cryptanalysis of Enigma, In Cryptologia 19 (4):321413, 1995.*
54. *Jovan Golic, Cryptanalytic Attacks on MIFARE Classic Protocol,* In CT-RSA 2013, LNCS 7779, pp. 239–258, Springer, 2013.
55. Jovan Dj. Golic, Christophe Tymen: *Multiplicative Masking and Power Analysis of AES,* In CHES 2002, LNCS 2523, pp. 198-212, Springer, 2003.
56. 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.
57. Viet Tung Hoang, Phillip Rogaway: *On Generalized Feistel Networks,* In Crypto 2010, LNCS 6223, pp. 613-630, Springer 2010.
58. Jongsung Kim, Raphael C.W. Phan: *Advanced Differential-Style Cryptanalysis of the NSA's Skipjack Block Cipher,* In Cryptologia, vol. 33, iss. 3, pp. 246-270, 2009.
59. Lars R. Knudsen: *Truncated and Higher Order Differentials,* In FSE 1994, pp. 196-211, LNCS 1008, Springer.
60. Paul C. Kocher, Joshua M. Jaffe, Benjamin C. Jun: *Prevention of side channel attacks against block cipher implementations and other cryptographic systems,* US Patent US 7787620 B2, `https://www.google.ch/patents/US7787620`
61. Lars R. Knudsen, Vincent Rijmen, Ronald L. Rivest, Matthew J. B. Robshaw: *On the Design and Security of RC2,* In FSE'98, LNCS 1372, pp. 206221, Springer, 1998.
62. Jiqiang Lu, Jongsung Kim, Nathan Keller, Orr Dunkelman: *Differential and Rectangle Attacks on Reduced-Round SHACAL-1,* , In INDOCRYPT 2006, pp. 1731, Springer 2006.
63. A.P. Mahon: *June 1945. The History of Hut Eight 19391945,* Catalogue Reference HW 25/2. The National Archives, UK. `http://ellsbury.com/hut8/hut8-000.htm`.
64. Mitsuru Matsui: *Linear Cryptanalysis Method for DES Cipher,* Eurocrypt'93, LNCS 765, Springer, pp. 386-397, 1993.
65. 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.
66. Jacques Patarin, Valérie Nachef, Côme Berbain: *Generic Attacks on Unbalanced Feistel Schemes with Expanding Functions,* in Asiacrypt 2007: pp. 325-341, Springer, 2007.

67. Valérie Nachef, Emmanuel Volte, Jacques Patarin: *Differential Attacks on Generalized Feistel Schemes,* In CANS 2013: pp. 1–19.
68. Elisabeth Oswald, Stefan Mangard, Norbert Pramstaller: *Secure and Efficient Masking of AES - A Mission Impossible?,* eprint.iacr.org/2004/134
69. Olaf Ostwald, Frode Weierud: *Modern breaking of Enigma ciphertexts,* , In Cryptologia, published online: 31 Jan 2017.
70. Ludovic Perret: *Gröbner bases techniques in Cryptography,* http://web.stevens.edu/algebraic/Files/SCPQ/SCPQ-2011-03-30-talk-Perret.pdf
71. Joan Boyar, René Peralta: *A New Combinational Logic Minimization Technique with Applications to Cryptology.* In SEA 2010: 178-189.
An early version was published in 2009 at http://eprint.iacr.org/2009/191. It was revised 13 Mar 2010.
72. Axel Poschmann, San Ling, and Huaxiong Wang: *256 Bit Standardized Crypto for 650 GE GOST Revisited,* In CHES 2010, LNCS 6225, pp. 219-233, 2010.
73. Vincent Rijmen, Bart Preneel: *Cryptanalysis of McGuffin,* In FSE '94, pp. 353–358, Springer, 1994.
74. Matthieu Rivain: *On the Physical Security of Cryptographic Implementations,* PhD thesis, 22 September 2009, University of Luxembourg.
75. Matteo Scarlata: *t310-diff,* Differential analysis tool for the phi function of the German T-310/50 "ARGON" cipher, developed as part of the GA18-Cryptanalysis course at UCL. Written in Python. https://gitlab.com/mtscr/T-310.
76. Klaus Schmeh: *The East German Encryption Machine T-310 and the Algorithm It Used,* In Cryptologia, 30: 3, pp. 251 257, 2006.
77. Maciej Skorski, *Shannon Entropy versus Renyi Entropy from a Cryptographic Viewpoint,* Eprint https://eprint.iacr.org/2014/967.pdf
78. Guangyan Song: *Optimization and Guess-then-Solve Attacks in Cryptanalysis,* PhD thesis, draft, will be presented at University College London in 2017.
79. Petr Susil, Pouyan Sepehrdad, Serge Vaudenay, Nicolas Courtois: *On selection of samples in algebraic attacks and a new technique to find hidden low degree equations.* in International Journal of Information Security vol. 15 iss. 1, pp. 51-65, Springer, 2016.
80. *Kryptologische Analyse des Chiffriergeräts T-310/50. Ministerium für Staatssicherheit, Berlin, 1980.*
81. *T. Siegenthaler: Correlation-immunity of nonlinear combining functions for cryptographic applications,* In IEEE Trans. on Inf. Th. 30(5): (1984).
82. Ko Stoffelen: *Optimizing S-box Implementations for Several Criteria using SAT Solvers,* In FSE 2016, cf. also https://eprint.iacr.org/2016/198
83. Michael Vielhaber: *Breaking ONE.FIVIUM by AIDA an Algebraic IV Differential Attack,* In https://eprint.iacr.org/2007/413.
84. Wikipedia article: *Birthday problem,* consulted 12 March 2017, https://en.wikipedia.org/wiki/Birthday_problem.
85. Wikipedia article: *Standard Deviation,* consulted 12 March 2017, http://en.wikipedia.org/wiki/Standard_deviation.

Appendix.

A Glossary

We provide tentative English translations for a selection of German terms used in T-310 literature. We do not guarantee the correctness of these translations.

Chiffrierverfahrens	encryption method
LZS, Langzeitschlüssel	long-term key (up to 94 bits)
Zeitschlüssel	current secret key (230+10 bits)
SpS, Spruchschlüssel	spelling key, the IV (61 bits)
Synchronfolge SYF	synchronisation sequence SYF, contains the IV
bestehend aus 25 FS-zeichen	consisting of 25 teletype characters
FS, Fernschreiber	teleprinter
physikalischer Zufallsgenerator	RNG, random number generator
PZG, Pseudozufallsgenerator	PRNG, pseudo-random number generator
Eingabeeinheit	input unit
Komplizierungsalgorithmus	complication algorithm
Komplizierungseinheit	complication unit
Zwischenfolge	sequence
Verschlüsselungseinheit	encryption unit
Bildung der Steuerfolge a_i	formation of the control sequence a_i
Synchronisationseinheit	synchronization unit
Erzeugung der Spruchschlüssel	generation of IVs / spelling keys
und seine Umkodierung in SYF	and their recoding in SYF
Umkodierung der empfangenen SYF	recoding the received SYF
in Spruchschlüssel (gerführtes gerät)	in a device controlled by IV
Kryptologische Eingenschaften	cryptologic characteristics
Minimalperiode	the period (for a sequence)
Primzahl	prime number
0-1 verhältnis statistisch	ratio/proportion of 0/1, balancedness
Grundtext-Geheimtext-Paares	plaintext-ciphertext pairs
Sicherheit der Chiffrierverfahren	security of encryption procedures
gegen dekryptierung	against decryption
Gebrauchsanweisung	instructions for use
zwei LZS-Klassen	two classes of long-term keys
eineindeutig	bijective

B A Description of KT1 Keys

Here we provide a description of key class KT1 following pages 58 and Section 2.2 of Annex 1 on pages 114-115 and also Section 4.1. page 117 in [80]. An incomplete (and therefore not quite correct) description which only included the conditions from page 58 of [80] can be found in [76]. Here we provide a complete set of conditions which define KT1.

$(P, D, \alpha) \in KT1 \Leftrightarrow$ all the following hold:

$$
\begin{cases}
D \text{ and } P \text{ are injective} \\
P(3) = 33, P(7) = 5, P(9) = 9, P(15) = 21, P(18) = 25, P(24) = 29 \\
\text{Let } \; W = \{5, 9, 21, 25, 29, 33\} \\
\forall_{1 \geq i \geq 9} D(i) \notin W \\
\alpha \notin W \;\; \text{(note: cf. also Fig. 9.9 page 22)} \\
\text{Let } \; T = (\{0, 1, \ldots, 12\} \backslash W) \cap (\{P(1), P(2), \ldots, P(24)\} \cup \{D(4), D(5), \ldots, D(9)\} \cup \{\alpha\}) \\
\text{Let } \; U = (\{13, \ldots, 36\} \backslash W) \cap (\{P(26), P(27)\} \cup \{D(1), D(2), D(3)\}) \\
|T \backslash \{P(25)\}| + |U \backslash \{P(25)\}| \leq 12 \\
D(1) = 0 \\
\text{There exist } \; \{j_1, j_2, \ldots, j_7, j_8\} \text{ a permutation of } \{2, 3, \ldots, 9\} \;\; \text{which} \\
\quad \text{defines } D(i) \text{ for every } i \in \{2, 3, \ldots, 9\} \;\; \text{as follows:} \\
\quad D(j_1) = 4, D(j_2) = 4j_1, D(j_3) = 4j_2, \ldots, D(j_8) = 4j_7 \\
P(20) = 4j_8 \;\; \text{(note: this value is not any of the } \; D(i)) \\
(D(5), D(6)) \in \{8, 12, 16\} \times \{20, 28, 32\} \;\cup\; \{24, 28, 32\} \times \{8, 12, 16\} \\
P(6) = D(8) \\
P(13) = D(7) \\
P(27) \neq 0 \bmod 4 \\
\forall_{1 \geq l \geq 9} \exists_{1 \geq i \geq 26} P(i) = 4 \cdot l \\
D(3) \in \{P(1), P(2), P(4), P(5)\} \\
D(4) \notin \{P(14), P(16), P(17), P(19)\} \\
\{P(8), P(10), P(11), P(12)\} \cap \{D(4), D(5), D(6)\} = \emptyset
\end{cases}
$$

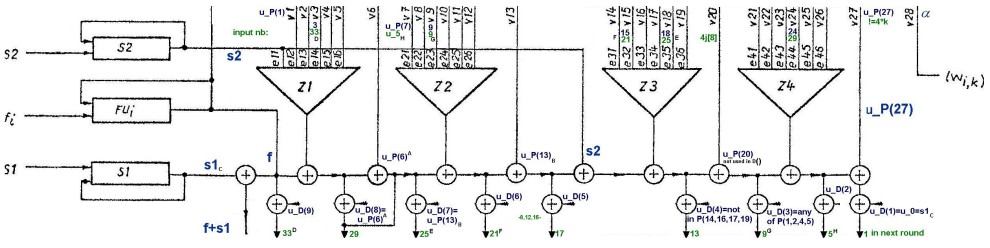


Fig. 2.11. Some observations about internal dependencies inside one encryption round ϕ, which hold for all KT1 keys (many also work with KT2), cf. also Fig. 9.9 page 22.

