Cryptographic Security Analysis of T-310*

Nicolas T. Courtois[1], Klaus Schmeh[3], Jörg Drobick[5], Jacques Patarin[2], Maria-Bristena Oprisanu[1], Matteo Scarlata[1,4], Om Bhallamudi[1]

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

Abstract. T-310 is an important Cold War cipher [96]. 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 has 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.

***** This is our "master paper" on T-310. It can be seen as an extended version of several papers which appear in Cryptologia in 2017 and 2018. This eprint paper is our extended monography paper which shows how different questions and different attacks are related and connected to each other. It contains a substantial amount of additional research and technical details.

Key Words: Cold War, block ciphers, T-310, SKS V/1, unbalanced compressing Feistel ciphers, algebraic cryptanalysis, ElimLin, SAT Solvers, Differential Cryptanalysis, Linear Cryptanalysis, correlation attacks, slide attacks, self-similarity attacks, ciphertext-only attacks.

Acknowledgments. We thank Bernd Lippmann, Jens Raeder, Bernhard Esslinger, Marek Grajek, Philippe Guillot, Nathan Keller, Jean-Jacques Quisquater, Angela Sasse, Mate Soos, Kristina Zatylna and Bingsheng Zhang for their comments and suggestions. Some of this work was done by UCL students doing project work for GA18 Cryptanalysis course taught at University College London in 2014-2018. Students who participated are: Om Bhallamudi, Simon Boehm, Kwok Cheng, Killian Davitt, Maxine Emuobosa, Mario D'Onghia, Mark Daniels, Lizhou Feng, Istvan Hoffer, Gaixin Hong, Marios Georgiou, Tereza Loffelmannova, Alexios Nikos, Maria-Bristena Oprisanu, Nikolai Rozanov, Matteo Scarlata, Qin Tang, Varnavas Papaioannou, Rei Valera, Moyu Wang, Dongni Zhang.

Table of Contents

1 Basic Facts and History of T-310

T-310/50 is an important historical cipher designed and built by mathematicians and 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 [96]. 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 [58, 96].

1.1 Chronology on T-310

Based on [61, 11, 56] we present here a short chronology on the development of T-310 cipher 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.
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 the iteration of a relatively complex block cipher. In this section we provide a first quick and informal description of the T-310 cipher.

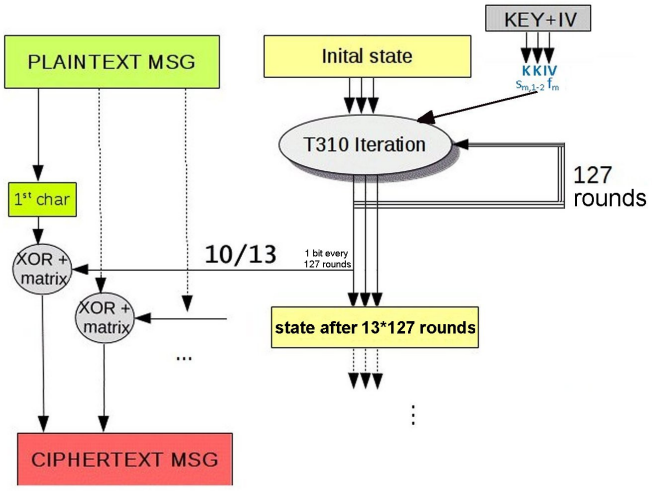


Fig. 2.1. T-310 Encryption Process.

T-310 contains an LFSR. However we cannot really hope to apply attacks on LFSR-based stream ciphers [30, 37], 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 later call "the T-310 block cipher". The block size is only 36 bits, the secret key has 240 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's 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 [15, 12, 40, 44, 38] should and will to some extent apply to T-310. For example, there exist techniques which break any cipher, if not too complex, cf. [15, 36, 37, 24, 99, 22]. Unhappily T-310 is quite complex.

[1] According to page 17 in [100], the round clocking frequency is 76.8 kHz which gives an encryption speed of about 46.5 characters per second.

2.1 A First Look At the T-310 Block cipher Internals

The "T-310 block cipher" is a keyed permutation on 36 bits with has potentially unlimited number of rounds [depending how much data in encrypted]. Each round depends also on 3 extra bits. Two come from the secret key of 240 bits, and 1 bit comes from the IV. In Fig. 2.2 below we show a first glimpse of how it looks like in the general case [the dotted connections are optional and do not exist for most historical keys].

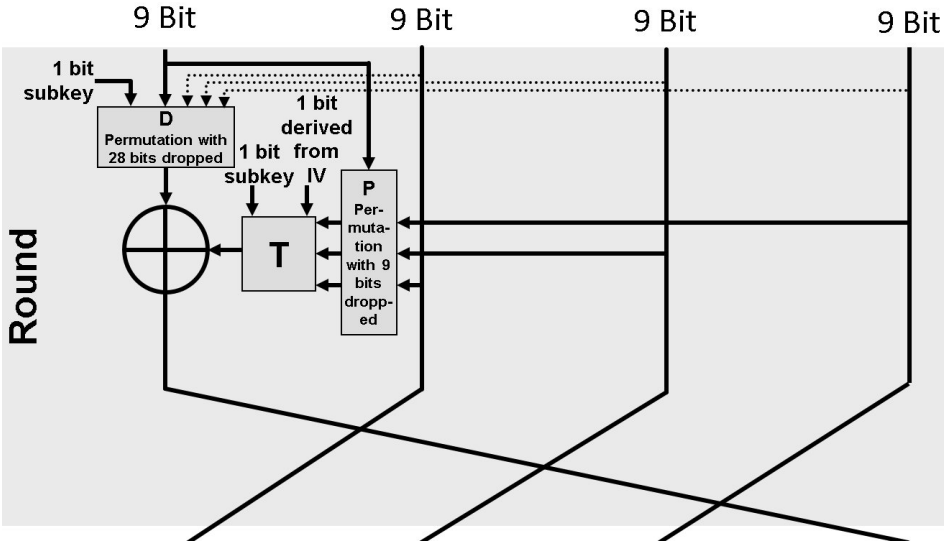


Fig. 2.2. Outline of one round of T-310

The wiring inside this permutation of the block cipher inside T-310 is the same in each round, it is in general quite complicated and is implemented as a small plug-in board inside the cipher machine. This board is the principal part of the so-called "long-term key" and implements two functions D, P cf. Def. 4.1.1 page 14. More information about LZS notations used in this paper can be found in Section 4.1 cf. also later Fig. 7.10 in Section 7.1.

The non-linear round "compression" component T will be specified in Section 9 and the non-linear Boolean function used inside will be specified in Section 10.1.

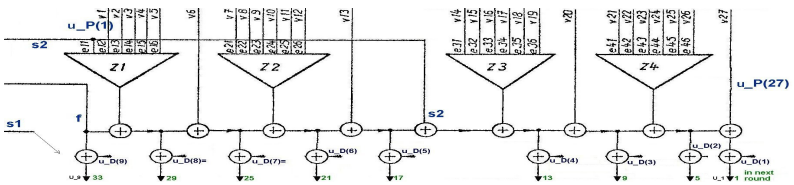


Fig. 2.3. A first look at the internal structure $T()$ inside one round together with the final XORs.

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. [84] and Fig. 4.5 below.

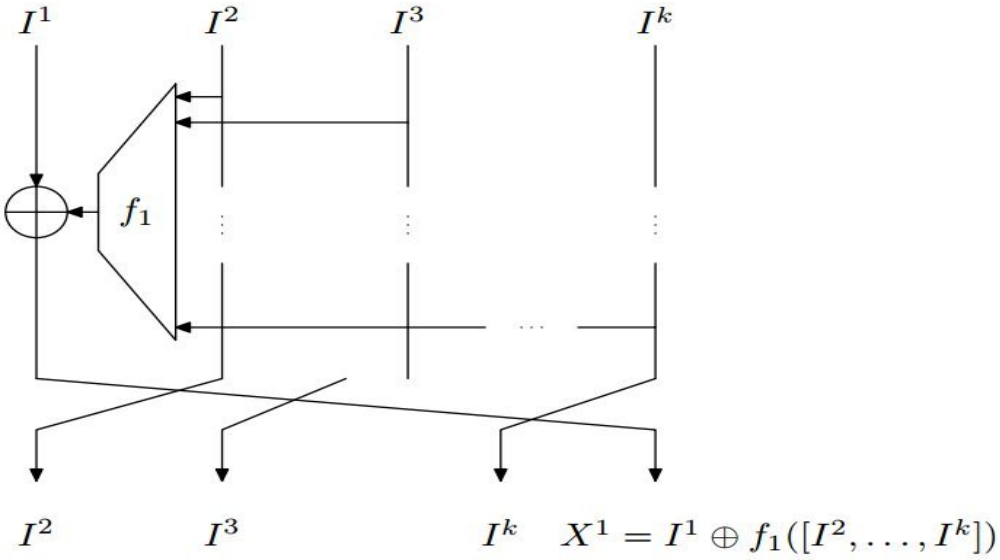


Fig. 3.4. A "Contracting Unbalanced Feistel" cipher with k Branches.

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 [65, 66]. Then East German cipher designers had already in the 1970s [11, 96, 61] 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. [78] with an (alleged) collaboration with the NSA. RC2 have been very widely used worldwide for real-life communications security, first 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. [86, 84, 85, 74].

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 [93]. The earlier RC2 has remained a trade secret for a longer time and only in 1997-1998 it was re-discovered and analysed (without great success) in crypto community [78]. 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 [85] sort and more like local application of the basic Feistel with only two branches at one time, with a lot of extra irregular structure [75]. 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" [8].

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 [84]. With very strong round functions this theory would recommend at least 8 rounds for a cipher with 4 branches [84], 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 [78]. Similarly, to this day there is no attack on the full Skipjack cipher cf. [75] and the SHA-1 when used in encryption is also quite robust [80]. 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 [84] is as follows.

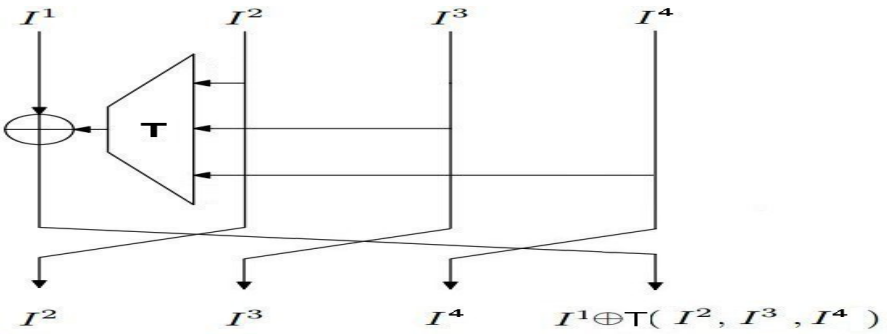


Fig. 4.5. A Contracting Unbalanced Feistel cipher with $k = 4$ branches.

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. [56]. The original term is LZS which is an abbreviation of *Langzeitschlüssel* as opposed to weeklykeys "ZS" of *Zeitschlüssel* which are perforated cards (with holes punched in them). The LZS would be changed roughly once per year, cf. [55] or only when "necessary" cf. [56].

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. 4.5 in a very substantial way 4.5. 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. 4.5.

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 [100].

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 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. 4.6 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. 5.7, but will also consider for example $\underline{\mathbf{D}} : \mathbb{F}_2^{9+3} \rightarrow \mathbb{F}_2^9$ cf. Section 5.6. 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 [14, 10], 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 8.6. There also exist a number of special anomalous keys listed in [55]. In this paper we also consider plenty of types of special (weak) keys, see Section 22, and at countless other places.

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 4.4-5.6 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. Some further classification of KT1 keys can be found in Sections 21.18 and 21.19. 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 cf. Sections 4.5, Section 18, 19 and in Sections 21.4 and **??** and in Appendix E and E.6. Some 7 keys of type KT1 have been used in practice in the period of 1979-1990, cf. Section 8 and [55] cf. also Section 8.3 and Appendix B.2.

[5] This is in principle mandatory in T-310, cf. page 115 of [100]. Not true for the so-called "testing key" 17 from 1979 which has $P(25) = P(26)$, cf. [55].

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. 4.5 by a permutation of wires $\underline{\mathbf{D}}$ applied to I^1.

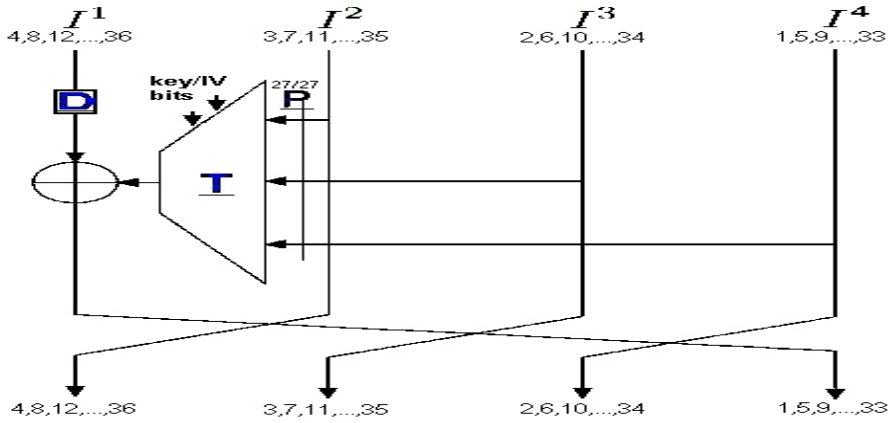


Fig. 4.6. T-310 variant which is like a classical unbalanced Feistel reinforced by D.

Then $\underline{\mathbf{P}}$ could also be 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. 4.5.

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[6] 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. 4.6 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[7] 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] Not without substantial additional differences, cf. Appendix F.1).

[7] The actual way to define D is slightly different in [100, 96], 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. 4.6 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. 4.6. Then we get an **unexpected result**. In such a configuration the bit called v_0 in [96] 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 "weekly" key would be reduced from 240 to 120 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 120 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.

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 [100].

More precisely, both types of recommended T-310 keys D according to [100]. 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 [100]. Moreover for the KT1 keys we always have $D(1) = 0$ cf. page 256 in [96] and page 55 in [100], while for KT2 keys we always have $D(i) = 0$ for some $1 \leq i \leq 7$, cf. page 59 of [100].

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. 5.7 which takes one additional input sometimes called v_0, cf. [96] 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 5.6 below.

[9] As we will see below, or at least in all historical cases known to us which are complaint with [100], 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.

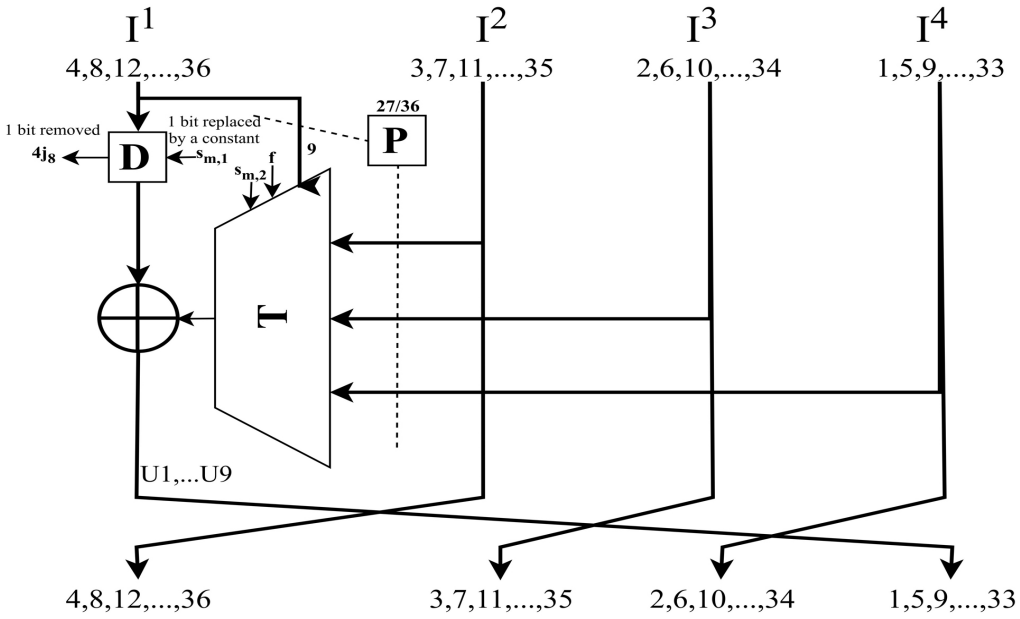


Fig. 5.7. T-310 is NOT exactly a simple unbalanced Feistel scheme. In the common KT1 case, the spec allows to use also bits from the leftmost branch I^1 under a number of highly technical conditions. It also disconnects[11] ONE of the 9 bits in the left branch and replaces it by a key-dependent constant $s_{m,1}$ 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 the KT1 case there will always be 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 [100, 96], instead a complex set of constraints which D and P must satisfy is given. cf. Appendix B. Now, for all KT1 keys of [100] we observe that just one[10] bit from leftmost branch I_1

[10] In [96] 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 [96].

gets disconnected[11] and it is replaced by a constant[12] 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 [100, 96] 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 T. Therefore in a typical version of T-310 as per [96], 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. 3.4 above. This means that there will be serious difficulties in making sure that our "tweaked" generalized Feistel is still going to be a bijection[13]. Depending on the choice of D, P in T-310 decryption[14] 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 for the KT1 keys 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. [84–86] 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[15] number of rounds than other similar ciphers [e.g. RC2] to be provably secure or secure against "generic" attacks.

---

[11] This has many interesting consequences in general (discussed in the present section) and in particular cf. later Thm. B.1.1 page 119.

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

[13] This question is studied in Appendix C.

[14] Decryption, in the sense of computing the previous states of the T-310 generalized Feistel variant, is **not** needed in the normal operation of the cipher. However for this cipher to be bijection, is needed as a structural property cf. our Appendix C. It was clearly imposed by designers of T-310 starting from 1974, cf. page 109 in [61], and this for some very good (security) reasons such as preventing the entropy of the cipher state from being depleted by iteration.

[15] 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 [84] - could make it substantially stronger and avoid attacks such as splitting the cipher in 2 loosely connected parts, cf. Fig. 3-4 in [44, 45].

5.5 Comparison to SKS Ciphers and How It Impacts T-310 Ciphers

The predecessor of T-310, SKS cipher already had the property that it was disconnecting one bit of $I^{(1)}$, see Fig. 6.41 page 148 and Fig. 2.28 page 119. This property is quite special. It turns out that many T-310 long-term keys also mandate a similar quite strong structure. This property is unusual and maybe even unnecessary and it is clear that it was NOT motivated exclusively by the question of invertibility, cf. Appendix C.11 page 132. It is rather related to the history of development of ciphers in Eastern Germany [61, 11]. It is about imitating indeed the structure pertaining to SKS ciphers seen on Fig. 2.28. This is illustrated in Fig. 2.29 page 119 and it is also the object of Thm. B.1.1 page 119.

5.6 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 Section D.1 more then one bits can be disconnected, but again, only one is replaced by a constant. This type of keys is less well understood.

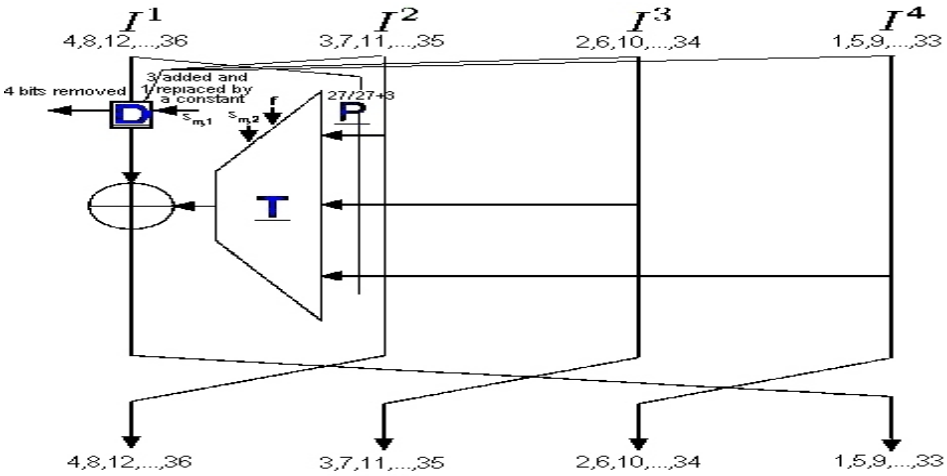


Fig. 5.8. Connections of T-310 when using the KT2 key 15 from [55].

Comparison KT1 vs KT2. Keys of type KT1 and KT2 differ very substantially. For example for the 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 always be disjoint, cf. Section D.1 or/and page 59 in [100].

An example of a real-life KT2 key is the key 15 from 1979 in [55] which we have verified to satisfy all the conditions of Section D.1. In this example only 3 bits of I^1 are used in T, while 9 are typically used for the KT1 long-term keys (which are illustrated in Fig. 5.7). In KT1 we always have $\underline{\mathbf{D}} : \mathbb{F}_2^{9+1} \rightarrow \mathbb{F}_2^9$, while in KT2 we may have $\underline{\mathbf{D}} : \mathbb{F}_2^{9+3} \rightarrow \mathbb{F}_2^9$ but also for example we could have $\underline{\mathbf{D}} : \mathbb{F}_2^{9+2} \rightarrow \mathbb{F}_2^9$ cf. Appendix E.3.

6 Detailed Description of T-310

Given Fig. 6.9 and Fig. 5.7, in order to fully specify the cipher T-310 we need:

1. To specify u_0 the initial 36-bit state I^{1-4} of the block cipher which is a constant equal to 0xC5A13E396, cf. [96].
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 $T : \{0,1\}^3 \times \{0,1\}^{27} \to \{0,1\}^9$ cf. Section 9 below.
4. To specify the key/IV scheduling: how the 3 bits of the key and IV $(f_m, s_{m,1}, s_{m,2})$, $m \geq 1$ used by $\underline{\mathbf{D}}$ and T are generated, for each round $m \geq 1$, cf. Section 13.
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. 6.9 below we show how all these things come together.

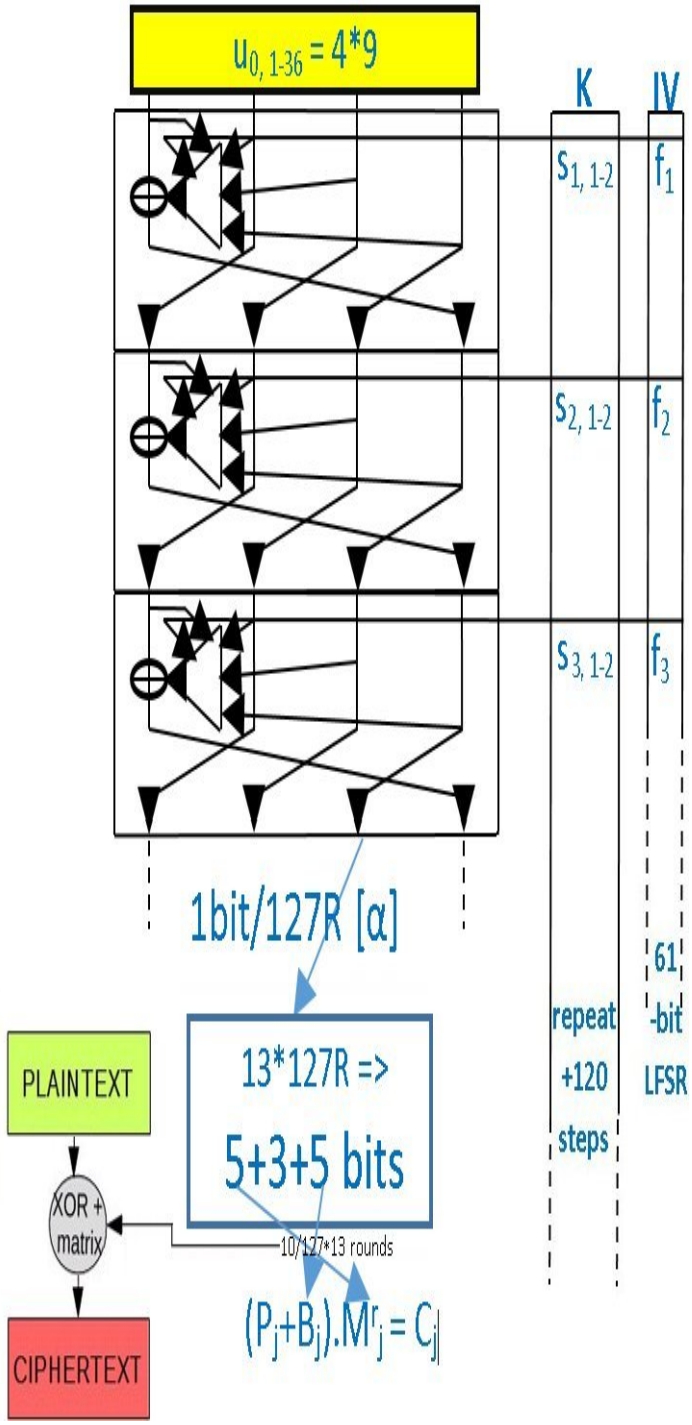


Fig. 6.9. T-310 Cipher.

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:

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

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. 4.5 and Fig. 5.7:

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

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

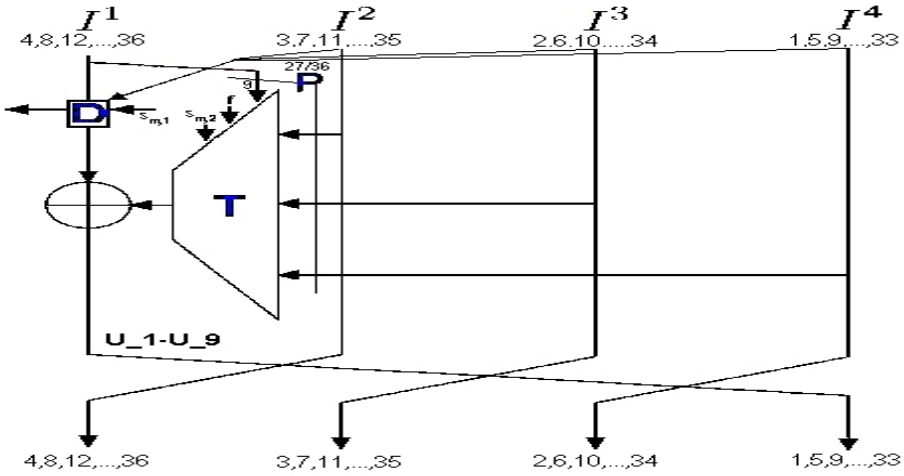


Fig. 7.10. High-level internal structure of one round of T-310

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[16] used (depending on cases). The k-th output of $\underline{\mathbf{P}}$ is defined as:

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

If we denote the 27 outputs of $\underline{\mathbf{P}}$ by v_{1-27} 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 $\underline{\mathbf{T}} : \mathbb{F}_2^{2+27} \to \mathbb{F}_2^9$ 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.

[16] Specific examples of missing bits are listed in Table 1 page 42 which leads to some important differential properties, cf. Section 12.3.

We should note that T has a complex internal structure which we will detail later. On Fig. 7.11 below we provide a quick preview.

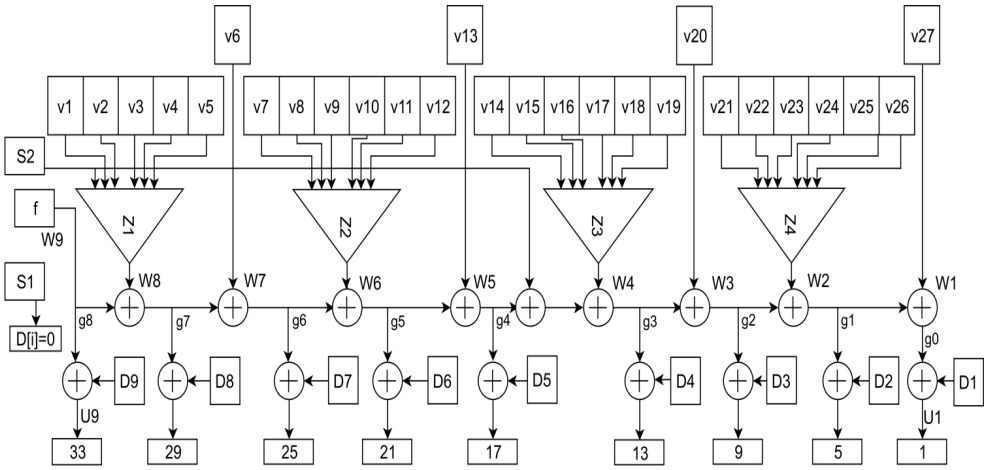


Fig. 7.11. Internals of T in one round of T-310. See Section 9 for more details.

Quick explanation how P, D work on Fig. 7.11. For example $P(26) = 5$ would mean that we connect the second rightmost output U_2 (or state bit 5 -here in green- from I^4 on Fig. 5.8 a.k.a. u_5) to v_{26} in the next round. Then $D(9) = 4$ means that first bit u_4 (in green) from I^1 on Fig. 7.10 was XORed to the state when computing U_9, which U_9 becomes bit 33 of I^4 in the next round. Furthermore let $u_0 = s1$, 0 is not a state bit number but a placeholder for $s1$, for example $D(1) = 0$ would mean that we would XOR $s1$ in computation of U_1.

7.4 Definition of $\underline{\mathbf{D}}$

We focus primarily on KT1 case represented in Fig. 5.7. 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[17] which "replaces" the bit which[18] is "removed" in the 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 of 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 $\underline{\mathbf{D}} : \{0,1\}^1 \times \{0,1\}^9 \rightarrow \{0,1\}^9$ defined as follows, where we use a quite unusual numbering of inputs to keep it partly compatible with [96] and Fig. 5.7.

$$\underline{\mathbf{D}}_i\,(s_1;\ u_4, u_8, u_{12}, \ldots, u_{36}) = s_1 \quad \text{when} \quad D(i) = 0$$
$$\underline{\mathbf{D}}_i\,(s_1;\ u_4, v_8, u_{12}, \ldots, u_{36}) = u_{D(i)} \quad \text{when} \quad D(i) \neq 0$$

[17] This bit was called v_0 in [96].

[18] It happens in fact at position equal to $4j_8$ following the notations used in [96].

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 5.7 and therefore we have for any $m \geq 0$:

$$(u_{m+1,1}, u_{m+1,5}, u_{m+1,9}, \ldots, u_{m+1,29}, u_{m+1,33}) \stackrel{def}{=}$$
$$(U_1, U_2, U_3, \ldots, U_8, U_9) \stackrel{def}{=}$$
$$\underline{\mathbf{D}}(s_{m,1};\ \ u_{m,I^1}) \oplus \underline{\mathbf{T}}\left(f_{m+1}, s_{m+1,2},\ \ \underline{\mathbf{P}}(u_{m,I^{1-4}})\right) =$$
$$\big(u_{m,D(1)} \oplus T_9(f_{m+1}, s_{m+1,2}, u_{m,P(1-27)}),$$
$$u_{m,D(2)} \oplus T_8(f_{m+1}, s_{m+1,2}, u_{m,P(1-27)}),\ \ u_{m,D(3)} \oplus T_7(f_{m+1}, s_{m+1,2}, u_{m,P(1-27)}), \ldots$$
$$\vdots$$
$$\ldots, u_{m,D(8)} \oplus T_2(f_{m+1}, s_{m+1,2}, u_{m,P(1-27)}),\ \ u_{m,D(9)} \oplus T_1(f_{m+1}, s_{m+1,2}, u_{m,P(1-27)})\big)$$
$$\text{where by convention input}\quad u_{m,0} \stackrel{def}{=} s_{m+1,1},\quad m \geq 0$$

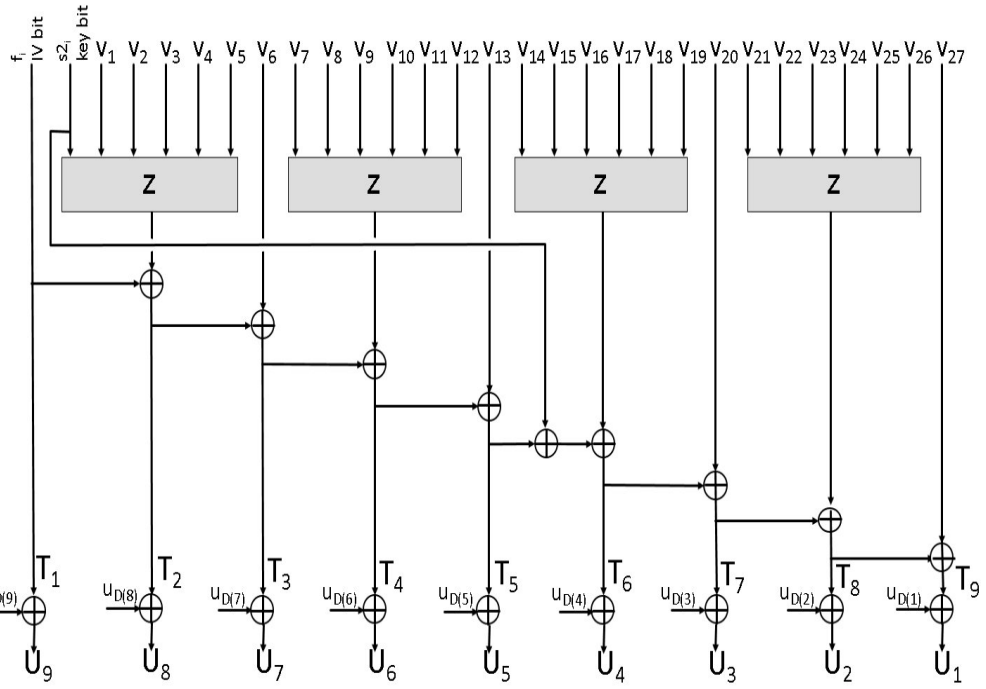


Fig. 7.12. Internal structure inside one round of T-310: computation of 9 outputs T_i of $T()$ which we will specify later in Section 9 is followed by 9 XORs (the U_i).

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

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

Notes on notation: We use the letter ϕ following [96] and we consider that $\phi : \mathbb{F}_2^{3+36} \to \mathbb{F}_2^{36}$. Similar but different notations are used in [100]: 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[19]. 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 [100] will still be denoted by ϕ.

[19] In [100] 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 [100].

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

There is something interesting which is revealed by our new notations **D** and **P**. We observe that ONLY half of the key bits (120 bits) will ever be used to form any of the $s_{i,1}$ used by **D**, and another disjoint half of key bits is used to form the $s_{i,2}$ used in **P**. These two sets of bits never mix and permanently play 2 different [disjoint] roles inside our iterated block cipher process. We also can observe on Fig. 7.11 that a change in $S1$ will flip just one bit, while a change in $S2$ will flip 4 bits typically.

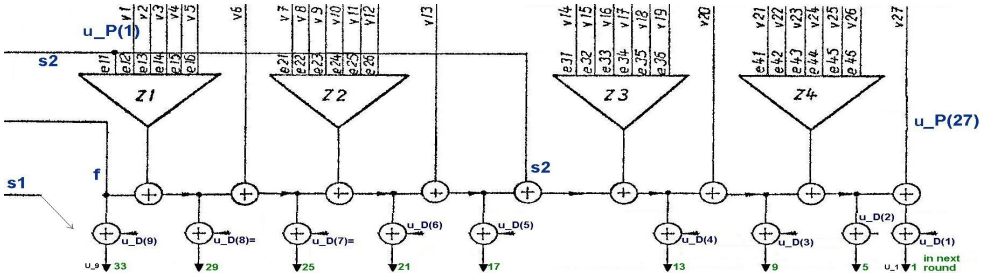


Fig. 7.13. The role of key $(s1, s2)$ and IV bits (f) in one round in T-310 and also in SKS ciphers adapted from an original drawing from [58].

This fact alone is potentially a serious **design flaw** in T-310 cipher (and for the SKS cipher) and suggests there might be some divide-and-conquer or guess-then-determine attack, where initially the attacker would guess only half of the key bits etc and confirm this guess without knowing the other half. For example:

1. It is easy to see that bits $S2$ affect more bits in one round, while $S1$ will affect just one bit[20].
2. Or we could work on 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 **D** primarily and if we guess this half of the key we should get a specific recognizable pattern.
3. Or we we could design an attack on T9 [or on one output bit correlated with $u_{P(20)}$] following the observations of Section B.2 cf. Table 26 page 120 and Conj. B.2.3.

We don't know if such attack will be efficient - the attacker does NOT easily get access to see all the flipped bits, cf. Section 14.1-16. Now 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 **D** and **P**. We are facing here a strong vulnerability almost inevitably susceptible to lead to some interesting attacks. Another interesting observation is that bits S2 is used twice each in round, bit S1 only once and it is initially combined linearly. This makes that sometimes key S1 will be "ineffective", effectively shortening the effective key size of T-310, cf. Thm. 12.4.1 page 44 or Section 12.4.

[20] Eventually $S1$ will affect ALL state bits 1-36 in a certain order cf. App. C.12.

8 Long Term Keys D, P

The long-term key D/P are not fully specified in [100, 96], however some historical examples of D/P can be found in [55]. It appears that the "main" historical versions of T-310 were primarily using KT1 keys. KT1 keys are defined in [100]. 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 [55]. Then there exists another substantially less popular class of long-term keys KT2. The sources and [100, 55] 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 [100, 96]. Instead a set of peculiar constraints on D/P are specified. In [96] only the so-called KT1 class of keys of [100] is specified and it is not specified[21] exactly. In Appendix B we provide a complete specification of this class. In [100], another class of keys KT2 is specified. Both these classes of keys are clearly meant to make Fig. 5.7 have the desired properties such as invertibility and possibly other which need yet[22] to be elucidated.