C On Bijectivity of One Round ϕ

In this section we study the question whether the round function is always a permutation, and what would be the [security] consequences of ϕ not being a bijection. In theory, from a pure encryption point of view, nothing forces ϕ to be invertible. However ϕ is bijective in any version of T-310 we have ever heard of. The original documentation clearly says it must be a bijection cf. pages 47 and 56 in [80]. It appears that if ϕ is required to be always bijective, this will be for security reasons, not for pure functional encryption reasons. Bijective ϕ allows to maintain high entropy of the state u_t at any time t.

C.1 Vanishing Differential Attacks

More importantly, this property of ϕ being bijective is able to prevents some **very strong** attacks on block ciphers. Such attacks are very well known for example in mobile telephone SIM cards. These attacks can be called by many different names such as vanishing differentials, all-zero output difference attacks, collision attacks or "Narrow-Pipe" attacks.

For example in the last 20 years it was relatively easy[51] to extract keys from SIM cards for certain mobile phone operators, and these attacks these keys exploit precisely vanishing differentials cf. [3, 4, 35, 28]. In general, the question of avoiding such rather strong differential properties is precisely the reason why we have many bijections in the design of block ciphers and hash functions. For example S-boxes in SERPENT, PRESENT, GOST [9, 39] and many other ciphers are permutations on 4 bits. In DES S-boxes are also always bijective and are on 4 bits when two (first and last) input bits are fixed [8].

C.2 Weaker Types of Vanishing Differentials

Vanishing differentials can also be applied at a different level: to a round function of a block cipher. It is possible to see that they exist for the DES round function, but only when we involve at least 3 consecutive S-boxes, cf. [8, 11] and they do not exist in GOST cf. slide 255 in [35]. These properties have been carefully engineered by the designers or DES, cf. slide 31 in [10] and [6, 8].

In T-310 it is also possible to find differentials which vanish totally. First, this is inevitable because $T()$ is of compressing type and cannot be bijective. Future works are going to show how good differential properties for iterated ϕ could be in the best case. However knowing that only up to 9 inputs of any of

[51] We and our students have ourselves extracted large numbers of keys from SIM cards as recently as in 2012 primarily for Chinese SIM cards, and we have also discovered that certain European mobile operators still used COMP128v1 until 2012. The basic attack was first outlined by Briceno-Goldberg-Wagner cf. [3], and is also described in page 6-15 in [4] and in Section 13.1 slides 249-255 in [35]. Moreover there exist more efficient variants of this attack cf. [28] which we have developed ourselves, cf. slide 230 in [28] and [52], and as far as we can see the full account of these attacks and their complexity have never yet been published so far. These attacks do not concern SIM cards which use more recent crypto algorithms.

the four Boolean functions $Z()$ are repeated elsewhere (due to being an output of $D()$), it is hard to imagine an impossibility result [52] here.

There exist also very simple vanishing differentials due to the fact that the round function T does not use all the bits it potentially be using. In this cases some bits which are flipped will be some of those not used by T cf. Table 1 page 26. This gives very good differentials for 3 rounds such that only 1 bit is flipped at the input, and only one at the output (with same IV) cf. Section 12.2.

C.3 Related Properties: Beyond All-Zero Differentials

Departing from the vanishing differentials, in T-310 it is also possible to obtain output differentials where exactly 35 bits out of 36 are flipped with different IVs cf. Fact. 12.4.1.

It is not clear however if or how the properties with 1 or 35 bits flipped mentioned above could be exploited by the attacker. In general some relatively strong differential properties in isolation will not yet allow to construct an interesting differential attack.

C.4 Are Vanishing Differentials A Problem in T-310?

In T-310 it is possible to see that vanishing differentials are less a threat than in other ciphers, due to the action of the IV which generates a complex a-periodic pattern. However T-310 is still vulnerable to some very powerful attacks. For example if ϕ was sometimes not bijective, we could have a situation where 2 encryptions with the same IV would collide for example on the first X rounds, and then a distinguisher [possibly a ciphertext-only distinguisher based on Friedman's Index of Coincidence [50, 63]] which shows that the keystream is identical starting from this point. This would make the cipher **extremely** easy to break by a software/algebraic attack or brute force attack. Such an attack would be easy because the first X rounds use only $2X$ key bits and for any previous rounds we could potentially avoid guessing the key but only the current state(s) at a certain location(s) which would be guessing only 36 bits par cipher state $u_{.,1-36}$ to be guessed.

C.5 On The Group Generated by Bijections in T-310 Cipher

The function has 3 key/IV bits s_1, s_2, f which makes that T-310 operates with combinations of exactly 8 fixed permutations on 36 bits which are called $\phi_0, \ldots, \phi_7$ in Section 1.5 in [80]. The document also calls $G(P, D)$ the group generated by these 8 permutations and contains some interesting results about composition of these permutations. From the cryptanalyst point of view it is crucial that this group $G(P, D)$ is quite[53] large.

[52] For example that that we cannot find a flip on few inputs of just one of the $Z()$ which would vanish immediately and give the same output bit $Z(.)$ with a probability $\neq 0$.

[53] If this group is small, the cipher would probably insecure. The opposite does **not** hold: Even if the group is so large that it contains all possible permutations, the security can still be very poor, see [14, 15].

C.6 Bijections vs. KT1/KT2 Classes of Long-Term Keys

It is not sufficient to say that ϕ should be a bijection, cf. [80]. It appears that all currently known long-term keys cf. Section 8 lead to bijective ϕ, and that the designers of two well-known classes of long-term keys KT1/KT2 have mandated that ϕ is going to be a bijection. It is not hard to see that a less strict set of rules can also lead to a bijection, cf. Section F.1, which key is NOT not compliant with all the rules of KT1 and nevertheless gives a bijection.

Nevertheless it appears that previous publications and historical resources have NOT mathematically proved that KT1 or KT2 will always be a bijection or at least such a proof does not appear in [80]. This property is crucial, and we cannot understand the security of T-310 for as long as we are not able to tell if KT1 or KT2 rules would allow the long-term key to be non-bijective which would allow some very powerful attacks such as described in Section C. We either need a mathematical proof that KT1/KT2 are secure, or to demonstrate that an attack is possible.

In this paper we finally resolve this question. First we are going to prove that ϕ is invertible for one historical key numer 26 and we will also show that there is more than one order in which the inversion can be performed. Then we provide a complete mathematical proof how exactly the inversion can be performed for all KT1 keys. We plan to resolve the KT2 case in a future update of this paper.

C.7 One Round Operation ϕ

We recall from Section 7.5 the 9 new bits which are created at each round:

$$\begin{aligned}(u_{m+1,1}, u_{m+1,5}, u_{m+1,9}, \ldots, u_{m+1,29}, u_{m+1,33}) &= \\ (U_1, U_2, U_3, \ldots, U_8, U_9) &= \\ \underline{\mathbf{D}}(s_1;\ \ u_{m,I^1}) \oplus \underline{\mathbf{T}}\left(f, s_2,\ \ \underline{\mathbf{P}}(u_{m,I^{1-4}})\right) &= \\ \big(u_{m,D(1)} \oplus T_9(f, s_2, u_{m,P(1-27)}),& \\ u_{m,D(2)} \oplus T_8(f, s_2, u_{m,P(1-27)}),\quad \ldots& \\ \ldots \qquad\qquad u_{m,D(9)} \oplus T_1(f, s_2, u_{m,P(1-27)})\big)&\end{aligned}$$

Now in KT1 keys we have $D(1) = 0$. Wee have then:

$$\begin{aligned}U_1 &= s_{m+1,1} \oplus T_9(f, s_{m+1,2};\ u_{m,P(1)}, \ldots, u_{m,P(27)}) \\ U_2 &= u_{m,D(2)} \oplus T_8(f, s_{m+1,2};\ u_{m,P(1)}, \ldots, u_{m,P(27)}) \\ &\vdots \\ U_9 &= u_{m,D(9)} \oplus T_1(f, s_{m+1,2};\ u_{m,P(1)}, \ldots, u_{m,P(27)})\end{aligned}$$

C.8 How to Invert the Encryption Round ϕ

In this section we will show how ϕ can be inverted for one type of long-term key of type KT1. We need to see how to recover all the missing nine I^1 bits numbered $4, 8, 12, 16, \ldots 36$. All the other bits with numbers $\neq 4k$ are already known. This will be be done potentially in a different way for each different long-term key.

First in Section C.9 and Fig. 3.12 we will show how this can be donee for one particular key number 26. Then in Section C.10 Thm. C.10.1 and Fig. 3.14 we will show it can always be done for all keys if type KT1.

If we put all the outputs of $\underline{\mathbf{D}}$ on the left hand side, and take into account how $T()$ is defined w.r.t previous bit in Section 9, we have already obtained in Section 9.2 that:

$$\begin{aligned}
U_1 \oplus s_1 &= U_2 \oplus u_{D(2)} \oplus u_{P(27)} \\
U_2 \oplus u_{D(2)} &= U_3 \oplus u_{D(3)} \oplus Z_4(u_{P(21-26)}) \\
U_3 \oplus u_{D(3)} &= U_4 \oplus u_{D(4)} \oplus u_{P(20)} \\
U_4 \oplus u_{D(4)} &= U_5 \oplus u_{D(5)} \oplus Z_3(u_{P(14-19)}) \oplus s_2 \\
U_5 \oplus u_{D(5)} &= U_6 \oplus u_{D(6)} \oplus u_{P(13)} \\
U_6 \oplus u_{D(6)} &= U_7 \oplus u_{D(7)} \oplus Z_2(u_{P(7-12)}) \\
U_7 \oplus u_{D(7)} &= U_8 \oplus u_{D(8)} \oplus u_{P(6)} \\
U_8 \oplus u_{D(8)} &= U_9 \oplus u_{D(9)} \oplus Z_1(s_2, u_{P(1-5)}) \\
U_9 \oplus u_{D(9)} &= f
\end{aligned}$$

Here we distinguish Z_1, Z_2, Z_3, Z_4, which by definition are 4 copies of the same Boolean function $Z()$ defined in Section 10.1.

C.9 Example of Inversion for Key 26

We now give a concrete example of such inversion which was developed by our student Maria-Bristena Oprisanu (during GA18 Cryptanalysis course at University College London), cf. Fig. 3.12 below. We recall the necessary definitions:

```
j=3,7,2,6,5,8,4,9 D=0,28,4,32,24,8,12,20,16 P=8,4,33,
16,31,20,5,35,9,3,19,18,12,7,21,13,23,25,28,36,24,15,26,29,27,32,11
```

Accordingly for this key 26 we get the following equations:

$$\begin{aligned}
U_1 \oplus s_1 &= U_2 \oplus u_{D(2)} \oplus u_{11} \\
U_2 \oplus u_{28} &= U_3 \oplus u_4 \oplus Z_4(u_{24,15,26,29,27,32}) \\
U_3 \oplus u_4 &= U_4 \oplus u_{32} \oplus u_{36} \\
U_4 \oplus u_{32} &= U_5 \oplus u_{24} \oplus Z_3(u_{7,21,13,23,25,28}) \oplus s_2 \\
U_5 \oplus u_{24} &= U_6 \oplus u_8 \oplus u_{12} \\
U_6 \oplus u_8 &= U_7 \oplus u_{12} \oplus Z_2(u_{5,35,9,3,19,18}) \\
U_7 \oplus u_{12} &= U_8 \\
U_8 \oplus u_{20} &= U_9 \oplus u_{16} \oplus Z_1(s_2, u_{8,4,33,16,31}) \\
U_9 \oplus u_{16} &= f
\end{aligned}$$

Remark: Here in line U_7 we observe that two of the u_i terms have disappeared which does always happen for KT1 keys which mandate that $D(8) = P(6)$.

We are now ready to explain how inversion can be performed. On Fig. 3.12 below we have added numbers 0-9 in blue to show in which order different bits $u_{4.l}$, $l \leq 9$ in I^1 can be computed, and below we detail how they are computed.

0 First we know $u_0 = s_1$.

1-2 We see that bits u_{16} and u_{12} can be obtained immediately from the U_7, U_8, U_9.

3 Then we observe that $P(27) = 11$ which is not a multiple of 4, (a property always true for KT1 keys cf. [76]) so u_{28} can be computed from U_1, U_2.

4 Then we observe that inputs of Z_2 do not contain any multiples of 4 and are all known. Therefore we can compute u_8.

5 Then due to the fact that $P(13) = D(7)$ for KT1 keys [76], we can compute u_{24}.

6 Then we observe that the only input of Z_3 which is a multiple of 4 is u_{28} which we have already computed. So we can compute u_{32}.

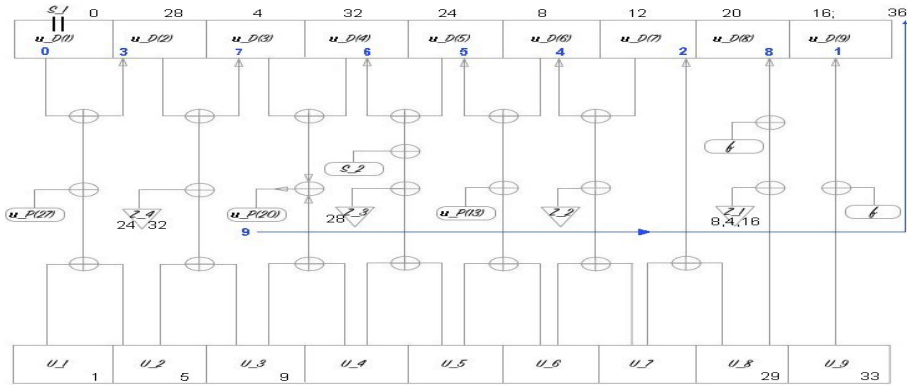


Fig. 3.12. Example of inversion of ϕ for key number 26.

7 Once we know $u_{24,32}$ all the inputs of Z_4 become known and we can compute u_4.

8 Now we know $u_{8,4,16}$ and all other inputs of Z_1, and we can compute u_{20}.

9 We note that $u_{P(20)} = u_{36}$ corresponds to the bit 36 which is NOT used by **D**, and is not in the image of $D()$, which is always mandated in KT1 keys cf. [76]. Until now we have computed 8 input bits without computing or using $u_{P(20)} = u_{36}$. However we need to compute this bit in order to invert ϕ completely. It is now computed simply as $u_{36} = u_{D(3)} \oplus u_{D(4)} \oplus U_3 \oplus U_4$.

This ends the analysis on how inversion is performed for key 26.

Overall our computation above could be very shortly written as the following sequence of events [compact notation]:

```
0 16 12 P27 28 Z2 8 P13 24 Z3 32 Z4 4 Z1 20 P20 36
```

In general this "compact notation" solution sequence is not unique: the order of some but not all events can be altered]. For example another possible solution is:

```
0 P27 28 12 P13 16 Z2 8 24 Z3 32 Z4 4 Z1 20 P20 36
```

This sequence of events will be similar for other keys in KT1 class. We give the general theorem below.

C.10 A Proof The ϕ is Bijective for All KT1 Keys

This proof was also developed together with our student Maria-Bristena Oprisanu. She also has provided the figures to illustrate it, as well as developed a software solution for showing in which order the inversion can be performed for different actual keys, and for checking that such solutions are correct.

For better readability we reproduce here the Fig. 5.5, we recall the compact description of the ϕ function in Section 7.1.

$$\phi\left(s_{m+1,1}, s_{i,2}, f;\ \ u_{m,I^1},\ u_{m,I^2},\ u_{m,I^3},\ u_{m,I^4}\right) =$$
$$\left(\ u_{m,I^2};\ \ u_{m,I^3};\ \ u_{m,I^4};\ \ \underline{\mathbf{D}}(s_{m+1,1};\ \ u_{m,I^1}) \oplus \underline{\mathbf{T}}\left(f, s_{m+1,2},\ \ \underline{\mathbf{P}}(u_{m,I^{1-4}})\right)\ \right)$$

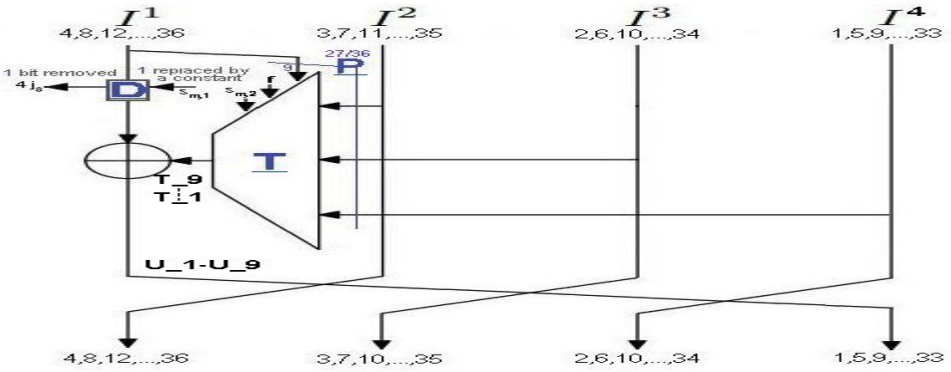


Fig. 3.13. T-310 for KT1 keys as on Fig. 5.5.

Now we recall how all this translates into a set of multivariate equations when $D(1) = 0$, cf. Section 9.2 or Section C.8. We will number these equations (1-9).

$$U_1 \oplus s_1 = U_2 \oplus u_{D(2)} \oplus u_{P(27)} \tag{1}$$
$$U_2 \oplus u_{D(2)} = U_3 \oplus u_{D(3)} \oplus Z_4(u_{P(21-26)}) \tag{2}$$
$$U_3 \oplus u_{D(3)} = U_4 \oplus u_{D(4)} \oplus u_{P(20)} \tag{3}$$
$$U_4 \oplus u_{D(4)} = U_5 \oplus u_{D(5)} \oplus Z_3(u_{P(14-19)}) \oplus s_2 \tag{4}$$
$$U_5 \oplus u_{D(5)} = U_6 \oplus u_{D(6)} \oplus u_{P(13)} \tag{5}$$
$$U_6 \oplus u_{D(6)} = U_7 \oplus u_{D(7)} \oplus Z_2(u_{P(7-12)}) \tag{6}$$
$$U_7 \oplus u_{D(7)} = U_8 \oplus u_{D(8)} \oplus u_{P(6)} \tag{7}$$
$$U_8 \oplus u_{D(8)} = U_9 \oplus u_{D(9)} \oplus Z_1(s_2, u_{P(1-5)}) \tag{8}$$
$$U_9 \oplus u_{D(9)} = f \tag{9}$$

We have the following result:

Theorem C.10.1 (KT1 Invertibility Theorem). For every key in class KT1 as defined in Appendix B and for every 3 bits $s1, s2, f$ the round function ϕ is bijective and given the 36 outputs the internal bits and the 9 input bits of the form $4 \cdot k$ which are the only bits which are modified can be computed in the order defined by the following sequence (written in a compact notation):

`0 D1 P27 D9 D2 D7 P13 Z2 D6 D5 Z3 D4 Z4 D3 Z1 D8 P20`

Proof: We need to recover 9 bits which are of type u_{4k}. For class KT1, cf. Appendix B, it is easy to see that inside these u_{4k} we have 8 which are of type $u_{D(i)}$ and one which is always $u_{P(20)}$. All the remaining 27 bits are known from the start, cf. Fig. 3.13 above. Thus we only need to show how to compute $u_{D(1-9)}$ and then $u_{P(20)}$ given the U_{1-9}.

`D1` We use the notation $D1$ in our compact notation to say that we know from the start that $u_{D(1)} = s_1$.

`P27` We have $P(27) \neq 0 \bmod 4$ for KT1 keys, cf. App. B, therefore we know $u_{P(27)}$.

`D2` The equation (1) can be used to compute $u_{D(2)} = U_1 \oplus s_1 \oplus U_2 \oplus u_{P(27)}$.

`D7` Then we use the fact that $P(6) = D(8)$ in KT1 keys, cf. App. B. Then the equation (7) becomes $U_7 \oplus u_{D(7)} = U_8$ and we can compute $u_{D(7)} = U_7 \oplus U_8$.

`P13` We observe that for all KT1 keys $P(13) = D(7)$, cf. App. B.

`D9` From equation (9) we get: $u_{D(9)} = U_9 \oplus f$.