Several real-life historical examples of keys D/P from 1977-1990 can be found in [55] which are given numbers[23] of type `Der Langzeitschlüssel 14: (1979)`. In our research by default we will use the following real-life long-term key number 26 from [55]. We have carefully checked that key 26 belongs to the so-called KT1 class which is fully described in [100] also [not completely] described in [96].

```
//Der Langzeitschl"ussel 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 of the other keys in [55] 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 [100, 96]. Other important properties of KT1 keys are studied in Section 5.3, Section 5.4 and in Appendix C.

[21] We demonstrate this fact in Section E.6.

[22] 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. [55].

[23] It appears that only keys 14,15,21,26,30,31,32,33 found in [55] are for T-310. Other are keys for different East German encryption machines other than T-310, which are also studied in [55]. Circumstantial evidence shows that this key numbering should be consistent with earlier documents, for example in page 42 of [100] we read that key 14 are of type KT1 and key 15 are of type KT2, which is also true for keys found in [55].

8.3 On Strength of Real-Life Cold War Keys

The real live KT1 keys 14,21,26,30,31,32,33 are expected to be particularly strong choices of KT1 keys and until now no weakness whatsoever for these keys have been reported. Here is one: in Table 26 page 120 we see that all except the first of these keys have $\alpha \in 1-4$. Then, at the first sight, this is potentially the strongest possible choice, due to the structure inside $T()$ cf. Fig. 7.12, these bits depend on all the inputs of all the 4 Boolean functions Z1-4, and on all the key bits, and on the IV bit. However if we study some related-key differential attacks, cf. Section 12.4, we see that strangely, bits 1-4 are very frequently those which appear in such properties. No keys are exempt from RKDC properties as far as we know. Moreover Table 26 page 120 also suggests that historical keys could be vulnerable to correlation attacks on S1 bits specifically. In general using bits 1-4 is a very strong choice, due to the fact that these bits depend on all of Z1-Z4, see for example Section 21.7 and 21.8.

8.4 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 [100]. We have checked that the only known authentic key of type KT2 which [24] is key 15 from 1979 in [55] 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 key-dependent constant could weaken the cipher. It is also worth noting that for KT2 keys as described in [100] more than one bit will be disconnected contrasting with KT1 keys, but in both KT1/KT2 cases **only one** is replaced by a constant.

The keys of type KT2 are also studied in Section 5.6, Appendix D and Appendix E.

8.5 Other Keys and Key Classes

We have tested all the keys which are indicated as keys for T-310 in [55] 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 [96]. In Appendix E.6 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.

[24] We also study this key in Fig. 5.8 of Section 5.6 page 22.

8.6 Key Sizes for the Long Term Keys

According to page 56 in [100], the entropy of (D, P) belonging to class KT1 is between 78.1 and 79.7 bits. For the 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 the 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 [100]. 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. A more recent evaluation can be found in Section 5.4. of [49] the space of KT1 keys has approximately $2^{83.2}$ elements.

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. [39]. This small LZS key space suggests that cryptanalysts could also hope to break T-310 when the long-term settings are unknown to the attacker.

8.7 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 [16] and in the same way as a bad choice is what makes GOST weak, cf. [40], and leads to some very good attacks, cf. again Fig. 3-4 in [45] and all the attacks of [44–46]. Several distinct classes of particularly weak keys for T-310 are studied in this paper, see Section 22.

9 Description of T

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

$$
\begin{aligned}
T_1\,(f;\ s_2;\ v_{1-27}) &= \mathbf{f} \\
T_2\,(f;\ s_2;\ v_{1-27}) &= T_1 \oplus Z(\mathbf{s_2}, v_{1-5}) \\
T_3\,(f;\ s_2;\ v_{1-27}) &= T_2 \oplus v_6 \\
T_4\,(f;\ s_2;\ v_{1-27}) &= T_3 \oplus Z(v_{7-12}) \\
T_5\,(f;\ s_2;\ v_{1-27}) &= T_4 \oplus v_{13} \\
T_6\,(f;\ s_2;\ v_{1-27}) &= T_5 \oplus Z(v_{14-19}) \oplus \mathbf{s_2} \\
T_7\,(f;\ s_2;\ v_{1-27}) &= T_6 \oplus v_{20} \\
T_8\,(f;\ s_2;\ v_{1-27}) &= T_7 \oplus Z(v_{21-26}) \\
T_9\,(f;\ s_2;\ v_{1-27}) &= T_8 \oplus v_{27}
\end{aligned}
$$

Here $Z : \mathbb{F}_2^6 \rightarrow \mathbb{F}_2$ is a Boolean function defined in Section 10.1.

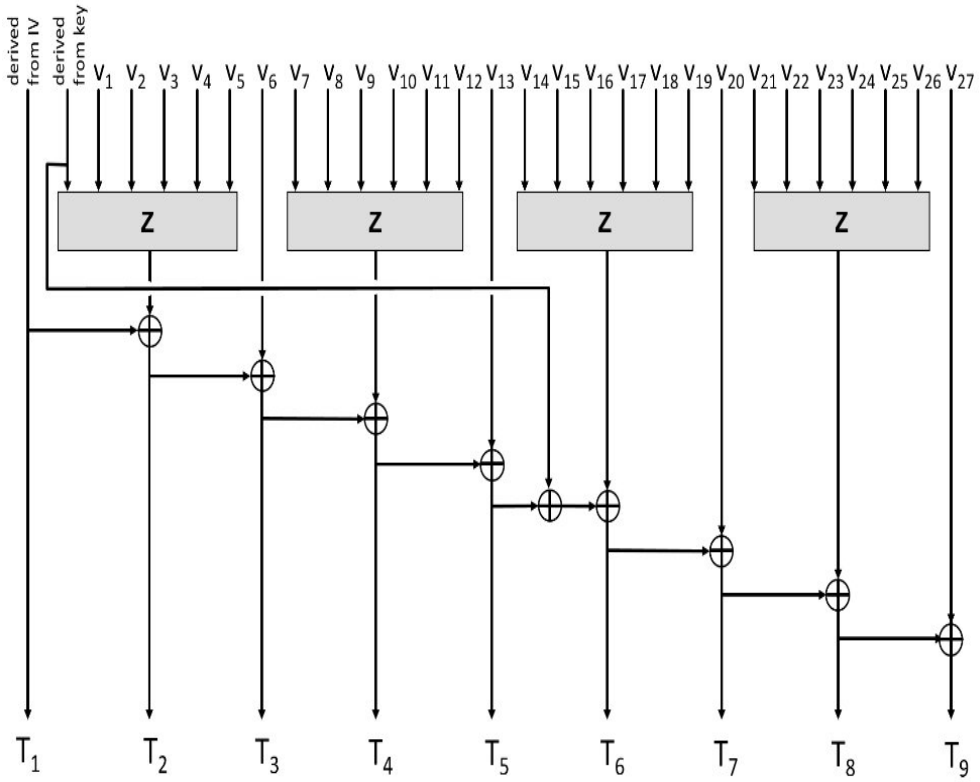


Fig. 9.14. Internal structure of T inside one round of T-310.

9.1 Design of T and Alternative Descriptions

We recall the input naming of [96]: $e_0 = f$, then $e_1 = s_2$, and then $e_2 = v_1$ up to $e_{28} = v_{27}$.

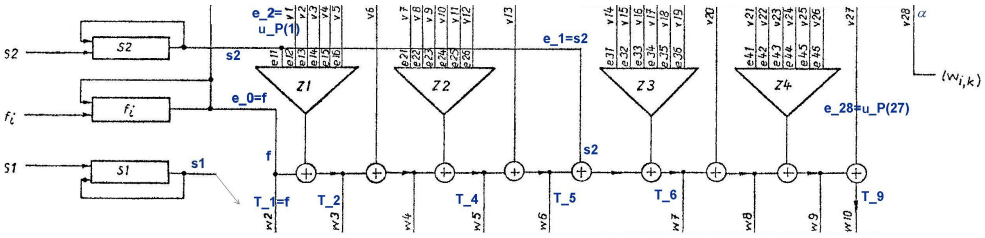


Fig. 9.15. A zoom inside the "complication unit" of SKS cipher [58] adapted for T-310, and comparison of notations in the description of $T()$ and e_i notations of [96].

The origin of Fig. 9.15 reveals that the structure of $T()$ was not quite designed for T-310 but in fact comes from an earlier cipher SKS V/1 which we study in Section F.

9.2 Design Criteria of T

We found an interesting document [59] which explains the origin why $S2$ is used twice inside this component. More precisely on page 54 of [59] from 1973 we found an earlier weaker design for the component $T()$ where $S2$ is used only once, and the authors report that there is "a certain restriction on the effectiveness of $S2$". More precisely we read that for exactly half of the inputs of the round function ϕ, the 9-bit output will be independent of the input $S2$. This property is clearly a property which the designers decided to remove.

9.3 Another Point of View on T and One Round ϕ

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

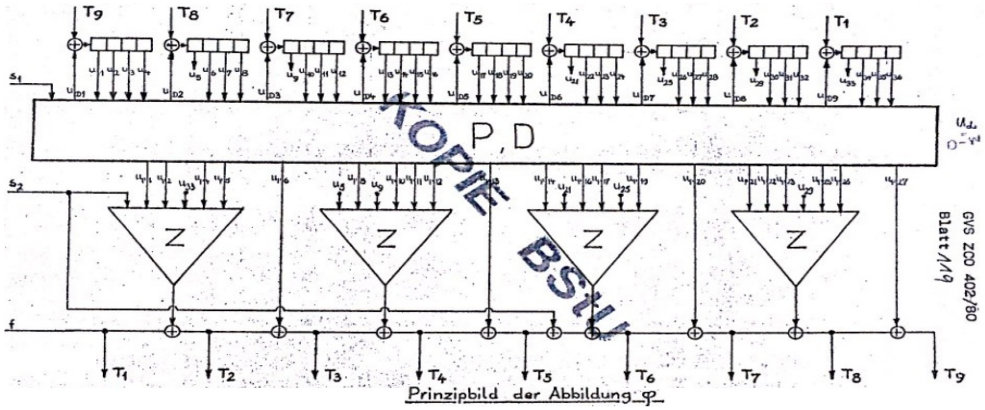


Fig. 9.16. T-310 round function seen as a stateful component $\mathbb{F}_2^{3+9} \to \mathbb{F}_2^9$ with 36 memory bits, cf. page 119 in [100] which also mandates a certain subset W of 6 special bits e.g. u_5 and u_{33} cf. page 117 in [100] and Fig. 2.27 in Appendix B.

9.4 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 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 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 at one time, in a specific order.

9.5 Vulnerabilities of the Whole T Component

It possible to see that the amount of non-linearity or Multiplicative Complexity [25] is quite low: there only four applications of Z for nine new $u_{i,j}$ state bits generated. Each application of Z has MC at most 51, cf. Appendix I.14 and Appendix I.16. In reality the result we would expect would be 5 or 6, see [9].

The T component is in fact not completely non-linear: it has linear I/O equations true with probability 1, for example on Fig. 9.16 we see that $w_9 \oplus w_{10} = v_{27}$. This has important consequences in terms of linear cryptanalysis, and there is a VAST amount of space for rich sets of linear attacks to exist, as we will see in Section 21.16 and in Section 21.9 and at many other places.

In Section 12.2 below we show that there is a systematic and specific 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 against differential cryptanalysis, cf. Section 12 below.

9.6 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 [25] 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:

$$u_0 \stackrel{def}{=} s_1 \tag{0}$$

$$U_9 = u_{D(9)} \oplus f \tag{9}$$

$$U_8 = u_{D(8)} \oplus U_9 \oplus u_{D(9)} \qquad \oplus Z_1(s_2, u_{P(1-5)}) \tag{8}$$

$$U_7 = u_{D(7)} \oplus U_8 \oplus u_{D(8)} \qquad \oplus u_{P(6)} \tag{7}$$

$$U_6 = u_{D(6)} \oplus U_7 \oplus u_{D(7)} \qquad \oplus Z_2(u_{P(7-12)}) \tag{6}$$

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

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

$$U_3 = u_{D(3)} \oplus U_4 \oplus u_{D(4)} \qquad \oplus u_{P(20)} \tag{3}$$

$$U_2 = u_{D(2)} \oplus U_3 \oplus u_{D(3)} \qquad \oplus Z_4(u_{P(21-26)}) \tag{2}$$

$$U_1 = u_{D(1)} \oplus U_2 \oplus u_{D(2)} \qquad \oplus u_{P(27)} \tag{1}$$

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}.

[25] 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.

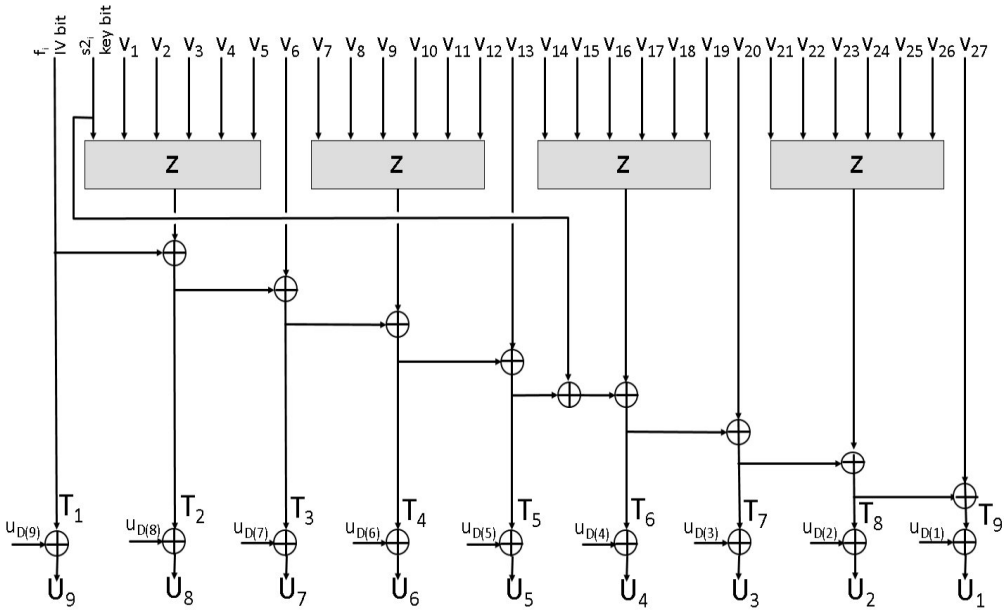


Fig. 9.17. Computation of $T()$ followed by 9 XORs with bits specified by $D()$

10 The Non-Linear Component of T-310

The only[26] 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 [100] 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 for SKS cipher in page 39 of [103], and which is the same as on page 256 in [96] except that the constant 1 is missing in [96].

Another Vulnerability. The fact that the same Boolean function is used everywhere is of course a potential vulnerability. For example, using the same

[26] 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.

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 [50] which attack was further improved in [68]. In this paper we exploit this property in Section 21.17.

10.2 Implementation of the Boolean Function Z

This question is studied in Appendix I.14 and Appendix I.16. In general T-310 is substantially more expensive than any other block cipher we have ever heard of. We refer to Appendix I.15 for more details.

10.3 Design Criteria for the Boolean Function Z from 1973

In this section we list the design criteria which were mandated by Eastern-German cryptologists in 1973 cf. [59]. These criteria were proposed for the Boolean function $Z()$ of the earlier SKS V/1 cipher, which was the father of T-310 cipher. We have verified that both Boolean functions are identical. These original design criteria are listed on page 53 of [59] as follows:

(1) $|\{X = (X_1, X_2, \ldots, X_6) \in \{0,1\}^6 | Z(X) = 0\}| = 2^5$

(2) $|\{X \in \{0,1\}^6 | Z(X) = 0, HW(X) = r\}| \approx \binom{6}{r} \cdot \frac{1}{2}, \quad r = 0..6$

(3) $|\{X \in \{0,1\}^6 | Z(X_1, .., X_i, .., X_6) = Z(X_1, .., X_i \oplus 1, .., X_6)\}| \approx 2^5, \quad i = 1..6$

(4) Z is not symmetric

It is noteworthy that the criterion (3) is related to Differential Cryptanalysis which was only officially studied in 1990s and they come from the same period of time when DES was designed [10, 14, 46]. This suggests that Differential Cryptanalysis could have been studied in the Eastern Bloc countries at least as early as in the U.S. and possibly earlier. This is confirmed by a detailed study of differential and higher-order differential properties of T-310 Boolean function found in [63].

10.4 Another Set of Design Criteria From 1975

We found another document from 1976, which gives a different detailed set of points to study or criteria which the Boolean function Z should satisfy, cf. page 30 in [61]. It also clearly states that these properties were specified 1 year earlier in 1975 and that they are studied in more detail in [63].

1. All derivations of Z were computed as Zhegalkin polynomials[27] and as value tables.

[27] This is the same as Algebraic Normal Form (ANF), and were initially invented by a Russian mathematician Zhegalkin as early as in 1927. It is noteworthy to remark that only 1936 an American mathematician Marshall Stone has reflected on the fact that this is a very useful and simple method to "arithmetize" the Boolean algebra which is of great interest and led Stone to substantially rewrite his paper, cf. [107].

2. Frequency of the function result being '1' with k fixed values was computed for $(k = 1, 2, 3)$.
3. The *statistic structure*[28] of the Boolean function was computed.
4. Z is not symmetric. This means that the function value changes if the arguments are permuted, if one or several arguments are negated, if the function is negated or if a combination of these three changes is applied.

Later the document also states that 1. and 2. are important requirements for further examinations. Then it expands 3. saying that statistic structure did not reveal any cryptographic advantages resulting from an approximation of the function Z via Boolean functions. Finally it says (in relation to 4.) that due to the asymmetry [29] of Z there are less equivalent long-term keys.

[28] Examination of pages 17-18 in [63] makes it crystal clear that this about computing linear characteristics of this Boolean function Z, and refers to a classified cryptography course, see our later Section 21.1 for more details.

[29] This property is demonstrated on pages 19-20 in [63].

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 multi-linear flavors which will be particularly interesting here cf. [18, 16] and many other, ElimLin and advanced variants thereof [15, 24, 99], 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 [100] 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 the 1970s have done a great deal of effort to make sure that T-310 LZS always lead to a bijective round function ϕ. Thus making powerful differential attacks with zero differentials impossible. In this paper we show several[30] examples of more or less devastating attacks which can be mounted against T-310 with non-bijective LZS. 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 [100].

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.

A similar result also holds for 1,2 and 3 rounds of T-310, with 2^{3n} images cf. page 49 in [100]. Then it does NOT hold for 5 rounds, see Section 12.3 below.

[30] See for example Section 20, Section 22.10 and Conj. 22.14.1 page 94. Cf. also Section C.1.

12 Differential Attacks and Vulnerabilities in T-310

Differential cryptanalysis is one of the oldest and one of the most powerful and generally applicable attacks on block ciphers, cf. [17, 46, 14, 44]. We refer to [46] for a survey and pointers on the "confidential" history of development of differential cryptanalysis. In [96] we read that it is not clear if the designer knew about linear and differential cryptanalysis (LC/DC). It is however clear that a most basic form of Differential Cryptanalysis when the attacker flips just one bit was already anticipated and explicitly defended against at the level of the Boolean function $Z()$, and this already for the predecessor of T-310 not later than in 1973, cf. the design criterion (3) in Section 10.3.

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 versions of T-310 are vulnerable to DC and that the designers [cf. Section 10.3] have been mistaken to assume that no obvious DC attacks can be mounted.

12.1 Structural Differential Attacks vs. S-boxes

An important point about differential cryptanalysis is that making components such as S-boxes secure against Differential Cryptanalysis (DC), cf. again (3) in Section 10.3, does **NOT** make a cipher secure against DC. Ciphers can contain serious "loopholes" w.r.t. differential attacks which are not due to the S-boxes but to the connections of the cipher. We will see a specific example below: differential vulnerabilities due to the fact that some bits are not used in a round function, cf. Section 12.2 below.

More generally 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 based on the way in which the components are connected. In many such attacks and to some extent we CAN ignore the exact content of the S-boxes cf. [86], or we can hope that inevitably an attack will be found for any set of S-boxes, cf. [44–46]. However for other ciphers, the properties of the S-boxes will be very important for DC attacks. There exist also many more advanced forms of differential cryptanalysis such Truncated DC, cf. [76, 44–46, 19]. DC with 3,4, and even 256 multiple simultaneous differentials [40, 35, 71] and where DC can be combined with other attacks such as software algebraic attacks [40, 38, 64]. Overall we need to understand that to insure the non-linear components of a cipher are somewhat "secure" against DC does not give any guarantee that the whole cipher is secure against DC.

12.2 Missing Bits - Serious Differential Vulnerability of T for Any P

We observe that in the situation of T-310 in Fig. 5.7 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

LZS 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 12.2.1 (Missing Bits in T-310). We conjecture that 9 bits are systematically missing, which is clear for the KT1 keys due to criteria listed in [96], 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 [100]. It is possibly due to the fact that the state in earlier SKS cipher had only 27, not 36 bits, and due to some engineering or cost considerations in the engineering or production phase.

12.3 Missing Bits - Applications

This property of Table 1 and Conjecture 12.2.1 is very likely to weaken the cipher against various differential attacks such as [44, 45] and in general.

Missing Bits - Applications - Example 1

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 12.3.1 (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.

Missing Bits - Applications - Example 2

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. 26.0.1 page 104.

12.4 Missing Bits - Application to Related-Key Differential Attacks

These missing bits play an important role in related key attacks on T-310. For example we have found the following KT1 key which we will call 716 and another similar key 722:

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

For these keys there exist differentials on the key which can be annihilated leading to no difference on the full 36-bit state of the cipher, this for only 5 rounds.

Table 2. Missing bits for some keys vulnerable to related-key differential attacks

LZS nb	bits which are not used in $P(j)$
716	1, 2, 10, 14, 15, 18, 22, 27, 35
722	2, 3, 6, 10, 13, 19, 23, 26, 31

For simplicity we focus on key 716. The exact form of internal differential that eventually is annihilated by a difference on a key bit is $1 > 2 > 3 > 4$ or if take into account also key and IV bits it is $s1 > 1 > 2 > 3, s1 > 4, s1$.

A Detailed Explanation for LZS 716

Table 3. Some LZS vulnerable to related-key differential attacks

LZS nb	related-key differential trail
716	$1 > 2 > 3 > 4$

More precisely we have the following sequence of events for 5 rounds:

1. In the 1st round, the difference on key bit $s1$ becomes a difference on $U_1 = u_1$, cf. Fig. 9.17 page 37.
   A formal explanation is as follows: we have $U_1 = u_{D(1)} \oplus U_2 \oplus u_{D(2)} \oplus u_{P(27)}$ following the last 10th equation in Section 9.6 Then we have $s1 = u_0$ and $D(1) = 0$ and we obtain the equation denoted as (1) in Appendix C.10: we have $U_1 \oplus s_1 = U_2 \oplus u_{D(2)} \oplus u_{P(27)}$. The right hand side does not change and a flip on key bit $s1$ in round 1 will affect bit u_1 at the input of the second round. Accordingly bit 1 and only this bit is flipped after the first round.
2. Then inside round 2, according to Table 2 this input bit 1 is NOT used, it does NOT enter the found function, and it simply becomes bit 2 [this is a substantial weakness]. Thus only bit 2 is flipped after the 2nd round.

3. Then in round 3, according to Table 2 this bit 2 is still NOT used otherwise than it becomes bit 3. Thus only bit 3 is flipped after the 3nd round. We have the following differential characteristic $s1 > 1 > 2 > 3$.

4a Then in round 4, we have $P(27) = 3$ and bit 3 is used. Again we have $U_1 \oplus s_1 = U_2 \oplus u_{D(2)} \oplus u_{P(27)}$ so the only thing a difference 1 on bit 3 would do is to flip the bit $T_9 = U_2 \oplus u_{D(2)} \oplus u_{P(27)}$ which flip will be annihilated by a flip in bit $s1$ inside the same round and overall the bit u_1 is not flipped after round 4. We have $s1 > 1 > 2 > 3, s1$.

4b At the same time, the flip will still be copied from wire u_3 which becomes u_4 now. We have $s1 > 1 > 2 > 3, s1 > 4$.

5a Then in round 5, we have a difference on bit 4 entering. The construction of T-310 prevents this from beign cancelled too easily. We have $P(23) = 4$ which is one of the inputs s_4 of Z_4 in 4th round. A detailed examination of the truth table of $Z()$ shows that the difference on bit s_2 is cancelled with probability close to 1/2 and this **no matter what are all the other 5 input bits** in each case, systematically and uniformly[31] over such choices.

5b At the same time we have $D(2) = 4$. In the fifth round the bit 4 has another effect: it would normally flip u_1, except that we have again $U_1 \oplus s_1 = U_2 \oplus u_{D(2)} \oplus u_{P(27)}$, and this modification can be cancelled by flipping bit s1 inside the 5th round.

!! Overall our differential is extinguished with probability 1/2 over 5 rounds as follows: $s1 > 1 > 2 > 3, s1 > 4, s1$.

Comments. This property is quite unusual and 716 is an example of an exceptionally weak KT1 key where this sort of property holds for 5 rounds. This sort of properties were a subject of considerable care for the designers of T-310 in the 1970s, see Thm. 11.2.1, which states that the same result should not hold for 4 rounds. Our simulations indicate that for other "typical" or random KT1 keys this property will be substantially weaker. It will typically be observed for 10 or more rounds, and almost never for less than 9 rounds, and also typically with substantially worse/lower propagation probabilities[32]. than 1/2.

A General Result

From the above we see immediately that:

Theorem 12.4.1 (A 5-round Related-Key Differential). The extinguishing related-key differential property $s1 > 1 > 2 > 3, s1 > 4, s1$ works for each long term key such that $D(1) = 0$, $D(2) = 4$, $P(23) = 4$ and such that $\forall_j P(j) \notin \{2, 3\}$. For such LZS, the property is triggered by imposing the $s1$

[31] This is quite surprising, and due to the requirement (3) listed on page 53 of [59], (cf. also Section 10.3 and Section 3.3. of [49]) which is exactly as follows:
(3) $|\{X \in \{0,1\}^6 | Z(X_1, .., X_i, .., X_6) = Z(X_1, .., X_i \oplus 1, .., X_6)\}| \approx 2^5, \ \ i = 1..6$

[32] These events and their relative probability will be conditioned/depending on the IV in these rounds, the keys $s1, s2$ in these rounds, and on the initial state on 36 bits.

key difference for 5 rounds is 0x19, and for any $s2$ key, for any IV, and for any cipher state on 36 bits and the differences are extinguished after 5 rounds with probability almost exactly equal to 1/2, cf. requirement (3) above.

Remark. A similar yet weaker result could be obtained for key 722. In this case the differential property is rather $s1 > 1 > 2 > 3 > 4, s1$ and the difference on s1 key bits is 0x11 which has only 2 active bits as we do longer need to flip a bit at round 4 as $P(27) \neq 3$ and bit 3 is not used. The propagation probability is 1/4, half of the previous result, due to the fact that in round 2 bit 1 is used as input of Z_4 with $P(26) = 1$, therefore the property (3) is used twice.

Software Tool. In Appendix I.11 we present a software solution for study of similar events and keys.

12.5 Examples of Differential Attacks on T-310

In this section we consider ordinary differentials ont he cipher state without modification of the key bits. Our student Matteo Scarlata (also another UCL student Mario D'Onghia has developed a different fast/parallel tool) has developed a software tool written in Python cf. Section I.13 and [95] for discovery of differential attacks on T-310. Here are some preliminary results for key 26.

Table 4. Some "good" differential properties for T-310.

LZS 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}
739	4	[1] → [33]	$2^{-1.0}$
739	4	[1] → [1,5,33]	$2^{-1.0}$
925	8	[7] → [19]	$2^{-4.6}$
925	8	[16] → [32]	$2^{-5.9}$
925	8	[28] → [1,5,9]	$2^{-4.2}$
925	16	[7] → [3,7]	$2^{-23.3}$

12.6 Differential Vulnerabilities with Different IVs

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

Fact 12.6.1 (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.

12.7 Differential vs. Linear Cryptanalysis

It is interesting to see that keys which are extremely weak with respect to Linear Cryptanalysis, cf. Section 21.3 and further Sections such as Section 21.14 etc, have also strong differential properties. We observe that long term keys which have very similar linear properties in Section 21.14 will however differ substantially in quality for related differential properties.

Table 5. Some differential properties for LC-weak keys from Section 21.14.

LZS nb	rounds	input → output	proba
706	2	[15] → [1,5,9,13,17,25]	$2^{-0.0}$
706	2	[35] → [1,5,9]	$2^{-0.0}$
706	3	[35] → [2,6,10]	$2^{-1.0}$
706	4	[35] → [1,3,7,11]	$2^{-2.0}$
706	8	[25] → [9]	$2^{-4.6}$
784	2	[9,13] → [11,15]	$2^{-2.2}$
787	2	[9,13] → [11,15]	$2^{-1.2}$
789	2	[9,13] → [11,15]	$2^{-1.0}$

13 Key and IV Scheduling Parts in T-310

13.1 Basic Facts About T-310 Keys

According to [56] 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.

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 [96] we have:

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

The initial key is $s_{1-120,1-2}$ which is 240 bits.

13.3 On Parity Bits

According to [96, 100] 10 out of 240 bits of the weekly/daily key ZS can be used as parity bits. It is very clear that such parity bits were initially specified by the designers of T-310, cf. page 117 in [100]. The exact specification of the parity check equations can be found in [96, 100] and in Appendix K.1. However based on informations provided by Jörg Drobick, cf. [55, 56, 58] parity bits were **never used in any real-life encryption**. Therefore we will always assume that the key size in T-310 is 240 bits.

13.4 IV Generation and Transmission in T-310

According to [96] 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 [100].

13.5 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 [96] is chosen at random. These bits are not used in encryption and the first bit used is f_1. The LFSR is defined by:

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

This corresponds to the polynomial $x^{61} = x^5 + x^2 + x^1 + 1$, cf. [58]. The period of this LFSR is $2^{61} - 1$ which is a prime.

13.6 A Zero-Attack on IV Expansion and f_m

We outline here a simple attack on T-310 if we have an access to a decryption oracle. We send a message where all the 61 IV bits are at 0 which is normally forbidden by the spec (cf. page 34 in [100]). Then we get a block cipher which has all IV bits at 0 and which will be subject to various self-similarity attacks: slide attacks, fixed point attacks, etc: the permutation is now the same and repeats every 120 rounds. We do not know if this attack is practical: how an original machine behaves in this case.

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 [96] (and called d in [55]) which governs the extraction of one bit every 127 rounds:

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

Then for each $127 \cdot 13$ consecutive rounds we discard 3 bits out of the 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 [100].

Remark. A basic observation is that a relatively large proportion of 10/13 of these bits will actually be used and conversely these bits are those for which the attacker may hope 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**[33] cipher design. It seems that it is actually potentially stronger[34] than other ciphers we have compared it to, such as RC2, DES, and Skipjack,. This is also what was intended by the designers in 1973 and this is also bellived to be a conclusion of the BSI report from 1990, 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 the cryptanalytic literature knows extremely few examples where a cipher could actually be broken under such difficult circumstances.

One major example is the so-called "Dark Side Attack" on MiFare classic [35, 71], 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 [35, 71]. 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.

[33] 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 [30].

[34] 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 [77].

15 Estimating the 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. [15, 36, 37, 24, 22, 40, 89]. 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. [31–33] 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[35] and is maybe comparable to Simon [24, 38, 22] which is a cipher of remarkable simplicity and extremely low multiplicative complexity, cf. [25]. Overall we do not expect that a SAT solver can break more than say 127 rounds of T-310 block cipher.

15.1 Attacks on 1 Bit - How to Access $u_{127,\alpha}$

Unhappily, in most cases, the attacker will dispose of only of up to 1 bit of information per 127 rounds and frequently less. The best possible case would be that the attacker disposes of $u_{127,\alpha}$. This is very difficult to obtain. Below we outline several possible scenarios which are studied later.

1. Even though the first 4 characters (or 20 bits) of the plaintext are always known to the attacker in real life situations. cf. Section 17.2 and Appendix I.5, this does NOT yet give access to the values $u_{127,\alpha}$, due to double one-time pad cf. Section 16.
2. Then it may seem that the attacker has little choice other than to work on the first character of the ciphertext C_1 and try to develop an attack on $11 \cdot 127 = 1397$ rounds simultaneously.
3. A possible solution could be our later "Zero-Value" attack and how it combines with KPA of Section 17.2. This does not work well, and gives one bit after 889 rounds, see paragraph **KPA Attack?** inside Section 17.4.
4. Better attacks can be designed if the attacker has access to a decryption oracle, see Section 23.

15.2 Attacks on Full State or P/C Pairs

In this paper show that there exists non-trivial methods which allow the attacker to generate full Plaintext/Ciphertext (P/C) pairs for less rounds, with 36 bits of information, in some[36] scenarios for as little as 120 rounds only[37]. It may be

[35] KeeLoq is really a bad example, a particularly weak cipher which can broken with time complexity as low as 2^{28} [31] and even 2^{23} for 15 % of keys, cf. [32].

[36] For example due to slide attacks.

[37] In some cases yet fewer rounds, for example due to faulty LZS e.g. in Appendix C.4.

indeed reasonable to expect that a SAT solver CAN break 120 rounds of the T-310 block cipher in a similar way as it can break 8 rounds of GOST, see [27] and Table 1, Section 9.1. in [40].

15.3 Computer Simulations

Our students Om Bhallamudi, Matteo Scarlata and Killian Davitt, have developed an open source software solution for testing the concept of software algebraic attacks on T-310. This software is described in Appendix I.10.
We give here a simple example of how it can be used:

```
python codegen.py 40-70 /fix3/4 /ins4 /xl /sat /LZS31 /timeout50*Nr
```

In this simulation we have fixed 3/4 of the bits of the key and determined another 1/2. The number of encryption rounds grows from 40 to 70. We have used CryptoMiniSat 2.96. Timings were done on one core of Intel i7-5600U CPU with nominal speed of 2.6 GHz with hyper-threading and under heavy load. We early abort all our computations and we report the first quartile Q_1 timing.

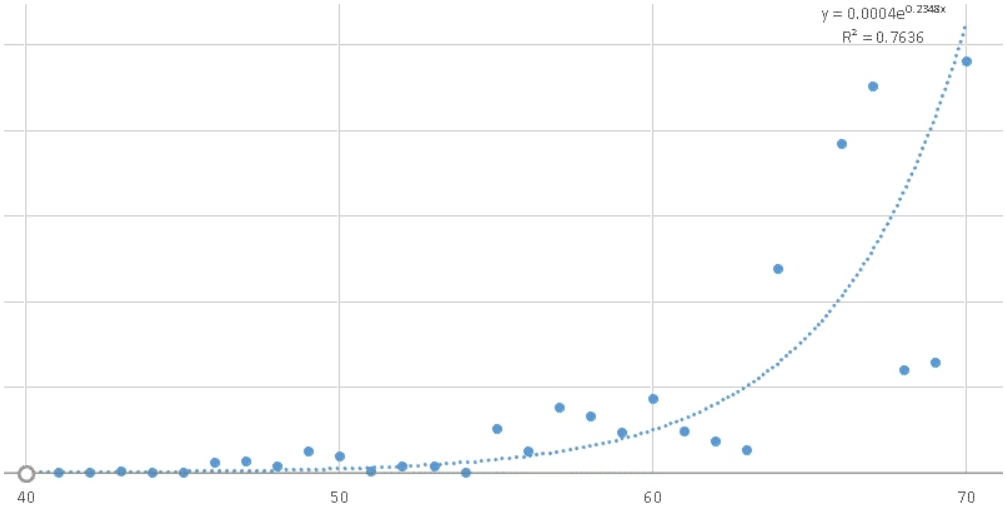


Fig. 15.18. Running time in seconds of our key recovery attack with a SAT solver.

It seems that when the attacker knows a fixed proportion of key bits, the running time of a SAT solver attack, in spite of very big variability, does not grow but it remains constant or grows slowly. This in spite of the fact that the number of rounds grows and the number of unknown secret key bits grows. Our conjecture is that if we fix a certain fixed proportion of key bits, the complexity should then grow slowly (polynomially or with a slow exponential which allows precise estimations). This seems to be confirmed by our computer simulations.

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 number characters $j = 1..$ and put:

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

where P_j/C_j is the plaintext/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 the 13 previously discussed and r_j is a "stepping" output which is derived from the FIRST consecutive 5 bits out of the 13 as follows:

$$r_j = \begin{cases} 0 & \text{if} \quad R_j = (0,0,0,0,0) \\ 0 & \text{if} \quad R_j = (1,1,1,1,1) \\ 31 - r & \text{if } R_j \cdot M^r = (1,1,1,1,1) \end{cases}$$

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

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

16.1 On the Choice of α

The choice of α plays an important role in many attacks in this paper, for example in Section 25.1 page 98 and Table 19 page 78. Important observations about the choice of α in principal historical keys can be found in Section B.3. In almost all historical keys we have $\alpha \in \{1, 2, 3, 4\}$ which is very strange.

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

17.1 Timing and 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**[38] **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[39] of a simple automated software algebraic attack [15, 36, 24, 22, 89]. However of course in historical teletype systems the timing was probably constant, so this remains a theoretical[40] attack.

Remark 1. There exists nowadays formal software methods for automatically synthesizing small size implementations for arbitrary small-size problems such as here, see [25, 102] and Section I.14. 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 [15, 26, 25, 40]. Exact complexity of such attacks will be studied in future updates of this paper. Our first software solver attack was developed by our students cf. Section 15.3 and it provides all the basic functionality of a software algebraic attack except (not as of yet) these advanced optimizations in the line of [25, 90, 102].

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 [77].

17.2 A Known Plaintext Attack on T-310

In Appendix I.5 we show that the first 4 characters (or 20 bits) of the plaintext are always known to the attacker in real life situations.

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

[39] We expect that a SAT solver attack will be suitable, and also an ElimLin-style attack [24, 99, 22] or/and also a correlation attack, cf. Section 15.

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

17.3 The Zero Value Attacks on T-310

The Zero-Value attack is a well-known folklore[41] attack in side channel cryptanalysis [52, 72]. The key vulnerability is nicely summarized in the PhD thesis by Matthieu Rivain [94], 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 the T-310! We recall the encryption formula from Section 16:

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

Theorem 17.3.1 (Zero-Value Vulnerability in T-310 block cipher). If $C_j = 0^5$ on 5 bits, then $P_j = B_j$ regardless of 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 the "double" one-time pad of Section 16 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.

17.4 Bad News - Tentative Applications of Zero-Value Attack

Unhappily, the designers of T-310 did well to make a number of attacks based on thm. 17.3.1 relatively unattractive. Following Section 16, the first bit of B_j comes from a_7, which comes from round as high as $7 \cdot 127 = 889$. Breaking T-310 with bit output after 889 rounds seems ambitious Below we discuss some possible attack scenarios.

KPA Attack? According to Section 17.2, the attacker knows the first 4 characters encrypted. We combine this with Thm. 17.3.1 If we observe a ciphertext character with 5 bits at 0, we know that $P_j = B_j$ on 5 bits. From the first character of the plaintext, which will be LS or 31 in decimal with all bits at 1, cf. Appendix I.5, we deduce that $B_j = 11111$ also. Unhappily again the earliest of these 5 bits is a_7, extracted from round as high as $7 \cdot 127 = 889$.

Cube Attack? However, possibly, 889 relatively simple rounds would not be out of reach for cube attacks [104, 54]. 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 [99]. Here the attacker will fix the plaintext to u_0 and generate equations for several IVs observed in the wild.

[41] It is typically attributed to Golic and Tymen [72, 94, 87] however the attack was known to ourselves and other researchers before, and while Golic and Tymen have developed one specific solution to this problem [72], other very different solutions exist cf. [52, 87].

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. [98]. The "only" problem again, is that 889 is a large number.

Cheating - modifying the Spec of T-310. 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. Such a modified T-310 would be a lot easier to break.

18 Preliminary Analysis for Correlation Attacks and the 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 the bit known as bit I in standard Baudot-Murray ITA-2 code. The question is then what is the probability that this bit is equal to 0 for a long plaintext. In general such probabilities are almost always biased. In the table below we report the exact biases we need based on simulations on 750 Mbytes of German language corpus downloaded from the online archives of Zeit magazine from 1980-2000, cf. www.zeit.de.

Table 6. Statistics for the bias on different bits which occur for German language with 5-bit Baudot-Murray ITA-2 code (upper table) older Baudot 1888 code (middle table) and 8-bit Ascii coding (lower table).

$P(\text{bit I} = 0)$	$P(\text{bit II} = 0)$	$P(\text{bit III} = 0)$	$P(\text{bit IV} = 0)$	$P(\text{bit V} = 0)$
$1/2+2^{-2.32}$	$1/2-2^{-3.67}$	$1/2+2^{-4.06}$	$1/2-2^{-3.89}$	$1/2+2^{-2.27}$

$P(\text{bit I} = 0)$	$P(\text{bit II} = 0)$	$P(\text{bit III} = 0)$	$P(\text{bit IV} = 0)$	$P(\text{bit V} = 0)$
$1/2 + 2^{-2.82}$	$1/2 - 2^{-3.98}$	$1/2 - 2^{-3.35}$	$1/2 + 2^{-4.19}$	$1/2 + 2^{-3.65}$

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

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 "umlaut" accented characters to as follows: German ü becomes ue, etc, while ß becomes ss.

Remarks: We see that the ITA-2 code for German language is MORE vulnerable to correlation attacks than the original 19th century Baudot code. We also that these biases would be worse for English language instead of German (due to how the ITA-2 code was designed).

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 several different sorts of such attacks. In this Section [and also in Section 22.10] 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 23 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 [105] 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 [97] 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 [2]. 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 of 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 [2] 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 [42] 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 [2], 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 [105] 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 have been verified to give highly accurate results in practice and will be used below for different LZS.

[42] 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 [55]. 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 7. Space shrinking properties: comparison of a regular KT2 key 15 with bijective ϕ, some anomalous keys from [55] and our "Rank Deficient" key 208. In these results we ignore bits $\subseteq$ 1-36 which are never used.

LZS 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	$2^{27.0}$	$2^{27.0}$	$2^{27.0}$	$2^{27.0}$
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	$2^{33.9}$	$2^{32.5}$	$2^{31.2}$	$2^{30.3}$
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	$2^{23.9}$	$2^{18.8}$	$2^{16.1}$	$2^{15.2}$
28	8,3,5,2,4,7,6,9,1	21,8,11,15,5,1,19,24,12, 14,17,6,3,10,26,13,25,22, 9,16,2,18,20,23,7,4,27	$2^{25.1}$	$2^{21.2}$	$2^{18.7}$	$2^{17.8}$
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	$2^{35.1}$	$2^{33.6}$	$2^{32.1}$	$2^{31.6}$

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 [100]. 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[43] occur in a real life attack. Nevertheless we can study how the space shrinks also in this[44] case.

[43] If it does occur we would probably be able to exploit its periodic structure in various slide or self-similarity attacks [31, 40].

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

In Table 8 below we present some results for ϕ_0^K space size and key 208 as defined in Table 28 and using the fast method of Section 18.4. On the left in Table 8 we have all bits at 0, and on the right we look at random sequences 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 8. 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 $s \in \{0-7\}^K$

LZS 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	$2^{31.2}$

LZS 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 do 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 9. New key 308 based on class KT2b except for matrix rank condition M_9.

LZS 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 10. Space shrinking comparison of keys 208 and 308 with ϕ_s^K, random s.

LZS nb	K	output space	LZS nb	K	output space
208	0	$2^{36.0}$	308	0	$2^{36.0}$
208	4	$2^{33.8}$	308	4	$2^{33.7}$
208	16	$2^{32.2}$	308	16	$2^{31.9}$
208	64	$2^{30.2}$	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. [55], 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:

$(P, D) \in KT3d \Leftrightarrow$ all of 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 the 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 8. In Table 11 we provide several examples of keys of type KT3d. For comparison purposes we also include key 16 of [55] which according to [100] 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 [55] which is of type KT2.

Table 11. Examples of keys of type KT3d and their space shrinking properties

LZS 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	$2^{30.3}$	$2^{23.6}$	$2^{21.0}$	$2^{20.0}$
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 the 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 $\mathbb{F}_2^{36} \rightarrow \mathbb{F}_2^{36}$ 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 11 and in Table 28.

Table 12. Simulations for ϕ^{16} which shows the average bias in absolute value for the U_α bit of cipher state after 16 rounds and with 32 key bits known, 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($\alpha \in \{1-36\}$,keys)
206	$2^{32.9}$	6	$2^{-16.6}$	$2^{-17.9}$
208	$2^{32.2}$	10	$2^{-15.2}$	$2^{-17.4}$
407	$2^{30.5}$	15	$2^{-14.4}$	$2^{-15.4}$
207	$2^{30.2}$	31	$2^{-14.3}$	$2^{-15.8}$
17	$2^{31.2}$	32	$2^{-15.6}$	$2^{-16.4}$
27	$2^{16.1}$	9	$2^{-8.4}$	$2^{-9.3}$
28	$2^{18.7}$	17	$2^{-9.4}$	$2^{-10.6}$
934	$2^{24.3}$	23	$2^{-10.5}$	$2^{-12.9}$
930	$2^{21.0}$	23	$2^{-11.5}$	$2^{-12.1}$
912	$2^{18.2}$	1	$2^{-10.6}$	$2^{-10.7}$
911	$2^{18.4}$	29	$2^{-9.8}$	$2^{-11.0}$
925	$2^{30.6}$	25	$2^{-5.4}$	$2^{-8.4}$

We observe that the bias is quite substantial for **any** value of α and for any weak key studied. Moreover in many cases we observed that it follows a simple law $\mathcal{O}(\sqrt{1/M})$ which is what we would expect for a random function with M possible outputs. This is except for key 925 which is an outlier and a weaker key for which the bias substantially worse than $\sqrt{1/M}$.

Note: Non-bijective LZS keys can be further substantially WEAKER than exhibiting just some biases on the internal state bits when key bits for one or several rounds are known. These state bits could be for example correlated to some secret key bits, see Table 25 page 92 and Section 22.14.

18.10 Application to SKS V/1

The same weak-key vulnerabilities and resulting biases on individual key bits can be observed in the SKS cipher. To see this we have against attempted to generate various T-310 keys by mutations while at the same time restricting them to satisfy the following all the conditions of class KS0 or KS1 which are the restrictions we defined in Section F.4 in order to design long-term keys for T-310 which could make it operate like an SKS cipher. With these restrictions wee have evolved some new special keys which are similar to LZS-16 except not bijective as follows:

Table 13. Examples of weak LZS keys for SKS emulation with T-310

LZS nb	D	P	$\lvert\phi_0^8(\{0,1\}^{27}\rvert$	$\lvert\phi_0^{16}(\{0,1\}^{27}\rvert$	$\lvert\phi_0^{24}(\{0,1\}^{27}\rvert$
161	0,19,35,27,23,11,3,15,31	23,19,33,18,14,15,5,34,9, 2,6,7,30,11,21,3,22,25,17, 1,32,10,27,29,26,35,13	$2^{23.1}$	$2^{21.8}$	$2^{21.4}$
162	0,3,23,19,11,7,35,15,31	6,34,33,18,23,15,5,14,9, 2,19,1,13,31,21,3,22,25,17, 7,12,10,27,29,26,35,30	$2^{23.5}$	$2^{22.6}$	$2^{22.1}$

For these keys we look at biases of different bits after 16 rounds:

Table 14. Average bias in absolute value for the U_α bit of SKS cipher state after 8 rounds and with 16 key bits known, for one well-chosen α_{best}, and averaged over $\alpha \in \{1-36\}$ and $\alpha \neq 4k$ over many keys.

LZS	M_{ϕ^8}	α_{best}	$\lvert P(U_{\alpha_{best}}=0)-1/2\rvert$	average($\alpha \in \{1-36\}$,keys)
161	$2^{23.1}$	9	$2^{-11.7}$	$2^{-13.6}$
162	$2^{23.5}$	18	$2^{-11.9}$	$2^{-13.4}$

These results also confirm the $\mathcal{O}(\sqrt{1/M})$ rule.

In Section 20.3 we will show how to use these properties in a ciphertext-only attack on SKS V/1 with weak keys.

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 [96] because this property is NOT required in the normal operation of the cipher, see Section 11.1. Yet it is more or less clearly stated inside page 56 [100]. Now we have several interesting 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 have non-bijective ϕ? Is there a really fast attack significantly faster than 2^{240}?
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[45] 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. [70, 88].

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 of KT2 [as specified in pages 59-60,114-115 and 117 in [100] and also transcribed fully in Appendix D.1 on page 136 of the present paper], except just one, the very last rank condition about the matrix B of page 60 of [100], which we detail in Appendix D.4 page 137.

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. We definitely should expect that these properties will be checked by government employees in charge with approval of long-term keys.

[45] For sure the designers of T-310 knew about such attacks cf. [100, 58] 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 23 and another in Appendix G.

However the original documents do NOT mention if KT2 have been proven to be bijective [100] and if they are actually necessary for the cipher to be secure. Even today cryptologists can have serious doubts if it is useful for all these properties to hold, which properties are really required, and which properties potentially just do not matter. Our current understanding is that not all properties are equally important, see Appendix D and Thm. D.6.1 in particular, which definitely does not require all the properties mandated for KT2 keys to hold.

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 [100]. 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 138. Moreover this condition requires a computer simulation and just **cannot** [46] **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 could possibly be disregarded[47] in real-life situations.

[46] 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 the 1980s would not have access neither to any computers of any sort, nor to computer algebra software we take for granted today.

[47] 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[48] 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[49]. It combines four major vulnerabilities of T-310 we have previously uncovered: the Zero-Value attack of Section 17.3, the plausibility of a weak key being used in the real life studied in Section 19, the correlations of Table 12, and the plaintext statistics of Table 6. 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[50] 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 [92]. Then attacks on GOST have literally exploded, cf. [40, 44] for pointers to abundant 200+ pages long recent research paper on this topic which also contains pointers to other papers by the same and other authors.

Initially we thought that, in contrast with GOST, nothing could go wrong with a perfectly periodic key schedule in T-310, because the strongly aperiodic character of the IV handling in T-310 which makes that we do **never** obtain identical permutations for a very large number of rounds. We were wrong as we are going to see below. This is due to a new attack scenario which we have not anticipated.

---

[48] 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.

[49] 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.

[50] Such as many other ciphers which were badly broken in the past cf. for example [12, 40, 44, 31–33] and this would a certainly be a good reason for a cipher to be rejected as a candidate for an encryption standard, cf. [12, 40, 44].

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 12 and biases on the plaintext due to Table 6 and Thm. 17.3.1 in order to decrypt T-310 communications in the ciphertext-only scenario.

1. We apply the Zero-Value attack and we exploit a proportion of 2^{-5} of the available ciphertext data. We discard other ciphertext data.
2. We recall from Section 17.3 that if $C_j = 0^5$ we have $P_j = B_j$.
3. We can now express certain, but not all, bits of the plaintext as a function of the internal state bits as
$$P_{j,I-V} = B_{j,0-5}$$
which equation holds for **all** ciphertext characters $C_j = 0$ we selected.
4. We can then 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 12.
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 the 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 12, we need to know the 32 key bits for 16 rounds preceding the actual bit extracted which are $u_{m,\alpha}$ with $m = 127(B + 13(j-1))$ with five possible $B \in \{7 - 11\}$.
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 of key and any 16 bit IV for ϕ_s^{16}. This table requires only 1 Terabyte of storage (2^{48} bits).
9. We have a window of 24 rounds where the key bits are known and it is repeated with a period of 120 rounds. We consider that positions of type $m = 127(B + 13(j-1))$ span the interval $0 - 119$ uniformly at random. We are interested in positions where key bits are known for at least 16 rounds before m, i.e. the window $m - 15, \ldots, m$ must fall within our window of 24 rounds. The probability of this is $(24 - 16)/120 \approx 2^{-3.9}$.
10. Accordingly, 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 known to the attacker, is equal to $2^{-3.9}$.
11. For simplicity, we will only work on plaintext bits I and V in upper part of Table 6 which have a bias of approximately $\pm 2^{-2.3}$. We need to pay attention to the signs; let $\sigma_I = +1$ and $\sigma_{III} = -1$ for these two bits.
12. 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.

13. 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 12, 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.
14. 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 attacker 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".
15. We apply Matsui's piling-up lemma [82] and we see that the overall bias for our bits which are $(1 + \sigma_I\sigma_{K,IV})/2$ or $(1 + \sigma_V\sigma_{K,IV})/2$ is going to be equal to $\gamma = 2^{-2.3}\beta$.
16. In order to distinguish these biased distributions and have results which is stronger than 8 standard deviations we need to generate about $8^2\gamma^{-2} \approx 2^{16+4.6}\beta^{-2}$ biased bits in "the $B - I$ set".
17. We need to work with 8 standard deviations exactly: we apply the Gauss Error function cf. [45, 83] which leads to a probability of $2^{-49.5}$ of a false positive which is sufficient to confirm if our 48-bit key is correct.
18. 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 $2^{-3.9}$.
19. Therefore we need overall $2^{16+4.6+3.9+5}\beta^{-2}/2 \approx 2^{28.5}\beta^{-2}$ of encrypted characters in order to recover 48 bits of the key in time which is approximately $2^{48+28.5-5-3.9}\beta^{-2} \approx 2^{68}\beta^{-2}$. Here $-5 - 3.9$ comes from the fact the we can pre-select ciphertext bytes and m values for the attack independently of the key depending on the window position.
20. 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 24 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.

Application using key 206: With our "Rank-Deficient" key 206, we have $\beta = 2^{-15}$ and the attacker can recover the full 230-bit encryption key in a time of 2^{98} given about 2^{59} characters of encrypted data in ciphertext-only scenario.

Application using key 27: With the original key 27 from [55], we have $\beta = 2^{-8.0}$ and we can recover the full 240-bit encryption key in a time of 2^{84} given about 2^{45} characters of encrypted data in ciphertext-only scenario.

Note - Generalized Non-Uniform Attacks: This attack can be further optimized in several ways. A better attack will no longer be uniform where different sorts of events are counted in the same way. If we dispose of more data we can start with the number of key bits guessed smaller than 48 bits. If we dispose of less data, we need to guess more bits and time complexity will increase. In addition, for many locations we know the key for more than 16 rounds before, up to 24 rounds. Therefore the attack could have additional prefiltering steps based on stronger biases obtained for (smaller) subset of locations where the bias is stronger. Finally we could use all the 5 bits of the ciphertext.

20.3 A Ciphertext-Only Correlation Attack on SKS V/1

We refer to Section F and also Section F.1 for a detailed description of SKS V/1 and to see how it differs from T-310. It is possible to see that given a weak long-term key for[51] SKS V/1, we can mount a ciphertext-only attack which recovers the secret key in a way which is simpler than for T-310. We do not need any "Zero-Value" vulnerability such as in Thm. 17.3.1 and we do need to only consider parts of ciphertext with $C_j = 0$. For example, we can cryptanalyse T-310 with key 162 which emulates the SKS V/1 cipher as follows.

- We assume that we have a ciphertext in German language, encoded on 5 cf. Table 6 page 56. This implies that some 16/41 of encrypted plaintext bits are equal to 0 with probability $1/2 \pm 2^{-3}$ and they occur at positions which are known to the attacker, and the signs of these biases are known.
- For key 162, each of these bits is XORed with a bit with a bias equal to 2^{-13} approximately, cf. Table 14 page 64. This gives a combined bias of $\varepsilon = 2^{-16}$ for a percentage of 16/41 keystream bits W_i if we apply Matsui's piling-up Lemma [82].
- We guess 16=8+8 key bits in the 16 rounds preceding the round at which the key bits are extracted, which is round $p - 16 \bmod 104$ and is the same every 104 rounds, see Section F. Moreover the key bits are the same for every bit and for every 104 rounds, and all windows with key bits used are aligned exactly and we can exploit a guess of 16 key bits to the full extent by predicting biases on the bits after 8 rounds which are always perfectly aligned, this is due to the fact that in SKS V/1 the periodic structure which repeats every 104 rounds is such that keystream bits are always extracted simply at the end of each block of 104 rounds. cf. Section F.
- Again, as in Section 20.2 in order to confirm 16 key bits we need to see if our statistics is outside of the interval of 4 standard deviations. This is due to the Gauss error function, we need less than one false positive in 2^{16} experiments, $1 - erf(z/sqrt2) < 2^{-16}$, which leads to $z \approx 4$, cf. Table in [106]. This leads to data complexity of about $k = 41/16 \cdot 4^2 \cdot \varepsilon^{-2}$ with $\varepsilon = 2^{-16}$ encrypted ciphertext bits. This is approximately $k \approx 2^{37}$ bits.
- For each key on 16 bits we need to go examine these $k \approx 2^{37}$ bits. The time complexity is approximately $T \approx 2^{37+16} = 2^{51}$.
- Once we have the last 16 key bits used in last 8 rounds out of 104, we can then guess few more key bits, predict a slightly stronger bias will occur in the attack and recover more key bits with lower data complexity. The complexity of these additional steps is expected to be substantially less than the first main step with 16 key bits and 2^{37} bits of ciphertext, therefore we ignore these extra steps.

Overall we see that with a faulty LZS for SKS V/1 cipher machine such as 162, the attacker can recover the full 208-bit key in a time of about 2^{51} given about 2^{37} bits of encrypted data.

[51] which in this paper [for simplicity] are replaced by weak-long-term keys for T-310 emulating SKS V/1, cf. Section F.4.

21 T-310 and Linear Cryptanalysis

In this section we study the resistance of T-310 against Linear Cryptanalysis (LC).

21.1 Historical Background

There is no doubt that the topic of Linear Cryptanalysis (LC) was somewhat known and studied by Cold War cryptography specialists. It was known under the name of "Statistische Struktur" cf. page 30 of [61]. On pages 17-18 of [63] we find a table which gives a complete set of linear characteristics of the non-linear component of T-310.

Tabelle 3.1-2

α	Δ_α^Z	t	α	Δ_α^Z	t
000000	32 0	32	L00000	0	32
00000L	2	34	L0000L	6	38
0000L0	-4	28	L000L0	0	32
0000LL	6	38	L000LL	6	38
000L00	-4	28	L00L00	-4	28
000L0L	-2	30	L00L0L	2	34
000LL0	0	32	L00LL0	4	36
000LLL	2	34	L00LLL	2	34

Fig. 21.19. Fragment of Table 3.1-2 in page 18 of [63] dated 1976 which contains a complete set linear characteristics of Z.

The document [63] suggests that the study of resistance of a cipher component against linear and differential cryptanalysis was a routine task in Eastern block cryptography in the 1970s. A precise definition is provided on page 17 of [63]. The table a fragment of which we reproduce above contains all the possible values Δ_α^Z for any $\alpha \in \{0,1\}^6$ and which is defined exactly as:

$$\Delta_\alpha^Z = 2^{6-1} - ||Z(x) - \sum_{i=1}^{6} \alpha_i x_i||$$

where $||g(x)||$ is the number of times $g(x) = 1$. We can also remark that

$$\Delta_\alpha^Z = t - 2^{6-1}$$

where t is the number of times $g(x) = 0$ while we have $g(x) = Z(x) - \sum_{i=1}^{6} \alpha_i x_i$.

Moreover it also very clearly indicated that this definition comes from an earlier Eastern block written source, namely Section 2 inside Chapter 2 "Boolean Functions" from a classified course or lecture notes on cryptography which apparently was delivered by Soviet specialists, known under reference number 2243, and is not dated.

On Modern Notion of Non-Linearity. From here standard cryptographic literature would define the nonlinearity of the Boolean function Z as the Hamming distance from the set of all affine functions. The earliest reference in the open academic literature which contains this definition is Pieprzyk and Finkelstein, cf. Def. 7 page 326 in [91] from 1987/88.

21.2 Application to Several Rounds of T-310

In this section we give some linear characteristics of T-310. They allow the reader to see that the resistance of T-310 against Linear Cryptanalysis (LC) [82] for various long-term keys. The last column contains a rough estimation of the probability that the exact propagation probability/bias we report in the previous column holds for a random key and IV pair.

Table 15. Some linear characteristics for T-310

LZS nb	rounds	input → output	bias	proba
14	5	[17] → [34]	$2^{-1.0}$	1.0
14	10	[17] → [27,31]	$2^{-1.0}$	1.0
14	12	[19] → [23,31,33]	$2^{-2.6}$	0.1
14	12	[19] → [23,31,33]	$2^{-3.0}$	0.4
14	15	[17] → [24,32,34]	$2^{-3.00}$	1.0
15	6	[17] → [35]	$2^{-1.00}$	1.0
21	5	[30] → [35]	$2^{-1.00}$	1.0
21	8	[2,18] → [34]	$2^{-2.97}$	0.8
26	8	[3,31,35] → [27,31,35]	$2^{-1.00}$	1.0
26	8	[4] → [28]	$2^{-8.0}$	0.2
26	8	[1] → [25]	$2^{-8.5}$	0.1
30	5	[14] → [35]	$2^{-1.00}$	1.0
30	6	[13] → [35]	$2^{-1.00}$	1.0
30	8	[4,12,32] → [12,16,32]	$2^{-3.00}$	0.4
31	7	[17] → [36]	$2^{-1.00}$	0.3
31	10	[19] → [25,29,35]	$2^{-3.00}$	0.3
31	12	[17] → [25,29,35]	$2^{-3.00}$	0.3
31	12	[5,13,25,29] → [5,13,25,29]	$2^{-8.3}$	0.5
31	24	[5,13,25,29] → [5,13,25,29]	$2^{-15.5}$	≥ 0.2
33	12	[2,7] → [12,19,31]	$2^{-5.0}$	TBC

We see that the resistance of T-310 against LC depends strongly on the long-term key and some historical keys are stronger than other historical keys. For example we see in Table 15 that very clearly (earlier) key 14 is substantially less secure against LC compared to (later) key 31, while both keys are real-life keys cf. [55].

21.3 Invariant Linear Characteristics for T-310

Some linear characteristics are particularly interesting: this is when the set of bits used at the input and at the output are the same. Such characteristics can be joined with themselves and lead to periodic linear properties which propagate for an arbitrary number of rounds. We give some examples below.

Table 16. Some invariant linear characteristics for T-310

LZS nb	rounds	input → output	bias	proba
31	12	[5,13,25,29] → [5,13,25,29]	$2^{-8.3}$	0.5
31	24	[5,13,25,29] → [5,13,25,29]	$2^{-15.5}$	≥ 0.2
606	1	[1-6] → [1-6]	$2^{-1.00}$	1.0
607	1	[1-7] → [1-7]	$2^{-1.00}$	1.0
702	8	[17,21] → [17,21]	$2^{-1.00}$	1.0
704	4	[17,21,33] → [17,21,33]	$2^{-1.00}$	1.0
703	2	[1,3,5] → [1,3,5]	$2^{-1.00}$	1.0
783	2	[1,3,5] → [1,3,5]	$2^{-1.00}$	1.0
784	8	[12,16] → [12,16]	$2^{-1.00}$	1.0
785	4	[9,13,25,29,33] → [9,13,25,29,33]	$2^{-1.00}$	1.0

In Table 16 only key 31 is an actual historical key. All other keys are weak LZS which we have created and which are listed below. For these keys, the linear properties are particulary strong. Moreover in Section 21.14 we show that some of these keys have several such properties.

21.4 Generating Very Weak Long Term Keys for LC

The keys such as 606 or 784 can be seen as long-term LZS keys which are somewhat backdoored: they have invariant linear characteristics true with probability 1 which can therefore propagate for an arbitrary number of rounds. We have the following definition:

Definition 21.4.1 (LC-weak keys).
We say that a long-term key LZS is **LC-weak** if it exhibits at least one invariant linear characteristics true with probability 1.

Such keys are extremely weak w.r.t. Linear Cryptanalysis and the invariant property can be extended for an arbitrary number of rounds. Here are several examples of such keys:

```
606: P=32,7,33,30,12,36,5,4,9,27,19,2,16,11,21,1,8,25,20,28,
24,23,35,29,15,31,6 D=0,4,32,12,24,8,16,36,20
607: P=28,8,33,22,16,36,5,2,9,15,1,24,32,10,21,23,34,25,35,
12,26,4,18,29,31,20,7 D=0,4,16,28,8,20,32,36,24
702: P=22,24,33,32,14,4,5,28,9,11,27,18,36,16,21,15,20,25,35,
8,1,6,23,29,19,12,13 D=12,16,0,36,28,32,24,4,20
703: P=7,14,33,23,18,36,5,2,9,16,30,12,32,26,21,1,13,25,20,8,
24,15,22,29,10,28,6 D=0,4,24,12,16,32,28,36,20
704: P=24,34,33,32,14,4,5,28,9,26,27,18,36,16,21,15,20,25,35,
8,1,6,23,29,19,12,13 D=28,16,12,36,24,0,32,4,20
782: P=30,3,33,32,35,24,5,17,9,14,6,16,12,8,21,18,20,25,23,36,
15,28,4,29,7,11,1 D=0,4,32,28,8,20,12,24,16
```

```
783: P=8,32,33,11,1,20,5,26,9,24,4,7,12,2,21,34,28,25,3,36,31,
13,18,29,19,16,6 D=0,4,8,32,28,16,12,20,24
784: P=3,1,33,11,32,20,5,26,9,2,4,7,12,24,21,34,31,25,8,
36,28,13,18,29,19,16,6 D=0,4,32,28,24,8,12,20,16
785: P=28,12,33,20,27,24,5,32,9,4,26,22,36,10,21,2,11,25,8,16,
19,13,30,29,17,23,31 D=0,4,28,12,8,20,36,24,32
787: P=3,24,33,27,36,20,5,4,9,7,30,13,12,16,21,34,8,25,23,28,
17,1,26,29,19,32,6 D=0,4,36,32,24,8,12,20,16
788: P=26,19,33,36,4,20,5,27,9,17,2,11,12,31,21,22,1,25,7,
28,16,24,32,29,8,30,34 D=0,4,36,32,24,8,12,20,16
789: P=36,19,33,15,23,20,5,22,9,4,11,3,12,16,21,24,8,25,26,
28,17,35,32,29,27,2,7 D=0,4,36,32,24,8,12,20,16
```

These examples were generated by trial and error and linear algebra [testing for linear equations true with probability 1] or by a dedicated "imitation" technique described in Section 21.10 below.

21.5 LC-Weak Keys with One-Bit Correlations

A special case is keys with correlations with HW=1 suitable for slide attacks of Section 24 specifically. Such keys are more rare but they still do exist as we are going to show now. These keys have been found by software simulations by Marios Georgiou, Matteo Scarlata and Nicolas Courtois. Keys 741 and 729 are of type KT1. Other keys are not guaranteed to have a bijective round function.

```
701: P=31,10,33,6,32,8,5,3,9,15,13,26,19,28,21,7,16,25,34,12,
22,17,35,29,30,23,4 D=4,2,17,32,12,35,0,24,20
741: P=15,24,33,27,19,12,5,22,9,31,3,7,8,34,21,36,32,25,18,28,
35,20,4,29,16,14,2 D=0,36,24,4,32,16,8,12,20
625: P=7,32,33,30,22,20,5,18,9,34,35,31,36,28,21,24,27,25,26,
16,4,23,19,29,8,12,11 D=0,32,24,8,12,28,36,20,4
729: P=7,23,33,16,31,4,5,1,9,12,14,13,36,8,21,3,24,25,32,20,
2,6,30,29,28,26,18 D=0,12,16,28,8,32,36,4,24
517: P=27,4,33,16,23,31,17,21,18,14,10,29,3,25,1,5,22,9,7,19,
32,28,20,2,30,36,24 D=36,4,20,16,0,32,24,8,28,
533: P=11,4,27,33,36,3,17,34,29,15,23,31,32,8,26,20,28,35,30,
5,14,2,25,13,12,10,16 D=20,0,28,16,12,24,8,4,32,
433: P=27,5,11,35,31,15,7,34,17,33,29,25,32,22,28,8,6,10,12,
36,14,9,23,21,24,30,16 D=24,0,28,16,8,12,4,32,36
217: P=16,10,33,32,22,18,27,24,13,6,20,8,9,12,25,23,29,17,26,
28,15,21,31,30,1,11,35 D=16,12,0,28,32,4,24,20,8
221: P=35,19,13,3,27,22,28,12,24,23,32,9,30,33,31,14,21,36,26,
4,10,17,11,8,1,15,18 D=24,28,0,4,8,32,36,16,12
```

Table 17. Some one-bit invariant correlations in T-310

LZS nb	rounds	input → output	bias	prop./keys
701	7	[30] → [30]	$2^{-11.2}$	0.2
701	7	[36] → [36]	$2^{-8.7}$	TBC
741	7	[29] → [29]	$2^{-4.4}$	1.0
741	7	[30] → [30]	$2^{-4.4}$	1.0
729	7	[29] → [29]	$2^{-6.0}$	TBC
625	6	[25] → [25]	$2^{-5.4}$	TBC
517	5	[17] → [17]	$2^{-10.5}$	TBC
533	5	[33] → [33]	$2^{-5.7}$	TBC
433	4	[33] → [33]	$2^{-1.0}$	1.0
217	2	[17] → [17]	$2^{-6.0}$	TBC
221	2	[21] → [21]	$2^{-5.3}$	TBC
27	7	[30] → [30]	$2^{-6.7}$	TBC
14	24	[11] → [11]	$2^{-17.8}$	TBC

These few results (keys we have discovered or constructed) can be compared to a result which we discovered accidentally for one "faulty" historical key 27 from [55] and a weaker result obtained also accidentally for a regular KT1 historical key 14. Further keys of this type can be found in Section 22.6.

21.6 A Short Explanation for Key 741

Here is a short explanation why key 741 has the properties stated above. First we observe that [29] → [30] → [31] → [32]. Then we are going to show that:

Table 18. A Detailed Explanation for key 741

LZS nb	rounds	input → output	bias	prop./keys
741	3	[29] → [32]	$2^{-1.0}$	1.0
741	1	[32] → [10,17,25]	$2^{-3.0}$	1.0
741	3	[10,17,25] → [29]	$2^{-2.4}$	1.0

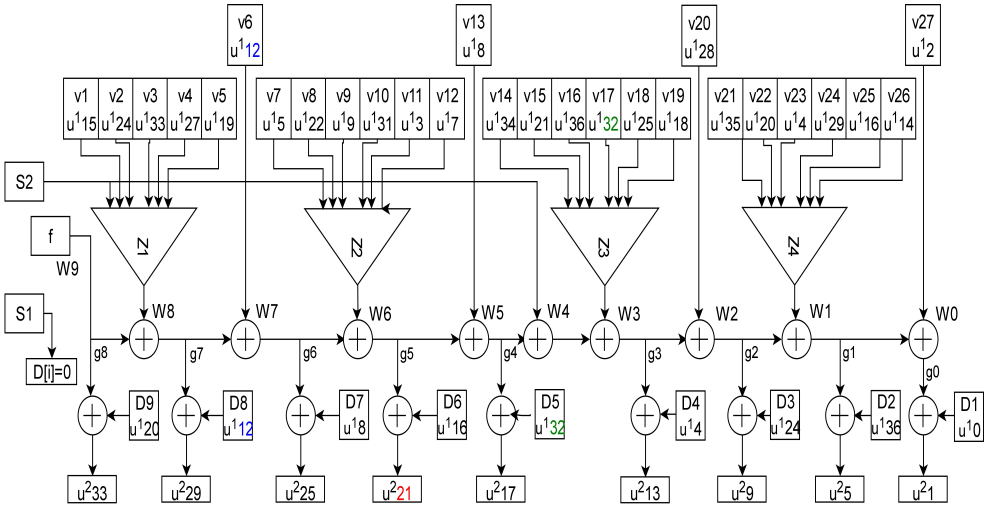


Fig. 21.20. One round of T-310 for key 741.

1. Let $X^{(j)}$ denote values inside round j.
2. We observe that $P(13) = D(7) = 8$ therefore $v13 = u_8^{(1)} = u_{D(7)}$. Then $D(5) = 32$. We have:

$$u_{25}^{(2)} \oplus u_8^{(1)} \oplus Z2^{(1)}(v7 - v12) \oplus u_8^{(1)} \oplus u_{32}^{(1)} = u_{17}^{(2)}$$

here $u_8^{(1)}$ appears twice and is eliminated, and we have

$$Z2^{(1)}(v7 - v12) = u_{25}^{(2)} \oplus u_{32}^{(1)} \oplus u_{17}^{(2)}$$

now we add $u_9^{(1)}$ on both sides

$$Z2^{(1)}(v7 - v12) \oplus u_9^{(1)} = u_{25}^{(2)} \oplus u_{32}^{(1)} \oplus u_{17}^{(2)} \oplus u_9^{(1)}$$

and observe that this bit becomes number 10 in the next round:

$$Z2^{(1)}(v7 - v12) \oplus u_9^{(1)} = u_{25}^{(2)} \oplus u_{32}^{(1)} \oplus u_{17}^{(2)} \oplus u_{10}^{(2)}$$

finally we observe that $Z2$ is correlated to $u_9^{(1)}$ which is one of its inputs, cf. Fig. 21.19. Therefore the following expression is biased:

$$u_{32}^{(1)} \oplus u_{10}^{(2)} \oplus u_{25}^{(2)} \oplus u_{17}^{(2)}$$

3. Thus we have shown that we have $[32] \to [10, 17, 25]$ in one round.
4. It remains to see that $[10, 17, 25] \to [29]$ in 3 rounds.

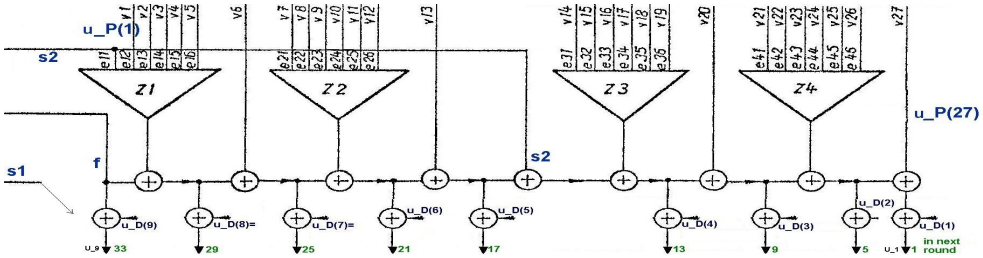


Fig. 21.21. Internal structure of T inside one round of T-310.

5. After first 2 rounds bits $u_{10}^{(1)}, u_{17}^{(1)}, u_{25}^{(1)}$ become $u_{12}^{(3)}, u_{19}^{(3)}, u_{27}^{(3)}$ which are $v6, v5, v4$ inside 3rd round. Furthermore $D(8) = 12 = P(6)$. The output of $Z1^{(3)}$ is correlated to XOR of 2 of its inputs $u_{19}^{(3)} \oplus u_{27}^{(3)}$.
6. Moreover $f^{(3)} \oplus Z1^{(3)} \oplus u_{12}^{(3)} = u_{29}^{(4)}$. Thus we have

$$Z1^{(3)} \oplus u_{19}^{(3)} \oplus u_{27}^{(3)} = u_{29}^{(4)} \oplus f^{(3)}$$

and if the left hand side is biased, the right hand also.

21.7 A Classification of One-Bit Correlations $\alpha \rightarrow \alpha$

Further keys of this type can be found in Section 22.6. We can classify different keys similar to 625 or 741 which exhibit 1 bit invariant linear properties by the order of Boolean functions which they use. For example we can say that key 741 is of type Z2-Z1 and key 625 is also of type Z2-Z1. Then we can look at the set of active bits after round 4. We call Sr4 this set, for example in key 741 we have $Sr4 = [10, 17, 25]$. This analysis leads to an important insight about the security of T-310 long term keys. We have observed that correlations of type say $[29] \rightarrow [29]$ are apparently the easiest to construct, and the Hamming weight of Sr4 will be typically 3. Then constructing correlations of type $[25] \rightarrow [25]$ already leads to the Hamming weight of Sr4 being 4. We conjecture that the smaller is the number α the harder it is to construct correlations for this α. This would suggest that most real-life keys (which happen to have a small $\alpha \leq 4$) are quite secure. This observation was made by UCL student Marios Georgiou and confirmed by hundreds of cases of vulnerable keys we have examined.

21.8 Specific Types of Near-Bit Correlations

Finding 2-bit linear properties in T-310 is not as hard in general as finding 1-bit properties, see for example 768 778 or many other found in Section 21.20 and elsewhere. However the attacker can be specifically interested in properties of type for example $\alpha, \alpha + 1$ or similar. Here we give three examples which are all of type KT1:

```
974: P=12,14,33,10,20,36,5,24,9,15,13,23,4,22,21,2,28,25,17,8,
30,16,32,29,34,1,11 D=0,16,20,32,28,12,4,36,24
937: P=1,20,33,28,32,4,5,7,9,11,22,23,8,27,21,2,17,25,18,36,16,
30,34,29,12,24,3 D=0,24,28,12,16,32,8,4,20
917: P=20,18,33,22,2,36,5,11,9,35,13,14,12,28,21,8,31,25,6,32,
16,1,15,29,24,4,17 D=0,16,20,4,8,28,12,36,24
```

Table 19. Some near-bit invariant correlations in T-310

LZS nb	rounds	input → output	bias	prop./keys
974	5	[33,36] → [33,36]	$2^{-7.4}$	TBC
937	2	[21,24] → [21,24]	$2^{-5.4}$	TBC
917	2	[1,2,4] → [1,2,4]	$2^{-13.2}$	TBC
31	5	[21,22,24] → [21,22,24]	$2^{-19.1}$	TBC
32	5	[1,2,4] → [1,2,4]	$2^{-30.7}$	TBC

We see that real-life historical keys are not exempt from such correlations, though they seem quite strong compared to some other KT1 keys. We refer to Section 25.8 to see why such keys matter in slide attacks. In the same way as in a previous section, we observed that finding keys with this sort of correlations, for bits with low numbers like $\alpha = 1$ is substantially harder for bits with higher numbers (like $\alpha = 33$).

21.9 A Detailed Example of How T-310 Can Be Weak w.r.t. LC

In this section we show in detail HOW LC-weak keys work in detail in one specific case. It works for LZS number 703, 783 and 787 as specified in Section 21.4 and which are LC-weak keys exhibiting exactly the same property as explained below.

We recall the last equation of Section 9.6:

$$u_{m+1,1} = u_{D(1)} \oplus u_{m+1,5} \oplus u_{D(2)} \oplus u_{P(27)}$$

We have $D(1) = 0$ which makes that $u_{m,0} = s_{m+1,1}$ and $D(2) = 4$ and $P(27) = 6$. Therefore we have

$$u_{m+1,1} = s_{m,1} \oplus u_{m+1,5} \oplus u_{m,4} \oplus u_{m,6},$$

and this leads to the following linear approximation for one round:

$$[4, 6] \to [1, 5] \quad 1R \quad P = 1$$

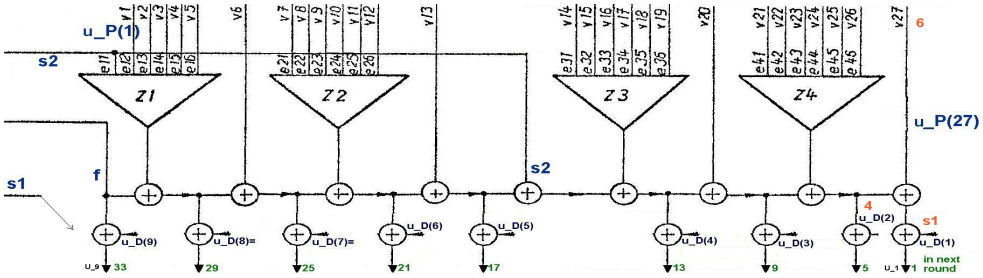


Fig. 21.22. For convenience we show the bits involved here, cf. also Fig. 7.11.

Using that fact that bits $\neq 4k$ are just shifted in our Feistel with 4 branches, this can be trivially extended for one earlier round as follows:

$$[3, 5] \to [1, 5] \quad 2R \quad P = 1$$

Finally it is trivial to see that $[1] \to [2] \to [3]$ for two rounds also with certainty, which property can be combined with the previous one and we obtain finally:

$$[1, 3, 5] \to [1, 3, 5] \quad 2R \quad P = 1$$

It follows that the same linear characteristic works for any even number of rounds. It is also easy to see that

$$[2, 4, 6] \to [2, 4, 6] \quad 2R \quad P = 1$$

21.10 Can More KT1 Keys be Pathological?

From here we are tempted to generate a regular KT1 key such that $D(2) = 4$ and $P(27) = 6$ in order to obtain a linear approximation true with probability 1. A quick examination of the KT1 rules cf. Appendix B suggests that this should be possible if $j_1 = 2$. It is however a non-trivial task to generate a KT1 key with or without this extra property. A plausible method to find such a key is to first generate one very weak key, then investigate what properties make this key weak, and finally find a key of type KT1 which also has these properties. This is exactly what we did. Below we detail all the steps performed.

1. First we generated about a million of totally random keys and selected several which had linear approximations true with probability 1 for 8 rounds.
2. Inside those, we found one key called 703 where this [accidentally] worked also for 2 rounds.
3. Then we investigated why this property is true, cf. Section 21.9 and found that this was because $D(1) = 0$, $D(2) = 4$ and $P(27) = 6$ and that these conditions are sufficient conditions.
4. Then we examined the KT1 rules cf. Appendix B and found that $D(1) = 0$ is always satisfied, and $D(2) = 4$ should be possible if $j_1 = 2$ and also nothing seems to prevent $P(27) = 6$ given that $6 \neq 0 \bmod 4$.

At this stage we are able to propose an efficient algorithm method for generating the KT1 keys which we need.

21.11 Generation of KT1 Keys [General or LC-Weak]

Here is a simple method for generating weak KT1 keys.

1. We generate about a million of permutations $\{j_1, j_2, \ldots, j_7, j_8\}$ of the set $\{2, 3, \ldots, 9\}$ with an added condition $j_1 = 2$.
2. For each permutation we observe that $D()$ is defined uniquely following the rules of Appendix B. We recover $D()$ and check if $(D(5), D(6)) \in \{8, 12, 16\} \times \{20, 28, 32\} \cup \{24, 28, 32\} \times \{8, 12, 16\}$, if not OK, we restart.
3. Now we try to generate suitable $P()$ initially ensuring the following conditions $P(3) = 33, P(7) = 5, P(9) = 9, P(15) = 21, P(18) = 25, P(24) = 29$ and $P(20) = 4j_8$ and $P(6) = D(8), P(13) = D(7)$, and $\forall_{1 \geq l \geq 9} \exists_{1 \geq i \geq 26} P(i) = 4 \cdot l$ and $D(3) \in \{P(1), P(2), P(4), P(5)\}$.
4. For all of the other conditions we counted on luck: we restart generation of j_i's until a valid KT1 key is found.

A ready software tool for generating KT1 keys (with two different methods implemented) can be found in Appendix 21.23.

After generating KT1 keys as above, we check if they are weak. A basic version of this method was used to generate our special key 783. It satisfies all the conditions in KT1. All these keys are specified in Section 21.4.

Further weak keys. By a similar method and with more patience we also generated keys 606 784 and 787 and few other which also are both of type KT1 and have invariant linear characteristics true with probability 1, cf. Table 16 above.

21.12 More Complex Periodic Properties

We have observed that various KT1 keys exhibit various invariant linear patterns with periods of 1,2,3,4,6 and 8 rounds. Here are two examples where the period is equal to 3 and 6 with full internal details of their primary/main linear approximations for 3 and 6 rounds respectively:

```
733: P=28,23,33,36,3,24,5,11,9,12,35,1,4,20,21,22,16,25,27,
32,10,8,15,29,31,34,13 D=0,12,28,36,16,20,4,24,8,
[3,7,9,12-13,26,30,34]->[1,4-5,8,10,27,31,35]-s1->
      [2,6,11,25,28-29,32-33,36]-f->[3,7,9,12-13,26,30,34]
```

In the second example we see that even though native properties for 3 rounds exist, twice more properties exist for 6 rounds.

```
706: P=8,2,33,4,13,20,5,14,9,22,30,31,16,19,21,32,3,25,28,36,
27,11,23,29,12,24,10 D=0,28,8,4,24,12,16,20,32
[3,7,10,14,25,28-29,32,35]->[1,4-5,8,15,26,30,33,36]-s1f->
      [2,6,9,13,16,27,31,34]->[3,7,10,14,25,28-29,32,35]
[1,5,15,33]-s1f->[2,6,16,34]->[3,7,25,29,35]->[4,8,26,30,36]->
      [9,13,27,31]->[10,14,28,32]->[1,5,15,33]
```

Properties for 7 and 11 rounds can be found in Section 21.16.

21.13 Periodic Properties which Involve Key Bits

In the above examples we see that for keys 706,733, and also keys 703, 783 and 788 studied above, the linear property depends of key bits S1. We can also recall that keys 612 and 636 studied earlier depend on S1.

```
612: [1-7,9-16,25-32]-s1->[1-7,9-16,25-32]
636: [1-7,33-36]-s1f->[1-7,33-36]
```

Here are some additional examples of long-term keys with linear properties which involve S1. It turns out than more than half of all LC-weak keys will use S1. This will be used in a new sliding attack on T-310 with $d = 0$ in notations of Section 24, which was recently submitted to a journal.

```
774: P=17,5,22,2,20,7,11,15,24,33,31,23,10,21,25,29,6,28,27,
12,34,36,1,4,18,19,9 D=8,12,16,0,24,20,36,28,32
[9,13]-s1->[10,14]->[11,15]->[12,16]->[9,13]
```

```
868: P=36,4,33,16,30,28,5,17,9,19,11,23,20,26,21,24,22,25,1,
12,35,8,31,29,32,7,6 D=0,36,16,32,24,8,20,28,4
 [1,3,5,17,21]->[2,4,6,18,22]->[3,7,19,23,33]-f->[4,8,20,24,34]
  ->[17,21,33,35]-f->[18,22,34,36]->[1,5,7,19,23,35]-s1->
   [2,6,8,20,24,36]->[1,3,5,17,21]
```

It remains an open problem if we can find a property which would involve S2. We conjecture that such property does not exist.

21.14 Strongly Pathological LC-Weak Keys

It is possible to see that for some KT1 keys there are more than one linear invariant properties such as in Table 16. Below we give some examples.

Table 20. KT1 keys with multiple invariant linear characteristics for T-310

LZS nb	rounds	solutions
606	4	6
607	4	5
783	2	2
785	4	4
706	6	6
868	8	6
788	8	8
789	8	9
784	8	10
787	8	10

Currently the most pathological KT1 key known is 784 (or 787). This key 784 can be characterized in a very simple way as follows. It exhibits **simultaneously** a very similar type of 8-round periodic linear characteristic as key 702 [which was not KT1], AND exactly the same 2-round periodic linear characteristic as key 783. Moreover all these concern a disjoint set of linear combinations with 10=8+2 total of linear invariant properties which happen to work with probability 1 for all keys and IVs. In addition here are full internal details about these periodic linear properties which are self-explanatory:

`key 784:`

[1,3,5]→[2,4,6]→[1,3,5]
[9,13]→ [10,14]→ [11,15]→ [12,16]→
[25,29,33]f→ [26,30,34]→ [27,31,35]→ [28,32,36]→ [9,13]

A more detailed explanation and a picture for the second property which also uses the same colouring convention can be seen on Fig. 21.23 on the next page and a full explanation can be found in Appendix J.1 (cf. Thm. J.1.1 page 173).

21.15 Weak LZS with 8 Round Properties

We recall the property of key 788 or 784 in Section 21.14 above:

[9,13]→ [10,14]→ [11,15]→ [12,16]→
[25,29,33]f→ [26,30,34]→ [27,31,35]→ [28,32,36]→ [9,13]

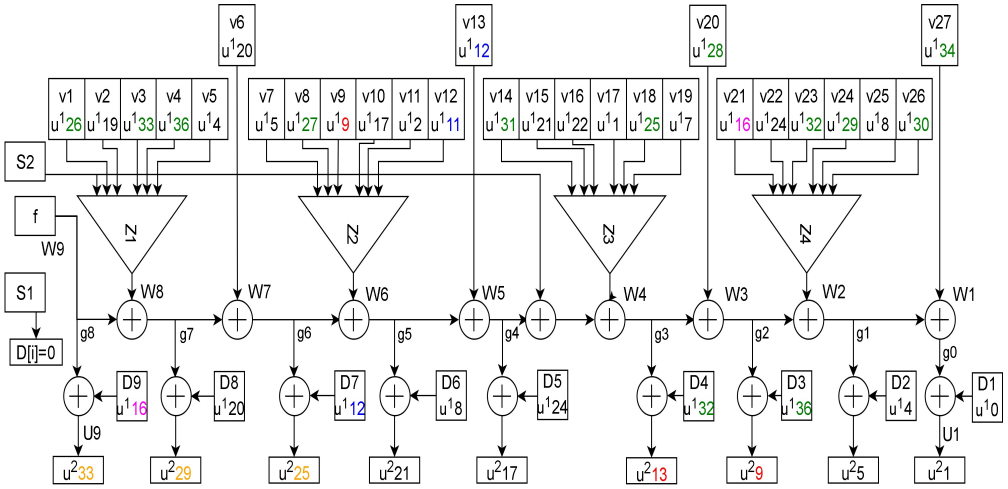


Fig. 21.23. One round of T-310 for key 788.

This is NOT the only property for 8 rounds. There exist more distinct 8R properties for example:

```
key 787:
[9,13]->[10,14]->[11,15]->[12,16]->[25,29,33]-f->
                    [26,30,34]->[27,31,35]->[28,32,36]->[9,13]
```

Another key with a non-trivial property for 8 rounds is the previously defined 868:

```
[1,3,5,17,21]->[2,4,6,18,22]->[3,7,19,23,33]-f->[4,8,20,24,34]
 ->[17,21,33,35]-f->[18,22,34,36]->[1,5,7,19,23,35]-s1->
  [2,6,8,20,24,36]->[1,3,5,17,21]
```

All these possibilities are compared in Table 23 below.

21.16 More Pathological LC-Weak Keys Not KT1

Here we report some examples of keys which are not KT1 and which are yet weaker than all those found in Table 23 or on page 81 or in later Table 23 etc. We also show the details of linear properties.

```
797: P=32,14,4,9,28,10,22,33,1,12,5,11,16,25,36,30,6,29,
34,35,31,2,19,20,3,13,15 D=24,20,4,8,16,0,32,28,12
 [1,5,12]->[2,6,33]-f->[3,7,34]->[4,8,17,21,35]-s1->
  [9,13,18,22]->[10,14,19,23]->[11,15,20,24]->[1,5,12]
811: P=6,25,15,3,16,33,1,23,12,7,32,31,27,8,36,17,19,14,18,
35,11,28,5,2,34,13,29 D=8,4,24,20,36,0,16,12,32
 [9,13]->[10,14]->[11,15]->[12,16]->[25,29,34]->[26,30,35]
  ->[27,31,36]->[17,21,32]-s1->[18,22,33]-f->
   [19,23,34]->[20,24,35]->[9,13]
712: P=28,26,33,32,30,24,5,15,9,8,22,13,4,6,21,10,20,25,
16,36,11,31,27,29,17,18,12 D=0,16,28,32,12,20,4,24,8
 [1,5,27,31,35]->[2,6,28,32,36]->[3,7,9,13]->[4,8,10,14]->[11,15,25,29,33]
  -f->[12,16,26,30,34]->[1,5,27,31,35]-s1->[2,6,28,32,36]->[3,7,9,13]
   ->[4,8,10,14]->[11,15,25,29,33]-f->[12,16,26,30,34]->[1,5,27,31,35]
813: P=8,18,17,13,29,33,26,28,12,32,30,19,9,27,10,34,16,5,
35,11,1,6,31,23,14,25,15 D=32,28,4,8,36,12,20,24,16
 [1,5,16]->[2,6,33]-f->[3,7,34]->[4,8,35]->[9,12-13,36]->[14,17,21]
  ->[15,18,22]->[16,19,23]->[20,24,33]-f->[25,29]->[26,30]
   ->[27,31]->[28,32]->[1,5,16]
913: P=20,4,5,9,7,33,1,15,22,16,13,17,23,25,30,28,34,27,
11,21,29,26,32,36,31,10,19, D=32,28,24,36,4,8,16,12,20
 [1,5,20]->[2,6,33]->[3,7,34]->[4,8,35]->[17,21,24,36]->[9,13,18]
  ->[10,14,19]->[11,15,20]->[12,16,33]->[25,29]->[26,30]
   ->[27,31]->[28,32]->[28,32]->[1,5,20]
```

Table 21. Keys with a large number of invariant linear characteristics which however are not KT1.

LZS nb	rounds	solutions
797	7	7
811	11	11
712	6	6
712	12	12
813,913	1	1
813,913	4	2
813,913	13	13

These keys have been generated in 1 hour approximately by our proprietary software which uses a SAT solver in order to generate keys with arbitrary specified characteristics together with a full formal mathematical proof of a linear property. All these keys are not bijective and therefore broken by powerful ciphertext-only attack of Section 20.2.

Even More Pathological LC-Weak Keys Not KT1

The longest not reducible cycle we found has 16 rounds. Below we give an example:

```
916: P=20,10,35,23,18,36,12,17,16,32,33,11,34,31,4,1,30,
25,2,7,27,8,5,26,29,6,28  D=0,32,24,28,16,12,8,4,36
 [1,5,33]-s1f->[2,6,34]->[3,7,35]->[4,8,36]->[25,29]->[26,30]
  ->[27,31]->[28,32]->[1,5]-s1->[2,6]->[3,7]->[4,8]->[25,29,33]
   -f->[26,30,34]->[27,31,35]->[28,32,36]->[1,5,33]
```

Table 22. Keys with 16 invariant linear characteristics, not KT1.

LZS nb	rounds	solutions
916	1	1
916	4	4
916	6	2
916	8	8
916	16	16

21.17 Keys With Self-Similarity and Level 2 Linear Cryptanalysis

It is possible to construct keys such that ALL inputs e2-e6 of Z1 and Z2 in one round are identical so that a hidden linear property is revealed. Such keys are not KT1 and their round function is not bijective [therefore are expected to be broken anyway by powerful ciphertext-only attacks, cf. Section 20.2].