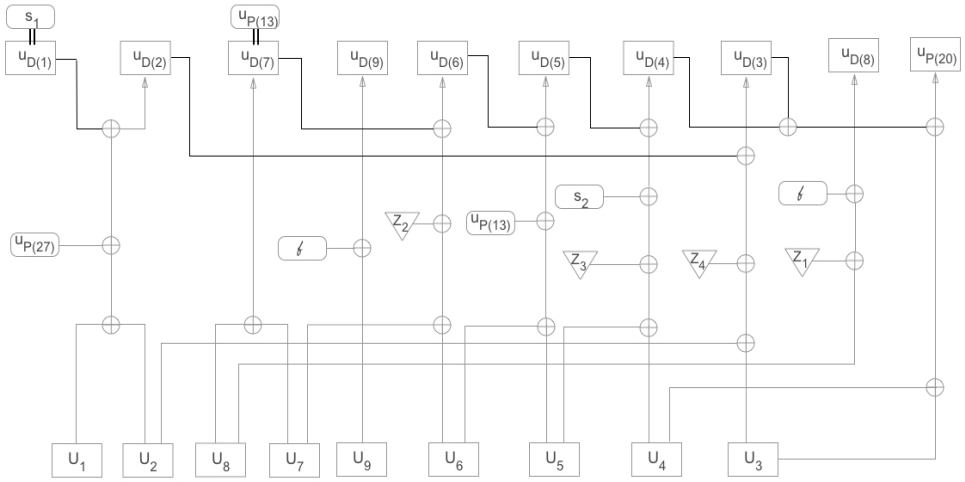


Fig. 3.14. A method for inverting ϕ which works for ANY key of type KT1.

`Z2` Now we are going to show that we know all the inputs of Z2 which are $u_{P(7-12)}$. which is not quite obvious. At this moment we have already obtained 4 bits of the 10 planned, and they remain only SIX bits of type u_{4*k} which remain unknown. These are $u_{D(3-6)}, u_{D(8)}$ and $u_{P(20)}$. Now $D(8) = P(6)$ cf. App. B.

In order to show that $Z_2(u_{P(7-12)})$ can be computed we need to show that: $\{D(3-6), P(6), P(20)\} \cap \{P(7-12)\} = \emptyset$. Moreover knowing that P is injective, we can exclude 6,20 and we just need to show that: $\{D(3-6)\} \cap \{P(7-12)\} = \emptyset$. Moreover, $\{D(3-6)\}$ only contains multiples of 4 and we have $P(7) = 5$ and $P(9) = 9$ due to W conditions in App. B. It remains to show that:

$$\{D(3-6)\} \cap \{P(8), P(10-12)\} = \emptyset$$

Now also following App. B, we have $D(3) \in \{P(1), P(2), P(4), P(5)\}$ and P is injective, so we can exclude $D(3)$ and it remains to show that:

$$\{D(4-6)\} \cap \{P(8), P(10-12)\} = \emptyset$$

which is exactly the last KT1 condition in Appendix B. This ends the proof that Z_2 is known.

`D6` Now we compute D6 using equation (6): $u_{D(6)} = U_6 \oplus U_7 \oplus u_{D(7)} \oplus Z_2(u_{P(7-12)})$.

`D5` Then after D6 we use equation (5) to compute $u_{D(5)}$ as:

$$u_{D(5)} = U_5 \oplus u_{D(6)} \oplus U_6 \oplus u_{P(13)}$$

`Z3` The inputs of Z_3 are $Z_3(u_{P(14-19)})$.
At this moment they remain only FOUR bits of type u_{4*k} which remain unknown. These are $u_{D(3-4)}, u_{D(8)}$ and $u_{P(20)}$. Discarding two $P(20), P(6)$ to injectivity of P as before, we need to show that:
It remains to show that:

$$\{D(3-4)\} \cap \{P(14-19)\} = \emptyset$$

We have $P(15) = 21$ and $P(18) = 25$ due to W conditions. and according to the penultimate condition in App. B, $D(4)$ can be excluded because it says precisely that $D(4) \notin \{P(14), P(16), P(17), P(19)\}$ and $P(15)$ and $P(18)$ were already excluded as not being multiples of 4. It remains to show that:

$$D(3) \notin \{P(14), P(16), P(17), P(19)\}$$

which is insured by the injectivity of P and condition pre-penultimate condition in App. B which says that $D(3) \in \{P(1), P(2), P(4), P(5)\}$.

`D4` Now that `D5` and `Z3` steps are done, we use equation (4) to compute $u_{D(4)}$ as:

$$u_{D(4)} = U_4 \oplus U_5 \oplus u_{D(5)} \oplus Z_3(u_{P(14-19)}) \oplus s_2$$

`Z4` The next step is to compute $Z_4(u_{P(21-26)})$. Can this intersect with any of the three remaining unknowns $u_{D(3)}, u_{D(8)}, u_{P(20)}$? The intersection is empty as $D(8) = P(6)$ and $D(3) \in \{P(1), P(2), P(4), P(5)\}$ and P injective makes that none of these can intersect with $P(21-26)$.

`D3` From Z_4 and $u_{D(2)}$ we compute $u_D(3)$ using equation (2). We obtain $u_{D(2)} = U_2 \oplus U_3 \oplus u_{D(3)} \oplus Z_4(u_{P(21-26)})$.

`Z1` This will enable the computation of $Z_1(s_2, u_{P(1-5)})$. Can this intersect with any of remaining unknowns $u_{D(8)}, u_{P(20)}$? Again not because $D(8) = P(6)$ and P injective.

`D8` From $Z1$ we can deduce $u_{D(8)}$ using equation (8) and we have: $u_{D(8)} = U_8 \oplus f \oplus Z_1(s_2, u_{P(1-5)})$.

`P20` The last unknown is determined using equation (2): $u_{P(20)} = u_{D(3)} \oplus u_{D(4)} \oplus U_3 \oplus U_4$.

This ends the proof that ϕ is bijective for any KT1 type key which is also a security proof against all sorts of attacks with "Vanishing Differentials" such as described in Section C.4, and also against all sort of correlation attacks such as described in Sections 18 and 20.

D A Study of KT2 Keys

D.1 Definition of KT2 Keys

$$D \text{ and } P \text{ are injective}$$
$$P(3) = 33, P(7) = 5, P(9) = 9, P(15) = 21, P(18) = 25, P(24) = 29$$
$$\text{Let } \ W = \{5, 9, 21, 25, 29, 33\}$$
$$\forall_{1 \geq i \geq 9} D(i) \notin W$$
$$\alpha \notin W$$
$$\text{Let } \ T = (\{0, 1, \ldots, 12\} \backslash W) \cap (\{P(1), P(2), \ldots, P(24)\} \cup \{D(4), D(5), \ldots, D(9)\} \cup \{\alpha\})$$
$$\text{Let } \ U = (\{13, \ldots, 36\} \backslash W) \cap (\{P(26), P(27)\} \cup \{D(1), D(2), D(3)\})$$
$$|T \backslash \{P(25)\}| + |U \backslash \{P(25)\}| \leq 12$$
$$A = \{D(1), D(2), D(3), D(4), D(5), D(6), D(7), D(8), D(9)\} \cup \{P(6), P(13), P(20), P(27)\}$$
$$A_1 = \{D(1), D(2)\} \cup \{P(27)\}$$
$$A_2 = \{D(3), D(4)\} \cup \{P(20)\}$$
$$A_3 = \{D(5), D(6)\} \cup \{P(13)\}$$
$$A_4 = \{D(7), D(8)\} \cup \{P(6)\}$$
$$\forall (i, j) \in \{1, \ldots, 27\} \times \{1, \ldots, 9\} : P_i \neq D_j$$
$$\exists j_1 \in \{1, \ldots, 7\} : D_{j_1} = 0$$
$$\{D(8), D(9)\} \subset \{4, 8, \ldots, 36\} \subset A$$
$$\forall (i, j) \in \overline{1, 27} \times \overline{1, 9} : P_i \neq D_j$$
$$\exists j_1 \in \overline{1, 7} : D_{j_1} = 0$$
$$\{D_8, D_9\} \subset \{4, 8, \ldots, 36\} \subset A$$
$$\exists (j_2, j_3) \in (\{j \in \overline{1, 4} | D_j ? \notin A_j\})^2 \wedge$$
$$\exists (j_4, j_5) \in (\overline{1, 4} \setminus \{j_1, 2j_2 - 1, 2j_2\}) \times (\overline{5, 8} \setminus \{j_1, 2j_2 - 1, 2j_2\}) \wedge$$
$$\exists j_6 \in \overline{1, 9} \setminus \{j_1, 2j_2 - 1, 2j_2, j_4, j_5\} :$$
$$j_2 \neq j_3 \wedge \{4j_4, 4j_5\} \subset A_{j_2} \wedge$$
$$A_{j_2} \cap (\overline{4j_1 - 3, 4j_1} \cup \overline{4j_6 - 3, 4j_6}) \neq \emptyset \wedge$$
$$\{8j_2 - 5, 8j_2\} \subset A_{j_3} \wedge A_{j_3} \cap (\overline{4j_1 - 3, 4j_1} \cup \overline{4j_6 - 3, 4j_6}) \neq \emptyset;$$
$$\{D(9)\} \setminus (\overline{33, 36} \cup \{0\}) \neq \emptyset$$
$$\{D(8), D(9), P(1), P(2), \ldots, P(5)\} \setminus (\overline{29, 32} \cup \{0\}) \neq \emptyset$$
$$\{D(7), D(8), P(1), P(2), \ldots, P(6)\} \setminus (\overline{25, 32} \cup \{0\}) \neq \emptyset$$
$$\{D(7), D(9), P(1), P(2), \ldots, P(6)\} \setminus (\overline{25, 28} \cup \overline{33, 36} \cup \{0\}) \neq \emptyset$$
$$\{D(6), D(7), D(8), D(9), P(1), P(2), \ldots, P(12)\} \setminus (\overline{21, 36} \cup \{0\}) \neq \emptyset$$
$$\{D(5), D(7), D(8), D(9), P(1), P(2), \ldots, P(13)\} \setminus (\overline{17, 20} \cup \overline{25, 36} \cup \{0\}) \neq \emptyset$$
$$\{D(7), D(8), D(9), P(1), P(2), \ldots, P(6)\} \setminus (\overline{25, 36} \cup \{0\}) \neq \emptyset$$
$$\{D(5), D(6), D(8), D(9), P(1), P(2), \ldots, P(13)\} \setminus (\overline{17, 24} \cup \overline{29, 36} \cup \{0\}) \neq \emptyset$$
$$\{D(5), D(6), D(7), D(9), P(1), P(2), \ldots, P(13)\} \setminus (\overline{17, 28} \cup \overline{33, 36} \cup \{0\}) \neq \emptyset$$
$$\{D(5), D(6), D(7), D(8), P(1), P(2), \ldots, P(13)\} \setminus (\overline{17, 32} \cup \{0\}) \neq \emptyset$$
$$\{D(5), D(6), D(7), D(8), D(9), P(1), P(2), \ldots, P(13)\} \setminus (\overline{17, 36} \cup \{0\}) \neq \emptyset$$
$$\{D(4), D(5), \ldots, D(9), P(1), P(2), \ldots, P(19)\} \setminus (\overline{13, 36} \cup \{0\}) \neq \emptyset$$
$$\{D(3), D(4), \ldots, D(9), P(1), P(2), \ldots, P(20)\} \setminus (\overline{9, 36} \cup \{0\}) \neq \emptyset$$

plus the "Matrix $rank = 9$ condition" M_9 defined in Section D.4 below.

D.2 Our Approach to KT2 Keys

The description of KT2 keys in [80] is excessively complex, cf. pages 59-60,114-115 and 117 in [80] or Appendix D.1 above. We are not sure why all these conditions have been imposed, possibly to obtain many very strong and exact mathematical properties and results such as Thm. 11.2.1 page 25 and many other such results which can be found in [80]. We don't believe that such exact results are actually needed for a cipher to be secure, and potentially they degrade the entropy of the long-term key down to relatively low levels, cf. Section 8.5.

D.3 A New Class of Keys KT2b

In this paper we define a new class of keys called KT2b which will contain only a tiny subset of the conditions of KT2. The selection was made as follows: we kept some particularly simple ones which occur for many other KT1 and KT2 keys, we also kept all those which are in some way "hard-coded" in Fig. 9.9 as this figure comes from the original specification of T-310 cipher in [80] and also those which avoid some particularly bizarre keys from [44] such as key 17 which has $P(25) = P(26)$. Then added few more conditions which are ONLY such as we judged necessary in order to be able to prove that ϕ will be bijective. We are not aware of any attack or security problem with any of the KT2b keys.

$(P, D, \alpha) \in KT2b \Leftrightarrow$ all the following hold:

$$\begin{cases} D \text{ and } P \text{ are injective} \\ P(3) = 33, P(7) = 5, P(9) = 9, P(15) = 21, P(18) = 25, P(24) = 29 \\ \text{Let } \ W = \{5, 9, 21, 25, 29, 33\} \\ \forall_{1 \geq i \geq 9} D(i) \notin W \\ \alpha \notin W \\ A = \{D(1-9),\} \cup \{P(6), P(13), P(20), P(27)\} \\ \forall (i,j) \in \{1, \ldots, 27\} \times \{1, \ldots, 9\} : P_i \neq D_j \\ \exists j_1 \in \{1, \ldots, 7\} : D_{j_1} = 0 \\ \{D(8), D(9)\} \subset \{4, 8, \ldots, 36\} \subset A \\ \text{the "Matrix } rank = 9 \text{ condition" } M_9 \text{ defined in Section D.4 below.} \end{cases}$$

Lemma D.3.1 (KT2$\Longrightarrow$ KT2b). Every key in class KT2 satisfies all the conditions of class KT2b which are simply a subset of conditions of KT2, cf. Section D.1 or page 60 in [80].

D.4 On M_9 Condition and Matrix B

Here we provide a statement of the "Matrix $rank = 9$ condition" which is defined as:

$$M_9 : \begin{cases} \text{The concrete values } D(i)/P(j) \text{ inside the formulas } \underline{\mathbf{D}}(s1, u_{I^1}) \oplus \underline{\mathbf{T}}(f, s2, \underline{\mathbf{P}}(u_{I^{1-4}})) \\ \text{which define the 9 "fresh" outputs } I^4 = \{1, 5, \ldots, 33\} \text{ of } \phi \text{ appear at such places} \\ \text{that all the 9 "fresh" outputs } I^4 \text{ of } \phi \text{ are sums of non-linear parts of type } Z(.), \\ \text{plus affine parts which involve various variables in } u_{I^{2-4}}, \text{ plus an invertible} \\ \text{linear transformation } B \text{ of rank 9 with the remaining 9 inputs of } I^1 = \{4, 8, \ldots, 36\}. \end{cases}$$

In addition we are going to show how to compute the coefficients of this matrix we will call B following[54] page 60 in [80]. We recall that we have in the general case the following relations which are a standard compact way to write ϕ in order.

$$\begin{aligned} u_0 &\stackrel{def}{=} s_1 \\ U_9 &= u_{D(9)} \oplus f \\ U_8 &= u_{D(8)} \oplus U_9 \oplus u_{D(9)} \oplus Z_1(s_2, u_{P(1-5)}) \\ U_7 &= u_{D(7)} \oplus U_8 \oplus u_{D(8)} \oplus u_{P(6)} \\ U_6 &= u_{D(6)} \oplus U_7 \oplus u_{D(7)} \oplus Z_2(u_{P(7-12)}) \\ U_5 &= u_{D(5)} \oplus U_6 \oplus u_{D(6)} \oplus u_{P(13)} \\ U_4 &= u_{D(4)} \oplus U_5 \oplus u_{D(5)} \oplus Z_3(u_{P(14-19)}) \oplus s_2 \\ U_3 &= u_{D(3)} \oplus U_4 \oplus u_{D(4)} \oplus u_{P(20)} \\ U_2 &= u_{D(2)} \oplus U_3 \oplus u_{D(3)} \oplus Z_4(u_{P(21-26)}) \\ U_1 &= u_{D(1)} \oplus U_2 \oplus u_{D(2)} \oplus u_{P(27)} \end{aligned}$$

We are now going to show that these equations have a certain property for any KT2b or/and any KT2 key such that some parts can be separated because they do not contain any numbers of type $4k$, and what remains will give the coefficients of B. More precisely we have:

Lemma D.4.1 (KT2b Separation Lemma). For every key which satisfies the conditions in class KT2b and ignoring the last M_9 condition, the 4 non-linear functions $Z()$ inside the round function ϕ depend only on variables of I^{2-4} which are not modified by ϕ, i.e. the $Z_{1-4}()$ do **not** depend on any of the input variables of type $4k$ in $I^1 \cup \{0\}$.

Proof: We recall that for every KT2b key we have:

$$\{4, 8, \ldots, 32, 36\} \subset \{D(1-9); P(6), P(13), P(20), P(27)\}$$

and all outputs of D and P are disjoint by definitions in KT2b. This implies that the inputs of 4 non-linear functions $Z()$ cannot contain any of the $\{4, 8, \ldots, 32, 36\}$. Moreover in KT2b one of $D(1-7)$ will be 0 (which is where $u_{D(i)}$ is replaced by $s1$ in the definition of ϕ). Accordingly, $u_0 = s_1$ cannot be any of the inputs of the $Z()$ either which are all either of the form $u_{P(i)}$ or s_2. This ends the proof for KT2b, and also for KT2, as KT2$\Longrightarrow$ KT2b, cf. Lemma D.3.1

D.5 Computation of Matrix B

In order to write the matrix B for any KT2b or/and any KT2 key we just need to discard all the $Z()$ and all the numbers not in $\{4, 8, \ldots, 32, 36\}$ in and we will a obtain a square 9×9 matrix $B = (b_{ij})$.

[54] Our matrix B will be an equivalent obtained by a linear transformation on rows (which preserves the rank and invertibility) of the matrix B as defined in page 60 in [80], which matrix it would be more complex to write due to the fact that the T_i are defined a sort of recursive straight-line program cf. Section 9 and decided to keep it that way which is very short and avoids very long summations.

We have then (the arithmetic is done mode 2):

$$\begin{pmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \\ U_6 \\ U_7 \\ U_8 \\ U_9 \end{pmatrix} = B \cdot \begin{pmatrix} u_4 \\ u_8 \\ u_{12} \\ u_{16} \\ u_{20} \\ u_{24} \\ u_{28} \\ u_{32} \\ u_{36} \end{pmatrix} + C \qquad \text{where } C \stackrel{def}{=} \begin{pmatrix} f \\ Z_1(s_2, u_{P(1-5)}) \\ u_{P(6)} \oplus \dots \\ Z_2(u_{P(7-12)}) \oplus \dots \\ u_{P(13)} \oplus \dots \\ Z_3(u_{P(14-19)}) \oplus s_2 \oplus \dots \\ u_{P(20)} \oplus \dots \\ Z_4(u_{P(21-26)}) \oplus \dots \\ u_{P(27)} \oplus \dots \end{pmatrix}$$

Here $\oplus \dots$ denotes some additional terms and will not occur in the first two lines, they will only occur if some of the $u_{D()}$ in the equations in Section D.4 above have terms which are not in $\{4, 8, \dots, 36\}$, in which case they need to be added to C, with a replacement of u_0 by s_1 in one case.

To make it more concrete, in Section E.4 page 76 we show a concrete (and a bit special) example of how this matrix looks like for one particular key.

D.6 On Invertibility of KT2 Keys

We have the following result:

Theorem D.6.1 (KT2 and KT2b Invertibility Theorem). For every key in class KT2b and therefore also for every KT2 key and for every 3 bits $s1, s2, f$ the round function ϕ is bijective and given the 36 outputs the 9 input bits of the form $4k$ can be computed by solving a linear system of rank 9.

Proof: Again due to KT2 Separation Lemma D.4.1, we know all the values in C and B is assumed to be invertible. Therefore we can do the inversion simply as:

$$\begin{pmatrix} u_4 \\ u_8 \\ u_{12} \\ u_{16} \\ u_{20} \\ u_{24} \\ u_{28} \\ u_{32} \\ u_{36} \end{pmatrix} = B^{-1} \cdot \begin{pmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \\ U_6 \\ U_7 \\ U_8 \\ U_9 \end{pmatrix} + B^{-1} \cdot C \qquad \text{where } C \stackrel{def}{=} \begin{pmatrix} f \\ Z_1(s_2, u_{P(1-5)}) \\ u_{P(6)} \oplus \dots \\ Z_2(u_{P(7-12)}) \oplus \dots \\ u_{P(13)} \oplus \dots \\ Z_3(u_{P(14-19)}) \oplus s_2 \oplus \dots \\ u_{P(20)} \oplus \dots \\ Z_4(u_{P(21-26)}) \oplus \dots \\ u_{P(27)} \oplus \dots \end{pmatrix}$$

Remark: K2 vs. KT1: In KT1 we had a very different situation, many inputs to $Z()$ were not initially known. Some concrete examples of this are bits 24,32,8,4,16 in Fig. 3.12 and our proof that these can be determined in the general case was far from being trivial and required to use many specific conditions mandated for KT1 keys, cf. Thm. C.10.1. Here for KT2 the proof is substantially simpler overall and uses extremely few of the conditions mandated for KT2.

E On Keys Similar to KT2

E.1 On Security of KT2 Keys and Chosen Long-Term Key Attacks

The known sources [80, 44] report only one KT2 key which is number 15 from 1979. In order to better understand the properties of these keys we are going to show some special keys.

E.2 Some Examples Of Abnormal Keys

While trying to mathematically prove that KT2 are bijective, cf. Thm D.6.1 above, we have generated several examples of keys which satisfy **all** conditions of KT2 except maybe the "Matrix $rank = 9$ condition" M_9 of Section D.4.

Table 10. Examples of keys which would be of type KT2 except for the matrix rank condition M_9.

key nb	D	P	rank of B
206	4,0,32,2,35,17,12,20,24	15,13,33,18,34,8,5,6,9,30,22,14,16,3,21,31,7,25,26,28,27,11,23,29,19,1,36	6
207	0,24,20,8,16,2,11,32,4	7,6,33,26,17,13,5,19,9,10,27,18,12,30,21,15,34,25,23,36,31,14,22,29,3,1,28	7
407	0,24,20,8,16,2,11,32,4	17,7,33,6,10,13,5,27,9,26,22,18,12,30,21,15,34,25,23,36,31,14,19,29,3,1,28	7
208	17,0,2,32,35,4,12,20,24	13,15,33,10,18,8,5,30,9,6,3,14,16,22,21,31,7,25,26,28,27,11,23,29,19,1,36	8
15	0,4,17,12,35,32,2,24,20	15,13,33,34,6,8,5,3,9,18,14,22,28,30,21,31,7,25,26,16,27,11,23,29,19,1,36	9

We call this sort of keys "Rank-Deficient" KT2 keys, cf. Definition 19.2.1 page 42. More such keys which satisfy less conditions can be found in Table 6 page 38.

E.3 The Anomalous Long-Term Key 207

We study the key 207 in more detail which is as follows:

```
D=0,24,20,8,16,2,11,32,4 P=7,6,33,
26,17,13,5,19,9,10,27,18,12,30,21,15,34,25,23,36,31,14,22,29,3,1,28
```

this key 207 has some[55] interesting properties. We recall that this key satisfies all the conditions for KT2 except the very last "Matrix $rank = 9$ condition" M_9. For this key the round function ϕ is **not bijective** (!).

In what follows we are going to show what exactly is the problem with this long term key. We recall that KT2 mandates the matrix B to be invertible. This is precisely is the only condition violated in our key 207, which will be easily seen if we re-write our equations in such a way which makes this matrix B appear explicitly.

E.4 Example of Computation of Matrix B for Key 207

We recall our set of multivariate equations cf. Section 9.2 or Section C.8. Let $z_i = u_{m+1,i}$ in order to distinguish the inputs $u_{m,i}$ and the outputs $u_{m+1,i}$ denoted simply by u_i in our compact notation.

[55] One particularity of this key is that it has $\underline{\mathbf{D}} : \mathbb{F}_2^{9+2} \rightarrow \mathbb{F}_2^9$, cf. Section 5.5.

$$\begin{aligned}
z_1 \oplus z_5 &= u_{24} \oplus & s_1 \oplus u_{P(27)} \\
z_5 \oplus z_9 &= u_{24} \oplus u_{20} \oplus & Z_4(u_{P(21-26)}) \\
z_9 \oplus z_{13} &= u_{36} \oplus u_8 \oplus u_{20} \oplus & 0 \\
z_{13} \oplus z_{17} &= u_8 \oplus u_{16} \oplus & s_2 \oplus Z_3(u_{P(14-19)}) \\
z_{17} \oplus z_{21} &= u_{16} \oplus u_{12} \oplus & u_{D(6)} \\
z_{21} \oplus z_{25} &= & u_{D(6)} \oplus u_{D(7)} \oplus Z_2(u_{P(7-12)}) \\
z_{25} \oplus z_{29} &= u_{32} \oplus & u_{D(7)} \oplus u_{P(6)} \\
z_{33} \oplus z_{29} &= u_{32} \oplus u_4 \oplus & Z_1(s_2, u_{P(1-5)}) \\
z_{33} &= & u_{D(9)} \oplus f
\end{aligned}$$

Here it is trivial to observe that the rank of B is at most 7: we have two empty lines in B.

E.5 A Collision For Key 207

We present one example of a collision with this key where most bits are at 0, and only very few bits are at 1, which makes this example easy to study and easy to verify. We have found the following collision:

$$U^{(b)} = \phi(0,0,0;U^{(a)}) = \phi(0,0,0;U^{(a')})$$

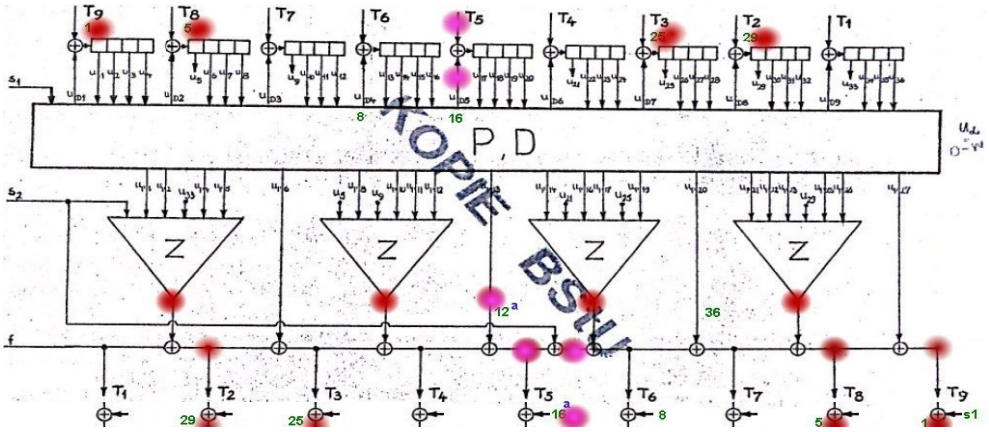
which is also shown in Fig. 5.15 below. Here we define $U^{(a)}$ as all bits being 0 except two $u_{D(5)} = u_{16} = 1$ and $u_{P(13)} = u_{12} = 1$. Then let $U^{(a')}$ is such that all bits are 0 except $u_{D(4)} = u_8 = 1$ and $u_{P(20)} = u_{36} = 1$. Finally let $U^{(b)} = \phi(0,0,0;U^{(a)})$. Here all bits are at 0 except four which are

$$\begin{aligned}
z_1 &= 1 \\
z_5 &= 1 \\
z_{25} &= 1 \\
z_{29} &= 1
\end{aligned}$$

One way to see how this collision can occur, is to re-write the 9 equations of Section E.4 in such a way that bits which will be at 1 for EITHER a or a' case are on the left hand side, and the bits which are zero in both cases on the right hand side. We have $1 = Z(0,0,0,0,0,0)$ in all four instances of our function $Z()$.

$$\begin{aligned}
z_1 \oplus z_5 &= s_1 \oplus u_{D(2)} \oplus u_{P(27)} \\
z_5 \oplus Z_4(u_{P(21-26)}) &= u_{D(2)} \oplus z_9 \oplus u_{D(3)} \\
u_{36} \oplus u_8 &= z_9 \oplus u_{D(3)} \oplus z_{13} \\
u_8 \oplus u_{16} \oplus Z_3(u_{P(14-19)}) &= z_{13} \oplus z_{17} \oplus s_2 \\
u_{16} \oplus u_{12} &= z_{17} \oplus z_{21} \oplus u_{D(6)} \\
z_{25} \oplus Z_2(u_{P(7-12)}) &= z_{21} \oplus u_{D(6)} \oplus u_{D(7)} \\
z_{25} \oplus z_{29} &= u_{D(7)} \oplus u_{D(8)} \oplus u_{P(6)} \\
z_{29} \oplus Z_1(s_2, u_{P(1-5)}) &= z_{33} \oplus u_{D(9)} \oplus u_{D(8)} \\
0 &= z_{33} \oplus u_{D(9)} \oplus f
\end{aligned}$$

In this form we see immediately that the both $U^{(a)}$ and $U^{(a')}$ have an even number of active bits on the left had side and therefore our collision is correct! We can also view this collision on the figure below.



has the same output $(U_1, U_2, \ldots, U_9) = (z_{33}, z_{29}, \ldots, z_5, z_1) = (0, 1, 1, 0, 0, 0, 0, 1, 1)$ as:

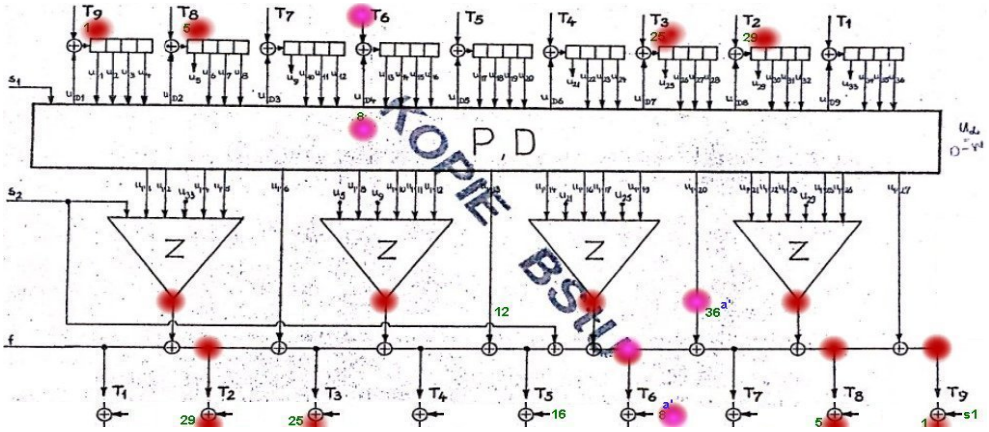


Fig. 5.15. A collision for ϕ with key 207: active wires at 1 are marked with color dots.

F Further Non-Standard Keys

F.1 An Example of Long-Term Key Which is Neither KT1 Nor KT2

In this section we demonstrate that the set of 10 conditions for the long-term keys specified in 256 in [76] is not quite correct and not compliant with the original document [80]. More precisely the author of [76] has forgotten to transcribe some additional conditions of Section 4.1. in Appendix 1 of [80] such as $P(3) = 33$ and few other conditions. In order to show that the spec of [76] is indeed incomplete we have also created (by trial and error) our own example of long-term key which satisfies 100 % of the criteria of page 256 in [76].

```
D=0,24,32,4,8,28,16,20,12 P=12,32,8,
14,4,20,21,26,30,24,17,25,16,1,27,23,18,5,13,36,2,34,15,28,10,6,3
```

However it is easy to see that $P(3) = 8 \neq 33$. This long-term key does not belong to class KT1. Interestingly, it appears that for every key/IV bits we obtain a bijection ϕ for a round function.

F.2 The Special Key 16 and SKS Cipher

The following non-standard key is described in [44] as a key number 16 for a special version of T-310 cipher machine called T-310/51 instead of the usual T-310/50. We ignore what the exact difference between these machines might be but in [44] we read that this key 16 is approved for both T-310/50 for some sort of testing and was also used in 1984 for testing of T-310/51.

```
//Der Langzeitschl{\"u}ssel 16: (1979)
D=0,35,19,23,27,11,3,15,31 P=14,19,33,
18,23,15,5,6,9,2,34,1,30,11,21,3,22,25,17,7,32,10,27,29,26,35,13
```

Moreover in page 42 of [80] we read that this (apparently the same key 16) is some sort of either mathematical or an exact functional equivalent of a key for an earlier encryption machine called SKS[56].

For this LZS number 16, the state has in fact only 27 distinct active bits instead of 36, and the other bits such as $4, 8$ and many other are simply not used, see Table 1 page 26. At the same time it still has the basic 6 properties [57] regarding the set W of Section B such as $P(3) = 33$. This leads to the following situation which we depict in Fig. 6.16 below.

[56] SKS is also mentioned in other T-310 sources such as [44].

[57] These can be traced to Section 4.1. in Appendix 1 of [80] and they hold for all of KT1 keys, all of KT2 keys, and also for this key 16.

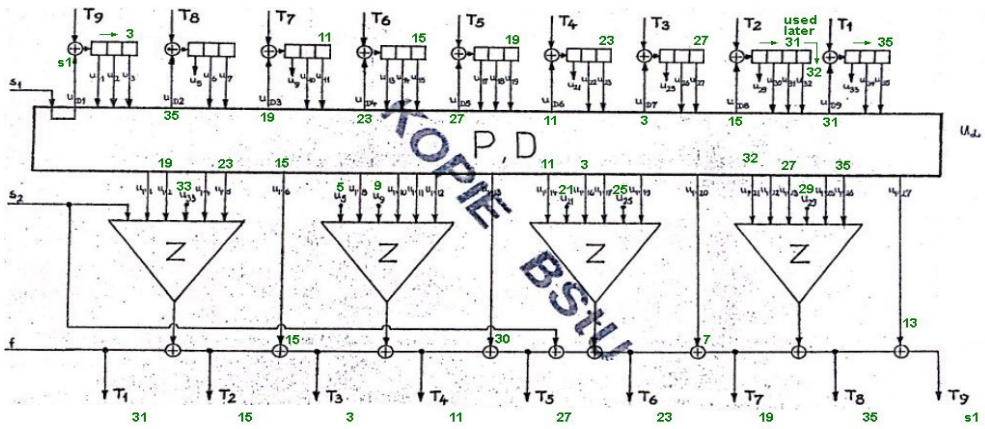


Fig. 6.16. T-310 round function based on page 119 in [80] with modifications due to the fact that most state bits of type $4k$ are no longer used except D(1)=0 and P(21)=32. We mark with green numbers bits 32, or those in W, and those XORed to the outputs $T1 - T9$ respectively, cf. Section 7.5.

Remark. The bit 32 in T-310 implementation of this permutation is simply there to represent the bit 31 which is used in a later cycle clock.

F.3 Special Key 16, SKS Cipher and KT0 Key Class

It is clear from [80] that SKS cipher is older than T-310 and clearly also substantially simpler. For this reason we will informally call this key a KT0 type.

On page 48 in [80] we read that:

> The function $\Phi_T : \{0,1\}^{36} \to \{0,1\}^{36}$ represents a generalization of the mapping $\Phi_S : \{0,1\}^{27} \to \{0,1\}^{27}$ in the sense that for each pair $(P,R) \in SG(1-27) \times SG(1-9)$ a pair (P',D) exists that satisfies the conditions of the definition of the encryption algorithm T-310, while the corresponding function Φ_T on the 27 components $u_1, u_2, u_3, u_5, u_6, \ldots u_{31}, u_{33}, u_{34}, u_{35}$ of the 36-digit vector $U = (u_1, u_2, \ldots u_{36})$ realizes the mapping Φ_S.

Here Φ_S denotes the round function of the SKS V/1 encryption algorithm, cf page 47 in [80], which we conjecture to be a bijection on 27 bits.

G An Advanced Birthday Paradox Sliding Key Recovery Attack with $d = 1$

The question is now HOW to break this block cipher knowing that $120s$ rounds is a large number and just one condition on 36 bits is not sufficient to uniquely determine a key on 230 bits. The answer is that we need to combine several variants of the above attack and apply Thm. 23.0.2 several times.

Here is one basic way to do it:

1. We will use the case $d = 1$ and several different $s \geq 18$ such that $d = 1$.
2. The attacker will try $2^{5.5}$ different s values, of the form $s = 18 + 127u$ with any $1 \leq u \leq 2^{5.5}$. For each s and for all possible IV we apply Thm. 23.0.2.
3. The attacker test all possible $2^{61} - 1$ IVs, to discover some $2^{61-36} = 2^{25}$ "good" IVs where he has $u_{120s} = u_0 = 0xC5A13E396$.
4. We expect that this set of 2^{25} "good" IVs is random, and different for each s.
5. The attacker will store many of these "good" IVs in a hash table, he stops if he finds a collision on 61 bits: IV,IV' are such that IV and IV' are shifted by 120 rounds, exactly (NOT a multiple of 120 rounds). To achieve this, we store in our hash table both IV and the IV shifted by 120 steps forward.
6. Memory required is about 2^{61} bits.
7. By birthday paradox, we need just about $2^{30.5}$ cases.
8. We see that if only we try $2^{5.5}$ values s and all 2^{61} IVs, some $2^{30.5}$ of which will work, we should obtain a desired collision.
9. The data complexity is about $4 \cdot 2^{61+5.5} \approx 2^{67.5}$ chosen IV chosen ciphertext decryption queries with which are $2^{5.5} \cdot 127 \cdot 120 \approx 2^{19.5}$ bytes each in length.
10. The time complexity is roughly about $4 \cdot 2^{61+5.5+7+7} \approx 2^{81}$ CPU clocks.

For example with large probability the attacker obtains the following type of collision: $u_{120s+0} = u_0 = 0xC5A13E396$ for one IV, and $u'_{120s'+0} = u'_0 = 0xC5A13E396$ for IV' shifted by 120 steps exactly which becomes "accidentally" equal to IV by the birthday paradox.

We obtain a situation where $u_{120} = u_0 = 0xC5A13E396$ for the the first IV. We have obtained a P/C pair for 120 rounds exactly.

With roughly $\sqrt{8}2^{30.5} \approx 2^{33}$ times more attempts, we can obtain more than one such colliding pairs for example 8 pairs. We expect that approximately 8 pairs will be needed in order to be able to recover the key by a SAT solver as in Section 15. As long as this step takes less[58] than $\sqrt{8} \cdot 2^{81} \approx 2^{82.5}$ CPU clocks, this does NOT change the complexity of our attack.

Overall we see that we can recover the 230-bit key of T-310 with about $\sqrt{8} \cdot 2^{67.5} \approx 2^{73}$ chosen IV chosen ciphertext decryption queries with messages of less than 2^{20} characters each. The time required is about 2^{83} CPU clocks and memory required is about 2^{61} bits.

[58] For example, in Table 1, Section 9, page 25 in [34], the time complexity is below 2^{83} starting from 6 PC pairs and decreases with more P/C pairs. We expect a similar result here and arguably 120 rounds of T-310 are the equivalent of 8 rounds of GOST in terms of complexity and key usage.

H Stream Ciphers, LFSRs and T-310

T-310 is also a stream cipher or a block cipher used in a mode which effectively transforms a block cipher into a stream cipher. T-310 also has components which are typically found only in stream ciphers, not in typical block ciphers. It incorporates an LFSR in the expansion of the IV which is the part which makes this block cipher a-periodic which can be compared to other block ciphers where regular periodic structure is a source of numerous attacks, e.g. GOST [34]. It is also possible to view the matrix operation used in T-310 encryption process as another (much smaller) LFSR which is clocked a variable number of steps.

Since Eurocrypt 2003 [23, 24, 31], many families of LFSR-based stream ciphers can be efficiently broken. Unhappily, compared to most traditional LFSR-based stream ciphers the LFSRs are used in T310 in a very different way. One is used to produce a-periodic sequence which is public (derived from the IV), and another as a secondary re-encryption process for data already potentially strongly encrypted. Attacks on stream ciphers have been developed initially on ciphers with "Linear Feedback" [23, 24] which comes from LFSRs. These attacks were later improved/enhanced to tolerate a proportion of arbitrary non-linear components, cf. [24] and Fig. 8.17 below. We reproduce this picture here to show that it DOES apply to T-310.

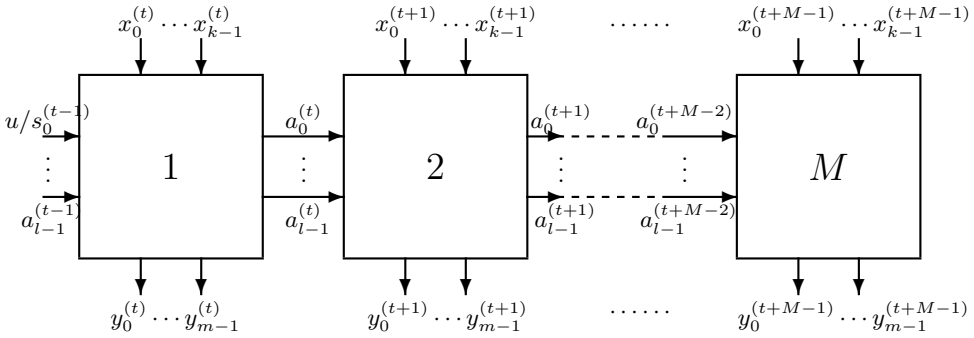


Fig. 8.17. M successive applications of a combiner with k inputs, m outputs and l bits of memory, a general setting in stream cipher cryptanalysis, cf. Fig. 2. in [24]

There are two major ways to apply it to T-310. In both cases we assume that the NON-LINEAR part of the state denoted $a_0^{(t-1)}, \ldots, a_{l-1}^{(t-1)}$ in Fig. 8.17 and in [24], is now going to be a combination of the secret key and the 36-bit block state. Therefore the internal state will now be 266 bits: 230 key bits bits $s_{j,1-2}$ which will repeat every and 36 state bits $u_{j,0-36}$. Now we can:

1. Either consider that each box as on Fig. 8.17 contains one iteration of ϕ and the key is encoded on 240 bits and both halves are rotated by one position modulo 210 when they exit the box to enter the next box, then we have $(k, m, l) = (1, \frac{10}{127*13}, 240 + 36)$ where $m = \frac{10}{127*13}$ means that we only use a small fraction of what is output.

2. Or consider that each box as on Fig. 8.17 contains 120iterations of ϕ and the key is on 230 bits and output identical to enter the next box, then we have $(k, m, l) = (120, \frac{120*10}{127*13}, 230 + 36)$.

In both cases the inputs come from one or several LFSRs, a linear component, exactly as in [24], which means that we can potentially apply the methods and ideas of [24]. This however will encounter very substantial difficulties: the main idea in [24] is that the more bits are output from such sub-system connected to one or several LFSRs, the easier it becomes to break by an algebraic attack, and attacks are particularly strong when m is large. Here we have $m \leq 1$ which makes it very difficult to hope that we can find I/O properties such as in [24] which eliminate all the 266 bits which are hard to predict for the attacker.

Consequently, it is clear than T-310 is a lot more robust than any stream cipher considered in [24] or it has a non-linear part of the state updated at each clock which is particularly important. In T-310 even the primary "internal" sequence of bits $u_{127j,\alpha}$ to which the attacker has no direct access, is produced by a highly non-linear component, which is also bijective, making it a block cipher. Not by a relatively small variation of an LFSR-based stream cipher. Overall we see little hope that any of the classical attacks on stream ciphers could be applied to T-310.

H.1 More About LFSR-based Stream Ciphers and T-310

The question is about designing an LFSR-based stream cipher with a potentially an extremely robust combiner/filter component, cf. Fig. 8.17 above and [24]. Overall the analysis of [24] can potentially be applied, at least in theory. The main point of [24], cf. Thm. 5.1 of page 7 in eprint version is that such a combiner/filter system could have a sort of "secondary key" in the form of I/O polynomial equations which the input bits [public in T-310 also for u_0] and the output bits which are those used for encryption in T-310, and which ELIMINATES totally all the internal variables of the combiner/filter system, which here would be all the intermediate states u_t of the T-310 block cipher and which are denoted by a_i on Fig. 2. of [24]. This is a strong result and could be applicable to T-310, it basically means that there exist a certain system of I/O equations which could be seen as a "secondary key" for K-310, and a recovery of these equations could be an option for the attacker.

Depending on the degree, size and sparsity of such equations this recovery might be possible, and such "secondary key" could potentially be used to decrypt communications routinely, under a number of technical conditions such as for example if the system of equations would extend with additional equations which allow to determine other unknown bits directly.

I A Reference Implementation of T-310

We provide a simple reference implementation of T-310 in C language. It is free to use and modify provided that the derived works contain a reference or a link to the present paper.

```
//9=>1
inline int T310ZFunction(int e1,int e2,int e3,int e4,int e5,int e6)
{
int sum=
1 + e1 + e5 + e6
+ e1*e4 + e2*e3 + e2*e5 + e4*e5 + e5*e6
+ e1*e3*e4 + e1*e3*e6 + e1*e4*e5 + e2*e3*e6 + e2*e4*e6 + e3*e5*e6
+ e1*e2*e3*e4 + e1*e2*e3*e5 + e1*e2*e5*e6 + e2*e3*e4*e6 + e1*e2*e3*e4*e5
+ e1*e3*e4*e5*e6;
return sum&1;//mod 2
};

//29=>9
inline void T310TFunction(
int &t1,int &t2,int &t3,int &t4,int &t5,int &t6,int &t7,int &t8,int &t9,
int e00,int e01,int e02,int e03,int e04,int e05,int e06,int e07,int e08,int e09,
int e10,int e11,int e12,int e13,int e14,int e15,int e16,int e17,int e18,int e19,
int e20,int e21,int e22,int e23,int e24,int e25,int e26,int e27,int e28)
{
t1=e00;
t2=t1+T310ZFunction(e01,e02,e03,e04,e05,e06);t2&=1;
t3=t2+e07;t3&=1;//mod2
t4=t3+T310ZFunction(e08,e09,e10,e11,e12,e13);t4&=1;
t5=t4+e14;t5&=1;//mod2
t6=t5 +e01+ T310ZFunction(e15,e16,e17,e18,e19,e20);t6&=1;
t7=t6+e21;t7&=1;//mod2
t8=t7+T310ZFunction(e22,e23,e24,e25,e26,e27);t8&=1;
t9=t8+e28;t9&=1;
};

//hard coded part of D, NOT the only possibility
int j[9]={-1,3,7,2,6,5,8,4,9};
//hard coded part of P, NOT the only possibility
int p[28]={-1,
8,4,33,16,31,20,5,35,9,3,19,18,12,7,21,13,23,25,28,36,24,15,26,29,27,32,11};

//input x=1..9 output=0..36
int D(int x){ if(x==1) return 0; else { for(int k=1;k<=8;k++){
if(x==j[k])
{
```

```
if(k==1) return 4; else return 4*j[k-1];//4*j_{k-1}
};}; }; printf("D(%d) undefined",x); return -9999;//should never happen
};

//input x=1..27 output=1..36
int P(int x){ return p[x]; };

//(c) Nicolas T. Courtois January 2017, and based on Klaus Schmeh:
//The East German Encryption Machine T-310 and the Algorithm It Used,
//In Cryptologia, 30: 3, pp. 251  257, 2006.
void T310BlockPhiEncryptOneRound(
int s1,int s2,int f,//extra inputs = key/IV
int o[37],//outputs [1..36]
int i[37]//inputs [1..36]: v[0]=s1 etc...
)
{
int v[37]={0};//internal 37 inputs: v[0]=s1 and last/proper 36
v[0]=s1;//one extra input
for(int k=1;k<=36;k++)
v[k]=i[k-1+1];
int j=0;
int t[10]={-1,0};//used 1..9, outputs of T
T310TFunction(
t[1],t[2],t[3],t[4],t[5],t[6],t[7],t[8],t[9],
f,s2,
v[P(1)],v[P(2)],v[P(3)],v[P(4)],v[P(5)],v[P(6)],v[P(7)],v[P(8)],v[P(9)],v[P(10)],
v[P(11)],v[P(12)],v[P(13)],v[P(14)],v[P(15)],v[P(16)],v[P(17)],v[P(18)],v[P(19)],
        v[P(20)],v[P(21)],v[P(22)],v[P(23)],v[P(24)],v[P(25)],v[P(26)],v[P(27)]
 );
for(j=1;j<=9;j++)
o[4*j-3]=( v[D(j)]+t[10-j] ) &1;
for(j=1;j<=9;j++)
o[4*j-2]=v[4*j-3];//starts at input v[1]
for(j=1;j<=9;j++)
o[4*j-1]=v[4*j-2];
for(j=1;j<=9;j++)
o[4*j-0]=v[4*j-1];//up to output o[36]
};
```

J Some Test Vectors For T-310

```
key=1234567890abcdef1234567890abcdef1234567890abcdef1234567890ab
IV16=1acdef1234567890
IV2=11010110011011110111100010010001101000101011001111000100100000
Input=00011 Output=00100
```

K Short Documentation for Our Software Algebraic Attack Tool

Our student Om Bhallamudi (with help from Maria-Bristena Oprisanu and Varnavas Papaioannou) has developed an open source software solution [2] for implementing software algebraic attacks on T-310 which we use in Section 15.1.

This software is a combination of several programs and we advise to run it under any version of Windows 64-bit with Python 2.7 x64 installed. Certain files necessary to work should be in the current directory and are the following:

```
codegen.py
helpers.py
argon.py
config.py
ax64.exe
minisat2.exe
cryptominisat-2.9.6-win64.exe
vcomp90.dll
```

These files can be obtained from [17] for example a direct download link would be `http://www.nicolascourtois.com/software/codegen_last.zip` or `http://www.nicolascourtois.com/software/ex64.exe` or `http://www.nicolascourtois.com/software/*.py`.

The basic command line reference is:

```
python codegen.py Nr /fix115 /insX /xl /sat /T310set26
Nr = number of rounds
X=number of instances, 8 recommended
/T310set26 uses LZS-26
/fix115 could be replaced by /fix1/2 which will fix half of the 230 key bits
```

L Short Documentation For Our DC Tool

Our Differential Cryptanalysis (DC) tool [75] was written by our student Matteo Scarlata and was used in Section 12.3. Here we provide a basic documentation. The download link which should also contain a more up-to-date documentation is: `https://gitlab.com/mtscr/T-310`

```
In the default mode, this tool looks for differential properties from
any delta-in to delta-outs of Hamming weight < 3. Compile with:
g++ -pedantic -Wall -std=c++11 -lcrypto -O3 -o t310-diff t310-lib.cpp
t310-diff.cpp
then run a quick demonstration:
./t310-diff -X | python2.7 counter.py

In order to make a longer computation and analyze the results later run:
./t310-diff | tee -a results.log
python2.7 counter.py results.log

### t310-diff
Basic options:
./t310-diff -r <number-of-rounds> -t <minimum-number-of-samples> -k <IV-key-bits>
-p <min-probability-to-show> -e <stop-after-x-computations>

A bigger <minimum-number-of-samples> will increase the precison of the results.
By default, the tool will iterate over all possible bitmasks (delta-in values),
this will take a long time even for few rounds.

Advanced options:
-P n: choose delta-in in the set of all the permutations of n "1" (and 36-n "0")
-X : choose delta-in of Hamming weight 1
-B <36bits-bitset> : select the active bits of the bitmask (default: all 1)
-S <36bits-bitmask> : start the computation from bitmask+1
-i : look for delta-outs of Hamming-weight >32 (experimental)
-H <delta-in-hw>: collect statistics about the distribution of delta-out
Hamming weights for a fixed delta-in Hamming weight

### counter.py
Run:
python2.7 counter.py -h
to show the help.

e.g. run:
python2.7 counter.py results1.log [results2.log ... ] -j -r <number-of-rounds>
to join the collected results for a low number of rounds in a result
for a higher number of rounds.
```