```
617: P=26,31,5,6,25,34,22,30,27,1,2,29,23,35,17,9,28,20,7,19,
3,12,14,32,13,16,33 D=28,32,12,24,36,20,4,8,0
[4,8,34]->[25,29]->[26,30]->[27,31]fs1->[28,32-33]
 ->[1,5]->[2,6]fs1->[3,7,33]->[4,8,34]
[17]->[18]->[19,33]->[20,33-34]->[21,34Z1Z2]->[22,33,35]
 ->[23,34,36]->[17Z1Z2]
```

The second linear property is "unlocked" by the first property which concerns the self-similarity of Z1 and Z2.

Remark: Another (weaker) example of "Level 2" linear attack where the linear property is reinforced or unlocked by another linear property can be found in Section 22.7.

21.18 First Classification of LC-Weak KT1 Keys

We recall from Section 21.9 that when $D(1) = 0$, $D(2) = 4$ and $P(27) = 6$ we have a weak key w.r.t. LC. The property $D(1) = 0$ is obligatory in all KT1 keys. What are other possible values for $D(2)$ and $P(27)$? We observed that there is a large number of LC-weak keys with $D(2) = 4$, below are two examples:

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

There also exist many cases with $D(2) \neq 4$. Overall there are 168=8*21 possible values for pairs $(D(2), P(27))$ for KT1 keys, cf. Fact C.13.2 page 135, and computer simulations show that 126 out of 168 are compatible with LC-weak keys. Below we show some examples:

```
D(2)=12 P(27)=7
612: P=15,19,33,28,18,36,5,27,9,3,35,13,16,24,21,26,20,25,11,
32,8,12,23,29,4,31,7 D=0,12,28,4,24,8,16,36,20
[1-7,9-16,25-32]-s1->[1-7,9-16,25-32]
D(2)=36 P(27)=7
636: P=15,13,33,28,18,12,5,27,9,26,3,31,8,16,21,23,24,25,11,
20,35,19,4,29,32,36,7 D=0,36,28,32,24,16,8,12,4,
[1-7,33-36]-s1f->[1-7,33-36]
```

21.19 A More Detailed Classification of LC-Weak KT1 Keys

We have found by long and extensive computer simulations that among all possible $8! = 40320$ values for $\{j_1, j_2, \ldots, j_7, j_8\}$ there exist exactly 4549 weak permutations which are compatible with LC-weak keys.

21.20 On Frequency of LC-Vulnerable KT1 Keys

We have approximately $2^{83.2}$ KT1 keys total, cf. Section 8.6 and [49]. Our computer simulations on generating and testing vast quantities of KT1 keys at high speed indicate that about 3.0 % of all KT1 keys are LC-weak. Inside these, some 10% or 0.3 % of the total are those with $D(1) = 0$, $D(2) = 4$ and $P(27) = 6$ which precisely those studied in Section 21.9 and in a recent paper [49]. In addition we found that there exist many other similar classes of keys for a total of 3.0 % of all KT1 keys.

21.21 How to Avoid LC-Weak KT1 Keys

This suggests a very simple method to avoid weak keys: avoid the 4549 cases[52] which leads to avoiding a proportion of about 11 % of 40320 cases. We recommend[53] this method.

[52] The full list of cases to be excluded can be downloaded from `http://www.nicolascourtois.com/equations/block/t310/T310_JinsideWeakKT1LCTaken.txt`.

[53] This method leads to avoiding all the 3.0 % LC-weak KT1 keys and about 8.0 % of additional keys which are potentially OK but will be excluded by this method. By this method we will obtain approximately $0.887 \cdot 2^{83.2} \approx 2^{83.0}$ KT1 keys. cf. Section 8.6 and [49].

21.22 Pre-Conditions for Selected LC-Weak KT1 Keys

In the following table we show a number of recurring strong patterns which repeat in many different LC-weak keys. The pre-conditions listed were found by UCL student Marios Georgiou (cf. Thm. J.1.1 and J.2.1 and J.3.1 in Appendix).

Table 23. Pre-conditions for various types of invariant linear trails observed

pre-conditions	linear trail	Nr	LZS nb
D(2)=4, P(27)=6	$[1,3,5]s1 \rightarrow [2,4,6] \rightarrow [1,3,5]$	2	783, 788
$\{D(3), D(4), P(20), D(7), D(9)\} \subset \{12,16,28,32,36\}$	$[9,13,25,29,33]f \rightarrow [10,14,26,30,34] \rightarrow [11,15,27,31,35] \rightarrow [12,16,28,32,36] \rightarrow [9,13,25,29,33]$	4	785
$D(3)=0, D(4)=12, P(20)=16$ or $D(3)=16, D(4)=0, P(20)=12$	$[9,13]s1 \rightarrow [10,14] \rightarrow [11,15] \rightarrow [12,16] \rightarrow [9,13]$	4	774
D(7)=16, P(27)=10 $\{D(2), D(9)\} \subset \{28,32\}$, $\{D(3), D(4), P(20)\} \subset \{4,8,36\}$	$[1,5,15,33]s1f \rightarrow [2,6,16,34] \rightarrow [3,7,25,29,35] \rightarrow [4,8,26,30,36] \rightarrow [9,13,27,31] \rightarrow [10,14,28,32] \rightarrow [1,5,15,33]$	6	706
$D(2)=36$, $D(9)=4$, $\{D(5), D(6), D(7)\} \subset \{8,20,24\}$, $P(27)=6$	$[1,3,5,17,21] \rightarrow [2,4,6,18,22] \rightarrow [3,7,19,23,33]f \rightarrow [4,8,20,24,34] \rightarrow [17,21,33,35]f \rightarrow [18,22,34,36] \rightarrow [1,5,7,19,23,35]s1 \rightarrow [2,6,8,20,24,36] \rightarrow [1,3,5,17,21]$ + a 2nd property for 8R, cf. Thm. J.2.1.	8	868
TBD	$[1,5] \rightarrow [2,6] \rightarrow [3,7] \rightarrow [4,8-9,13]s1 \rightarrow [10,17,21] \rightarrow [11,18,22] \rightarrow [12,19,23] \rightarrow [20,24,33]f \rightarrow [1,5]$	8	768
$\{D(1), D(2)\} \subset \{28,32\}, \{D(7), D(8)\} \subset \{4,8\}, D(9)=12, P(6)=10, P(27)=34$	$[1,5] \rightarrow [2,6] \rightarrow [3,7] \rightarrow [4,8] \rightarrow [11,25,29] \rightarrow [12,26,30] \rightarrow [27,31,33]f \rightarrow [28,32,34] \rightarrow [1,5]$	8	778
$\{D(7), D(9)\} \in \{12,16\}$, one of $\{D(3)/D(4)\}=32$, with the remaining of $\{D(3)/D(4)\} \in \{28,36\}$ and finally the remaining of $\{28,36\}=P(20)$	$[9,13] \rightarrow [10,14] \rightarrow [11,15] \rightarrow [12,16] \rightarrow [25,29,33]f \rightarrow [26,30,34] \rightarrow [27,31,35] \rightarrow [28,32,36] \rightarrow [9,13]$	8	782, 784, 787, 788, 789

21.23 Software for KT1 Key Generation Tool and LC-Weak Keys

One way to find KLT1 keys with special properties is to run the Python tool of Section I.11 as follows:

```
python3
keygen.py --linprop "4x [11,15,27,31]" --print
```

Here by specifying pre-conditions one by one we can narrow down the range of keys generated and find KT1 keys faster. Here removing printing option is useful, in order to see more stats displayed on screen, in particular the last column says what percentage of LZS with $D[2] = 4$ will satisfy our linear property:

```
python3
keygen.py --force "D[1]=0,P[2]=1" --linprop "6x [1,5,15,33]" --save_linprop
```

Here the option save_linprop allows to produce a file with example of long term keys.

It is also possible to specify additional masks for finding special keys for advanced linear attacks:

```
python3
keygen.py --linprop "8x [9,13]" --auxmasks "[25,29,33]" --save_linprop
```

Then the file level2_lc.txt can be inspected.

In addition we have hard-coded some extra complex conditions, for example "–clinprop 1" stands for Custom Linear Property 1, a set of hardcoded preconditions on P and D which are (in this case: $D(7) = 16, \{D(3)/D(4), P(20)\} \subset \{4, 8, 36\}, P(27) = 10 and \{D(2), D(9)\} \subset \{28, 32\}$, cf. Table 23 above.

```
python keygen.py --clinprop 1 --linprop "6x [1,5,15,33]"
     NKT1/total   Keygen Speed  CLinPr/tested   LinPr/tested    SpeedPostLP
23287/3583527        4773k/s       2^-15.08         2^0.00           0k/s
```

Similarly, we have hard-coded the set of conditions in the last line of Table 23 and this is known as Clinprop 2:

```
python keygen.py --clinprop 2 --linprop "8x [9,13]"
```

Another tool to generate KT1 keys with force conditions on D and P is as follows:

```
ax64 888102 /forceD[7]=16,P[27]=10
```

22 Weak LZS Keys and Backdoors

In this section we study the question of malicious or accidental selection of a long-term key LZS such that it is unusually weak. We also recall major weak key attacks studied in other parts of this paper.

22.1 Weak Keys for Related-Key Differential Attacks

One example of this is given in Section 12.4. One major reason why such attack are possible, starting from 5 rounds, are the observation that the round function of T-310 uses too few bits in each round for no reason, this is definitely a property which degrades the security of the cipher, cf. Section 12.2. No keys are exempt from such attacks, not ever KT1 keys, cf. Thm. 12.4.1 page 44. Even historical keys (subject to even more careful selection) are vulnerable, though for a larger number of rounds at most 16.

22.2 Weak Keys With Shrinking Properties

Many weak keys for T-310 are non-bijective. Such keys typically shrink the output space, cf. Section 18.5 and lead almost inevitably to powerful correlation attacks cf. Section 18.9 for almost every bit inside the cipher. The actual ciphertext-only attack on T-310 which exploits this is studied in Section 20.2.

22.3 Weak Keys With One Bit Correlations and Sliding Attacks

Such keys lead to decryption oracle sliding attacks studied in Section 25. These keys are also moderately weak w.r.t. LC but not extremely weak, see keys 701 (not KT1) and 741 (KT1) in Section 21.5.

22.4 Strongly Weak Keys for LC - Open Problems

It is an open problem to see if an how the weak keys such as 606 or 786 listed in Table 16, here invariant properties have more than 1 active bit, could (at all) be exploited in a key recovery attack. These are clearly super-pathological keys with respect to Linear Cryptanalysis, yet it is not clear if communications with T-310 can be decrypted with these keys. Currently no such attack in known[54] and we conjecture that in many cases such keys could be secure or secure enough in practice to protect communications.

However as a precaution we recommend to avoid LC-weak keys totally, and one practical method to achieve this which does not degrade the space of possible KT1 keys is proposed in Section 21.21.

Future research will show IF or HOW various weak keys can be exploited to design interesting key recovery attacks on T-310. The answer is almost certainly that many of these properties alone do no suffice to decrypt communications and that the cipher remains robust. In order to actually break T-310 we need to combine more than one vulnerability in one single long-term key.

[54] Except keys 529 in Section 22.8 below and key 421 in Section **??** below.

22.5 Weak Keys Which Combine LC-weakness with RKDC

To illustrate this we found that there exist keys which are simultaneously very weak w.r.t. LC and RKDC, for example key 718.

718: P=3,18,33,12,36,8,5,27,9,19,14,23,20,16,21,26,7,25,31, 28,32,15,4,29,24,22,6
D=0,4,36,12,24,16,20,8,32,

22.6 Keys Which Combine LC-weakness with One-Bit Correlations

We found two keys which will name 529 and 421 are simultaneously an LC-weak key and is also broken the decryption oracle attacks of [47]. These attacks are also and further studied in Section 25.

LZS nb	rounds	input → output	bias	prop./keys
421	4	[21] → [21]	$2^{-15.5}$	TBC
421	8	[9,13] → [9,13]	$2^{-1.0}$	TBC
529	5	[29] → [29]	$2^{-7.0}$	TBC
529	8	[9,13] → [9,13]	$2^{-1.0}$	TBC
617	8	[25,29] → [25,29]	$2^{-1.0}$	TBC
617	7	[17] → [17]	*	TBC

Table 24. Keys which are LC-weak and simultaneously have HW1 invariants correlations. *Key 617 is very special, we refer to Section 21.17 for more details.

We give a few extra explanations about what exactly happens for 421. It turns out that for this key a relatively weak correlation exists initially, bias of $2^{-15.5}$. With this figure, the attacker will have hard time to decrypt T-310 communications following the decryption oracle attacks of [47] and Section 25. Now the attack can be IMPROVED by using the second property as explained below.

22.7 More Details About Key 421

We look at what happens inside the 4R invariant property of bit 21. Nothing exceptional happens in the first three rounds, bit 21 becomes 23.

```
[21]->[22]->[23]
```

We only need to study the last round:

```
[24]->[21,Z1Z2]
Z1 inputs are s2 and 28 4 33 32 20
Z2 inputs are 5 3 9 11 10 14
24=P(6)
20=D(6)=P(5)
```

The input 21 is the XOR of:

$$u_{21} = U_6 = f \oplus Z_1(s_2, u_{P(1-5)}) \oplus u_{P(6)} \oplus Z_2(u_{P(7-12)}) \oplus u_{D(6)}$$

Now with key 421 we have D(6)=20 and P(6)=24 therefore:

$$u_{21} = f \oplus Z_1(s_2, u_{P(1-5)} \oplus Z_2(u_{P(7-12)}) \oplus u_{24} \oplus u_{20}$$

Now it turns out that the Boolean function $Z()$ has the following linear approximation:

`Z1=v5 in 4th round P=34/64`

which knowing that $P(5) = 20$ means that

$$Z_1(s_2, u_{P(1-5)}) = \oplus u_{20} \quad \text{with probability} \quad \frac{34}{64}$$

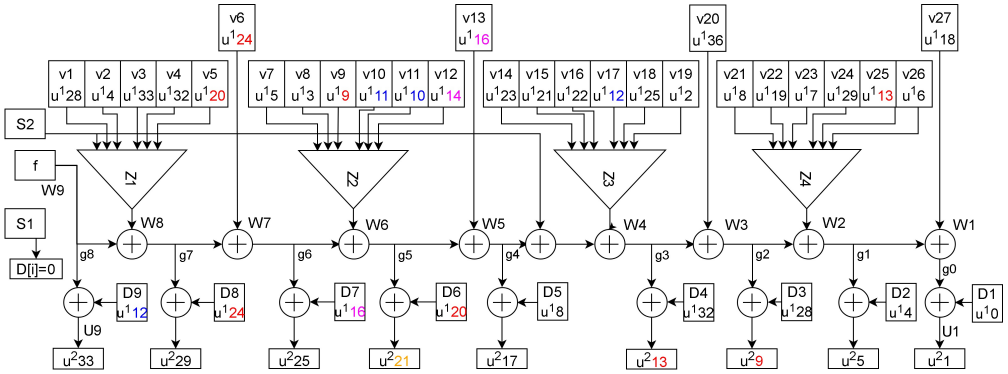


Fig. 22.24. One round of T-310 for key 421.

Overall the input 21 XORed with the same input 21 taken 4 rounds later is equal to Z2 output in the last 4th round with probability 34/64:

$$u_{21}^{(0)} + u_{21}^{(4)} = Z2^{(4)} \quad \text{with probability} \quad \frac{34}{64}$$

And that for some reason this leads to a weak bias of $2^{-15.5}$. We now show that this result can be included and that a stronger bias (for the same property) can be obtained.

22.8 A Correlation Attack with a Hidden Enhancement

We recall the exact linear property from Sections 21.14 Appendix J.1 and Table 23.

```
[9,13]->[10,14]->[11,15]->[12,16]->[25,29,33]-f->
                    [26,30,34]->[27,31,35]->[28,32,36]->[9,13]
```

We see that the attacker can predict the XOR of inputs e5 and e6 of Z2 which are always bits P(11)=10 and P(12)=14 for this key, at any number of rounds, knowing only a number of IV bits f_i which can all be computed from the IV transmitted in the clear during the communications. The attacker can therefore predict the XOR of inputs e5 and e6 at input of Z2.

This will be used to improve the previous attack as follows. It is easy to see that we have:

$$Z_2(u_5, u_3, u_9, u_{11}, u_{10}, u_{14}) = u_{10} \oplus u_{14} \quad \text{with probability} \quad \frac{38}{64}$$

And that $u_{10} \oplus u_{14}$ being known to the attacker, we get a very strong correlation attack on

$$u_{21}^{(0)} + u_{21}^{(4)}$$

with a bias of $2^{-8.4}$ (cf. Matusi's piling-up lemma [82]). This instead of bias of $2^{-15.5}$ without these extra deductions.

22.9 Weak LZS Which Leak The Encryption Key - Kleptography

It is possible and relatively easy (well if we do not bother about KT1 rules) to generate LZS for which the internal keystream a_i is strongly correlated to the key bit $s2$ at a given location. More precisely, by simple random search we generated the following keys:

```
925: P=34,24,33,26,14,4,5,28,9,32,12,18,36,16,21,15,8,25,35,
20,1,6,23,29,19,27,13 D=0,16,36,12,32,28,4,8,24
929: P=14,34,33,4,24,32,5,8,9,26,27,18,36,16,21,15,20,25,35,
28,1,6,23,29,19,13,12 D=0,36,20,32,8,4,12,24,16
921: P=28,16,33,26,23,12,5,20,9,31,4,13,36,6,21,17,8,25,10,
30,32,34,3,29,18,24,2 D=0,24,8,32,4,20,36,12,16
```

For these keys one bit of the cipher state, which can be α is correlated to a simple function of $s2$ which is the bits used at the current rounds, and f, which is the IV bit which will be known to the attacker.

Table 25. Examples of weak LZS where information about key leaks to the attacker directly. We also show examples of conditional bias when bits $s1, s2, f$ are fixed.

LZS nb	sum	α	$Pr[u_\alpha = sum]$
929	s2+f	29	$1/2$ -$2^{-3.4}$
925	s2+f	25	$1/2$+$2^{-5.6}$
921	s2+f	21	$1/2$+$2^{-8.6}$

LZS nb	s1	s2	f	α	$Pr[u_\alpha = 0]$
929	1	0	0	29	$1/2$ -$2^{-3.0}$
925	0	1	0	25	$1/2$ -$2^{-5.0}$
921	0	0	0	21	$1/2$+$2^{-8.2}$

22.10 Key Recovery With Weak Keys of Type 929

With these keys it is EXTREMELY easy to break T-310: every $120 \cdot 127$ steps the same key bit $s2_i$ is used again. Each bit of type $s2$ can be recovered at different places. In the known plaintext attack, we can recover this $s2_i$ bit by majority voting [after subtracting f value for each step]. Data complexity will be only thousands of encrypted characters.

In the ciphertext-only scenario we apply Thm. 17.3.1 which allows us to recover 5 bits out of 13 in 1/32 of cases where $C_j = 0$. Then we know that $P_j = B_j$ on 5 bits. This combined with plaintext bit biases such as in Table 6 we can recover the key using maybe 100,000 characters of ciphertext [rough estimation].

Remark. In Section B we read that $\alpha \notin W$ and therefore $\alpha = 25$ and $\alpha = 29$ are forbidden. This is however illusory: in the next round the **same** internal state bit is at $\alpha = 26$, then at $\alpha = 27$ and $\alpha = 28$, after which this bit will be forgotten. These bits can be used for encryption and for example we can have $\alpha = 27$ which is not forbidden and this is subject to the same correlation attacks as for $\alpha = 25$ except that they involve key bits 2 round earlier.

22.11 Detailed Investigation of Certain Weak Keys of Type 929

It is easy to see why key 929 works: inputs of Z2 are $s2, P(1-5)$ cf. Fig. 22.25 below. and the output of Z2 is XORed with bit $D(8)$. For key 929 we have $D(8) = 24 = P(5)$ which is the last inputs of $Z1$. Our backdoor correlation property is simply due to the fact the output of the Boolean function $Z(e_1, e_2, e_3, e_4, e_5, e_6) \oplus e_6$ is correlated to the first input $e_1 = s2$ and knowing that f is known to the attacker.

For key 925 it similar yet less simple: we have and $D(7) = 4 = P(6)$ therefore bit 25 is equal to $Z1 \oplus f$. Here we need to see that $s2$ is correlated to $Z(s2, e_2, e_3, e_4, e_5, e_6)$.

Applicability. None of these two pathological situations can happen with KT1 keys. In the first case we had for key 929 that $D(8) = P(5)$, however for the KT1 keys we have always $P(6) = D(8)$. For key 925 we had $D(7) = P(6)$ however for the KT1 keys we always have $P(13) = D(7)$. This makes that historical KT1 keys do not fall to our attack. Moreover, these keys are not-bijective cf. Table 11. It is an open problem if KT2 keys or any sort of bijective LZS would be vulnerable to this attack.

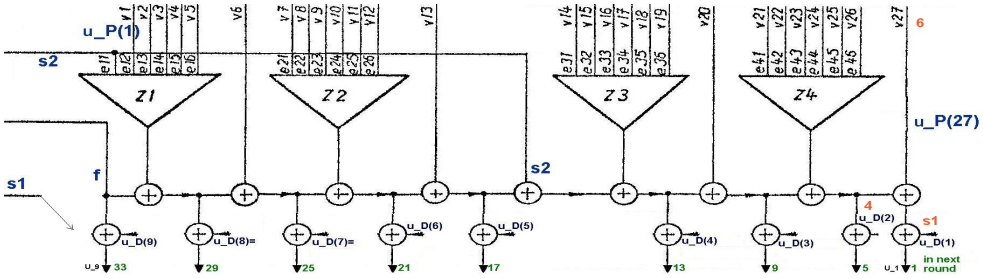


Fig. 22.25. For convenience we show the bits involved here, cf. also Fig. 7.11.

22.12 Can Bits S1 also be Recovered and 2R Correlation Attacks

It is possible to see that in order to find biases which involve S1 bits, one round of encryption is not sufficient and we need at least 2 rounds. For example the following key 942 has output 21 correlated to prevS1+f by which we mean the $s1_i$ bit used in the previous round XORed with the f bit used in the current round.

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

22.13 Further Correlation Attacks on S1

We outline some additional correlation attacks on S1 which could work for 4 or more rounds in Appendix B.3 and in Appendix C.12.

22.14 More Rounds, More Correlation Attacks, Attacks Without Correlations and General "Random Non-Bijective" LZS Case

Looking at Table 25 the reader might consider that we have found something which is a class of particularly weak keys with some very strong correlations cf. which do not have any practical importance. In this section we claim that, on the contrary, a simple attack described in Section 22.10 can be easily generalized and will work almost always for arbitrary long-term keys(!) provided that they are not bijective.

In addition, this attack has already been studied in a lot of detail in this paper, see Sections 18 through 20. In general we do **not** need to use a correlation attack and we do **not** need correlations. More generally we will have a conditional correlation attack or just an attack with a conditional bias, which can work nevertheless in many cases when no correlations exist whatsoever. The reason for that is that biases for some bit u_α can exist for particular values for key bits K in say last 4 rounds, even though if you average that over all K the biases are compensated and we frequently get no correlation at all. We have the following general attack:

Conjecture 22.14.1 (General Attack For Arbitrary Random Non-Bijective LZS). We conjecture that if we generate a non-bijective LZS key at random, and almost always [with overwhelming probability close to 1], after a few rounds, if the key bits used in these rounds are fixed, the internal state u_α will exhibit biases for **every** alpha.

Justification: Examples of such biases can be found in Table 12 page 63. Moreover our experiments show that this attack obey a relatively simple law with bias $\approx \mathcal{O}(\sqrt{1/M})$, see Section 18.9 where M is the output space size. The only keys known for which this formula is not correct are keys which are substantially weaker such as 925. Overall we see that what is exceptional for ϕ^1 cf. Table 25 above, is already systematic for ϕ^{16}, see Table 12 in Section 18.2. We conjecture that very few non-bijective long-term keys exist which can resist this attack.

23 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. We will assume that α is known or we guess it (it has low entropy and many choices are substantially weaker). In this case we are going to show (over the next few pages) that the cipher is not secure.

We will fist look at the first encrypted character C_j with $j = 1$. For example the attacker can send several messages with the same IV with:

$$C_1 = P_1 \cdot M^{r_1} \oplus B_1 \cdot M^{r_1}$$

Then in all these encryptions r_1 and B_1 will be the same. So for two encryptions with the same IV we have:

$$C_1 \oplus C_1' = (P_1 \oplus P_1') \cdot M^{r_1}$$

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

$$C_j \oplus C_j' = (P_j \oplus P_j') \cdot M^{r_j} \quad \text{for all } \ 1 \le j \le k.$$

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:

$$r_0 = \begin{cases} 0 & \text{if} \quad R_j = (0,0,0,0,0) \\ 0 & \text{if} \quad R_j = (1,1,1,1,1) \\ 31 - r & \text{if } R_j \cdot M^r = (1,1,1,1,1) \end{cases}$$

Therefore we need to discard the proportion of 1/32 of the cases where r_j is 0 modulo 31, which cases create an ambiguity on the bits R_j.

For example, for $j = 1$ we can determine $R_1 = a_{1-5}$ with probability at least 30/32, as one of the two problematic events happens with overall probability at most 2/32. Overall in at least 30/32 of the cases over all possible pairs $P/C, P'/C'$, we have the 5 bits of $R_1 = a_{1-5}$ which are uniquely determined and $r_1 \neq 0$ (only in case $r_1 = 0$ we cannot determine R_1, i.e. decide whether $R_1 = (0,0,0,0,0)$ or $R_1 = (1,1,1,1,1)$).

Theorem 23.0.1 (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. 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 $C_1 \neq C_1'$ 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 $C_1 \neq C_1'$ for any pair by selecting up to 32 chosen ciphertexts with different C_1. This cannot be done for more than 32 of the 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 $r_1 = 0$ is extremely small, of the order of $(1/31)^{\binom{32}{2}} \approx 2^{-2457}$.

Key Recovery Step. Access[55] to many values of $a_1 = u_{127,\alpha}$ for many different IVs should maybe be sufficient to recover the T-310 key with a SAT solver cf. Section 15 and [15] or ElimLin+ attack [99] in a similar way as for 160 of KeeLoq cf. [31, 32]. The exact time and data complexity of this attack will be studied in a future update of this paper.

23.1 General Black-Box Decryption Oracle Attack

We generalize the same attack for recovering most of the $a_1, a_2, a_3, \ldots, a_{13k}$ for k characters. It should be noted that we cannot hope to recover a_j with $j \equiv 0, 6,$ or 12 modulo 13, because these a_j are never used for encryption. We can however recover all these which are used for encryption (a proportion of 10/13 of all the a_i). 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. We also make a deliberate choice NOT to recover all the a_i which could be recovered, in order to minimize the data [decryption oracle query] complexity of our later attack. We explain our general method:

Theorem 23.1.1 (General Decryption Oracle Attack). For every IV chosen by the attacker, and for every $k \geq 1$, the attacker can obtain a proportion of $30/32 \cdot 10/13 + 1/32 \cdot 5/13 \approx 0.73$ of the internal keystream bits a_{1-13k} with a computation effort of about $2k$ and with about $K = 2$ "Chosen IV and Chosen Ciphertext" (CIVCC) queries on average, with one fixed chosen IV and random ciphertexts, and with ciphertexts length of about k characters. For the remaining values a_i we make the algorithm return "don't know".

Proof: We assume that we have K decryption CIVCC oracle queries (with chosen IV and chosen ciphertexts) with ciphertexts of length k characters. This would give 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 apply the analysis of the previous Section 23 and even if sth. go wrong, we could recover the a_i from another pair [except those never used].
Now in this paper we made a "minimalistic" choice of $K = 2$. Exactly, and only, two things can go wrong for our pair of decryptions obtained from the oracle. Either we have $C_j = C'_j$ or $r_j = 0$. Avoiding both cases happens with probability at least 30/32. In this case we can determine 10/13 of the bits uniquely from the decrypted pair. We also have a case where $C_j \neq C'_j$ but unhappily $r_j = 0$ and R_j cannot be determined for sure (ambiguity), in this case however B_j can be obtained from $C_j = P_j \cdot M^{r_j} \oplus B_j \cdot M^{r_j} = P_j \oplus B_j$. This happens with probability about 1/32, in this case we only get 5/13 of the bits of $B_j = (a_{7+13(j-1)}, \ldots, a_{11+13(j-1)})$.

[55] Even though the first 4 characters (or 20 bits) of the plaintext are always known to the attacker in real life situations. cf. Appendix I.5, this does NOT give the values $u_{127,\alpha}$, due to double one-time pad cf. Section 16.

24 A Decryption Oracle with a Slide Attack

In the previous section Thm. 23.0.1 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 $K = 2$ to recover most (but not all) of these bits. This gives access to a_1 which however still depends on 240 key bits. In addition we get access to further $a_2, a_3, \ldots$ in the same way by using longer decryption queries. 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. [73, 31, 40]. We want to exploit the 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 permutations. Then the question will be whether these identical permutations can have identical inputs. Traditionally researchers call such pairs of inputs a 'slid pair' [5].

Here is our first basic slide attack. First we consider some integer s such that:

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

For example $s = 18$ has $120 \cdot 18 = 127 \cdot 17 + 1$ and $d = 1$. Or[56] $s = 1$ given $120 \cdot 1 = 127 \cdot 1 - 7$ and $d = -7$. Now the main idea in the attack is that **if** by some sort of "happy" accident for some encryption with some IV, we have

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

then the attacker can detect this fact efficiently **if** there exist correlations on bit α for d rounds, cf. Section 25.1 **and if** the attacker has access to a certain type of chosen IV and chosen ciphertext attack (with partial recovery of the internal keystream) such as in Thm. 23.1.1.

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 to obtain a "slid pair".

[56] The first example, as we will see later, is one we have designed for "stronger" long-term key settings of T-310, cf. later Appendix G. The second example is meant to work more easily for "weaker" long-term key settings of T-310.

25 Slide Property Detection With Decryption Oracle and Internal Correlations

In this section we will first see what kind of correlations we need for our attack. Then in Section 25.2 we will describe the full procedure for the detection of a 'slid pair'.

25.1 On the Existence of Suitable $[\alpha] \rightarrow [\alpha]$ Correlations

We consider some relation of type

$$120s = 127t + d,$$

where d is small in absolute value and also s and t are not too large. For example $(s, t, d) = 18, 17, 1$ or $(s, t, d) = 1, 1, -7$. At this moment we will concentrate on the case of $d = \pm 7$, cf. Appendix G for $d = 1$ case. We want two bits used for encryption in two encryptions shifted by $120s$ rounds to be correlated, cf. Fig 25.2 page 100. In other words order to make our slide attack we need a correlation property of type: one special bit $\alpha \in \{1 - 36\}$ of the block cipher state is correlated with the same bit α after d rounds for some small d.

$$s_{i,\alpha} \stackrel{?}{=} s_{i+d,\alpha} \quad \forall i$$

This can be seen as a special case of Linear Cryptanalysis (LC) [82]. However we only look at invariant linear characteristics with Hamming weight 1, which will be substantially less frequent. To illustrate this we can compare Table 15, Table 16 and Table 19 in Section 21. Correlations of type $[\alpha] \rightarrow [\alpha]$ are clearly extremely few or are substantially weaker than other correlations. For example for the historical key 31, and for 7 rounds, we found the following linear characteristics $[17] \rightarrow [36]$ with probability 1 for some 30 % of key/IV cases, which unhappily is not invariant and it is not really useful for us, and for 17 rounds we have that $[17] \rightarrow [25,29,35]$ with probability $1/2 \pm 2^{-3.00}$ also for some 30 % of key/IV cases.

In general the answer depends on the values of d and the choice of the term key LZS with specific values for D, P, α. We conjecture that for every D, P, α there exists several d such that our attack can be made to work (with $s > 1$ it will be harder). In practice however in this paper we want to design a simple attack with $d = \pm 7$ and for this value of d we have NOT found a convincing real-life example with a sufficiently large correlation. This indicates that the East-German cryptologists have done a very good job at preventing our basic slide attack, described in this article, from being effective in practice. Until now we found no suitable strong correlations for our attack for any of the original keys from [55] except for the peculiar key 27 and for a different value of d, cf. Table 19 page 19. In general probably there is no way to prevent such attacks from working with a sufficiently large d or/and with a weaker correlation, in particular however, we do not have a convincing example. For this reason, and for simplicity, we are going to evaluate the complexity of our attack on a key which we have generated ourselves and which exhibits the correlation required for $d = \pm 7$ exactly. In this paper we use for example the following key called 701 for which we have a suitable invariant property $[30] \rightarrow [30]$ for some 20 %

of key and IV choices, cf. Table 19 page 78 which also contains the specification of this key 701. This can also be done for key 741 which is a KT1 key.

25.2 Slide Property Detection With Decryption Oracle and Internal Correlations - The Detailed Procedure

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

Theorem 25.2.1 (Sliding Property Detection with a Decryption Oracle). For every IV chosen by the attacker, and for every $s \geq 1$, the attacker can detect with near-certainty if $u_{120s} = u_0$ for the unknown key, by requesting a decryption of $K = 2$ ciphertexts for this IV with another related IV' which we specify below, with length $120s + k$ each, and with time complexity about $2k$ as in Thm. 23.1.1, where k is the decryption oracle query data capacity requested for a fixed set of parameters s, d, which will be determined later to achieve the desired confidence level four our distinguisher.

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 for both encryptions the attacker can recover most of the keystream with $K = 2$ decryption queries per IV, cf. Thm. 23.1.1.

Step 3. Step 3. We have $120s = 127t + d$ with d small. This means that IF again $u_0 = u'_{120s}$ the keystream extracted from the second encryption is shifted by $127t + d$, i.e. it is extracted at t "big" a_i-scale steps later with 127 rounds each, and with a ϕ^d offset. We can hardly hope that these bits will be identical BUT we can hope they will be in some cases correlated. We have

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

and

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

25.3 Step 4 - Simplified Correlation Analysis

Now as a first approximation, we see that the attacker has access to the 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

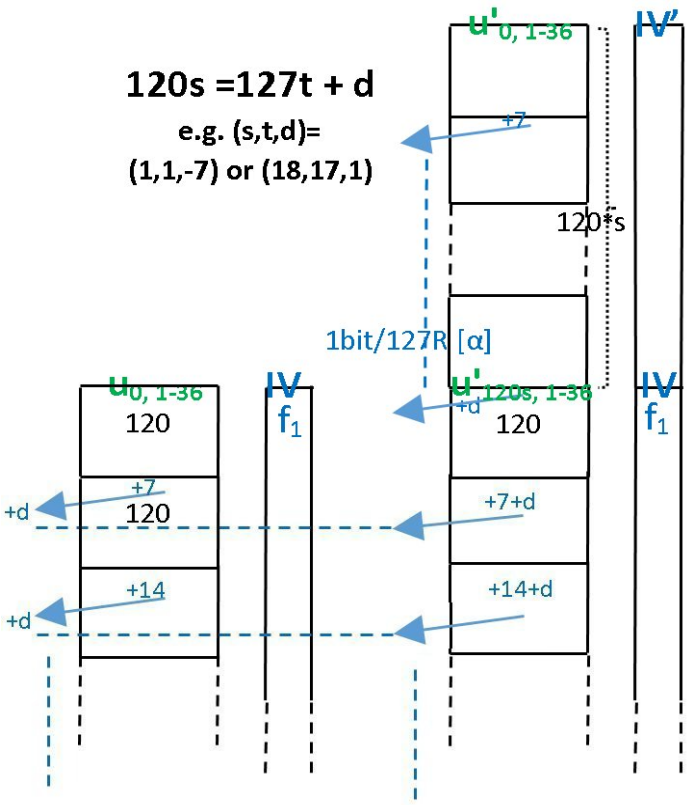


Fig. 25.26. Slide Attacks on T-310 - the IVs are identical at positions which are also distant by a multiple of 120, the keystream is shifted by ϕ^d, where d can be positive or negative.

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 the LZS 701 and $d = -7$ 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 detect if our sliding condition on 36 bits is true for $\alpha = 30$.

25.4 Step 4' - Actual Correlation Analysis

More precisely, it is not quite correct to say that the attacker has access to the sequences $u_{127j,\alpha}$ and $u'_{127j,\alpha}$ for every j. Following cf. Thm. 23.1.1 only 73 % of these bits can be recovered on each side. This makes that only some pairs $u_{127j,\alpha}, u'_{127(j-17),\alpha}$ will actually be available, actually a proportion of $(0.73)^2 \approx 0.53$. This is of course sufficient to detect the correlation with about twice higher k than otherwise needed, and we will estimate k below. This ends the proof of Thm. 25.2.1.

25.5 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 clocking the LFSR $120s$ steps backwards.

More precisely, following Thm. 25.2.1 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.

25.6 Data Complexity Required in Our Attack

At this stage we see that the attacker can generate P/C pairs for 120 rounds given that $s = 1$, and following Section 15 key recovery for 120 rounds with a SAT solver should be feasible. It is then easy to see that the we need to generate 7 such P/C conditions on 36 bits: one is not sufficient to uniquely determine a key on 240 bits. We need to estimate the data complexity needed to see if $u_{120s} = 0xC5A13E396$ will be simultaneously true in 7 cases with probability of at least 1/2 and to reliably discard as many as $2^{39} - 7$ cases. Therefore we need to operate with a precision which is sufficient to have the standard Gauss error function erf() to predict less than one false positive in 2^{39} experiments. We must be able to reject most cases with Thm. 25.2.1 operating at z standard deviations, where z is such that that $\text{erf}(z/\sqrt{2}) < 2^{-39}$, which gives $z = 7$, see the table in [106].

The standard deviation for N events, where equality of some two bits of type $u_{127i,\alpha}$ holds in Thm. 25.2.1, which is assumed true with probability $1/2 \pm \varepsilon$, with $\varepsilon = 2^{-11}$ here, will be about $\sqrt{N}$ and the deviation in observed probability will be $\sqrt{N}/N$. In order to detect correlations with confidence at or exceeding 7 standard deviations we need, approximately, $7\sqrt{N}/N \leq \varepsilon$. This leads to $N \geq 7^2 \cdot \varepsilon^{-2}$. Now, not all bits $u_{127i,\alpha}$ are simultaneously known in 2 distinct encryptions. Inside $13k$ possible bits a_i for each of $K = 2$ decryptions with k characters, only 73 % are

available, and out of these only 73 % are such that the correlated bit for the other decryption is also available to the attacker. This leads to $N \approx 13k \cdot (0.73)^2 \approx 6.9k$. We need $k = N/6.9 = 7^2 \cdot \varepsilon^{-2}/6.9 \approx 7\varepsilon^{-2}$ with $\varepsilon = 2^{-11}$.

25.7 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 then 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 240 bits).

1. We have $d = -7$ and $s = 1$.
2. The attacker will try some $7 \cdot 2^{36} \approx 2^{39}$ random IV_i on 61 bits. He can then expect that there exists some $2^{39-36} \approx 7$ "good" IVs where he has $u_{120s} = u_0 = 0xC5A13E396$. At this moment he does not know which 7 IVs are the "good" ones.
3. For each of IV_i, $i = 1 \ldots 2^{39}$ the attacker will step the IV exactly $120s$ steps 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. 25.2.1 cf. also Fig. 25.2. The attacker - with the help of a decryption oracle - can see if $u_{120s} = u_0 = 0xC5A13E396$ by aligning 2 sequences a_j and a'_{j+t}, where only $0.73^2 \approx 0.53$ of the pairs 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 $a_j = a'_{j+t}$.
   Following Section 25.6, the attacker needs to select 7 cases where $u_{120s} = 0xC5A13E396$ will be simultaneously true and reliably discard $2^{39} - 7$ cases. This leads to $k = N/6.9 = 7^2 \cdot \varepsilon^{-2}/6.9 \approx 7\varepsilon^{-2}$ with $\varepsilon = 2^{-11}$. We also need $120s$ more characters which is negligible.
7. Overall our attack requires $k = 7\varepsilon^{-2}$ characters of encrypted data where $\varepsilon = 2^{-11}$. We need about $k \approx 2^{25}$ characters of decrypted data per decryption query.
8. The data complexity is about $K \cdot 7 \cdot 2^{39} \approx 2^{43}$ chosen IV chosen ciphertext decryption queries, which are 2^{25} characters each in length.
9. The time complexity is about $2^{39} \cdot K \cdot 2^{25} \approx 2^{65}$ CPU clocks spent in examining correlations plus the time to recover the key from 7 P/C pairs for 120 rounds by a SAT solver attack. As long as this step takes less[57] than 2^{65} CPU clocks, this will NOT change the complexity of our attack. For the time being we assume it does.

Overall we see that we can recover the 240-bit key of T-310 with about 2^{43} chosen IV chosen ciphertext decryption queries with messages of less than 2^{25} characters each, cf. Section 25.7. The time required is about 2^{66} CPU clocks and the memory required is small.

[57] For example, in Table 1, Section 9, page 25, in [40], the time complexity decreases as the 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.

25.8 Further Sliding Attacks with $d \neq 0$

We can construct plenty of LZS keys for which this attack works, and this included numerous KT1 keys, for example key 741 for 7 rounds and key 529 for 5 rounds. for these key our sliding attack above works out-of-box. We can also construct numerous keys with "Near-Bit" correlations, cf. Section 21.8. This corresponds to a useful generalization of the present attack (it would require the attacker to have or assume additional sliding pairs of related encryptions). The main point is that bits of type up to $\alpha+3$ where $\alpha = 4k+1$ are accessible to the attacker at some earlier rounds. Moreover, in Section 21.8 we shows that real-life historical keys such as 32, are also vulnerable to slide attacks.

25.9 An Alternative Sliding Key Recovery Attack with $d = 1$

In the case of T-310 keys for which there is no correlation with $d = 7$, or if the SAT solver attack does not work as well as expected, we need to develop different sliding attacks. In Appendix G. we provide one such alternative sliding attack which was designed rather for the case of larger $s > 1$ and $d = 1$.

26 On Correlation Immunity in T-310

Correlation immunity, has been an important goal in the design of encryption machines for many decades, cf. [101]. 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 24 and Appendix G 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.

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. 3.31 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 $\alpha = 29$ from existing.

Another example is to see that some very strong correlations reported in Table 25 cannot exist for the KT1 or KT2 keys simply because these keys the round function is bijective, see Section C.10 and D.6 for detailed mathematical proofs for both classes of keys.

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 26.0.1. If α is not a multiple of 4 and it is one of the bits not used by T in Table 1 page 42, 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. 4.6 page 16. Then after one 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 from the input flip at α and these two bits are therefore not correlated.

We can also obtain a stronger result:

Theorem 26.0.2. 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. 4.6 after one 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 from 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. [96] and Section 5.4.

26.1 On Correlation Vulnerability in T-310

There some simple ways to make T-310 fall for truly devastating attacks where 120 bits of the key can recovered with extreme ease in the ciphertext-only scenario, see Table 25 and Section 22.10. This can be achieved for example by mandating that $D(8) = P(5)$ or that $D(7) = P(6)$, cf. Section 22.11, which cannot[58] happen for the KT1 keys, see Section B. More examples of correlation vulnerabilities and/or attacks with a weak LZS choice are listed in numerous Sections starting from Section 21.5 and in Section 22. Yet other types of correlation attacks are studied in Appendix B.3 and in Appendix C.12.

Remark. A natural extension of our sliding-correlation-decryption oracle attack of Section 25 with "Near-Bit" correlations is also studied in Section 21.8.

[58] More generally such correlations cannot happen for KT2 keys either because these LZS have a round function which is not bijective, cf. Appendix D.6.

27 Summary of Strong and Weak Points in T-310

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

27.1 On Per-Round Weakness vs. Number of Rounds

We discover that the round function has many issues. Serious weakness against Differential Cryptanalysis, cf. Section 9, and serious weakness against Linear Cryptanalysis either cf. Section 21 with as much as 3 % of KT1 long term keys which are pathologically weak. One round of this cipher clearly weaker than comparable historical block ciphers such as DES or RC2 and which also is not too strong w.r.t. software algebraic and/or SAT solver attacks, cf. Section 15. Then we have this peculiar internal triangular structure where we disconnect/replace just one bit in $\underline{\mathbf{D}}$ in an unbalanced Feistel scheme.

However due to an excessively large number of rounds per encrypted bit, the overall gate complexity of this cipher is incredibly high, cf. Appendix I.15. The **question is** now[59] can a "per round weakness" compensated by a larger number of rounds, and extremely large overall gate complexity per cf. [11], AND by the fact [60] that[61] extremely few[62] bits extracted from the internal state (1 bit every 127 rounds) are actually used for encryption.

Overall it is not obvious to claim that T-310 is broken. 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. Most vulnerabilities of T-310 should therefore be considered in relative terms. Our overall opinion on T-310 is that it exhibits a fascinating mix of properties, some of which make it extremely strong, other weaken the cipher very substantially.

[59] This is **not unusual**, the same problem occurs in for example in [40, 24, 38], and also for about half of block ciphers submitted to the NIST AES competition in the late 1990s.

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

[61] 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. [35].

[62] Needless to say smart people in the industry have known this for years, see for example [35, 77].

27.2 Definite Vulnerabilities of T-310

Below we provide a list of major **vulnerabilities** which we have identified in the design of T-310. Most of these properties are able to very substantially degrade the security of T-310 **for no apparent reason**. We believe that at least half of these properties simply cannot be defended[63] by any engineering or practical reasons known to us, such as the cost or speed of encryption. Therefore they should be rather considered as "bugs" rather than "features" this even though we do not always know how to exploit them in a convincing way. Several examples of such properties of T-310 with variable levels of severity are studied in this paper:

1. First we show that potentially key space could be halved, cf. Section 4.5, or the divide and conquer approach attacks on the key space are possible due to the permanent lack of mixing of its belonging to the two 120-bit halves of the secret key, cf. Section 7.6.
2. Then we show that long-term key space is surprisingly small, cf. Section 8.6,
3. Then we show that the structure of $T()$ has some issues, in particular the fact that 8 state bits per round are omitted has some important consequences, cf. Section 12.3, and Section 12.4. Other issues are studied in Section 9.4 and 21.4.
4. Then we show that the first 4 characters of plaintext are always known to the attacks, cf. Appendix I.5.
5. Then we show that the encryption procedure is subject to "Zero-value" attacks cf. Section 17.3 and Section 20.2.
6. At several places we exploit the periodic key scheduling of T-310, cf. Section 20.1, our slide attacks in Section 24.
7. The periodic key scheduling is in general hard to exploit. This with exception of our second "Zero-value" attack is described in Section 13.6 which is quite powerful.
8. At other places we show that weak long-term keys are a major concern and that validation of long-term keys is important in T-310. Several ways to backdoor the T-310 cipher are described in Section 22. If the attacker can select any LZS, the security of T-310 collapses totally, cf. Section 22.10.
9. More specifically we study powerful chosen-key attacks in Section 18 and show that they lead almost inevitably to powerful correlation attacks cf. Section 18.9 for almost every bit inside the cipher.
10. Many more types of correlation attacks on T-310 with weak or/and strong keys are studied in Section 20.2, several Sections starting from Section 21.5, Section 22, Section 22.9 and Section 25, and in Appendix B.3 and and also in Appendix C.12.
11. In this paper we show several ways in which a long-term key could be weak or backdoored, cf. Sections 4.5, 19 or in Sections 21.4 and 22.1 and many other.

[63] 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.

This poses serious problems to verify if long-term keys are bijective and if they should be approved for practical use, cf. Section C, Appendices C.10, D.6 and [48, 49, 55, 100].

12. Finally, are the actual historical keys the strongest possible? Almost everything we have done seems to indicate that yes, with two exceptions, cf. Section 8.3. In related-key differential attacks cf. Section 12.4 we have observed that the choice of α in these keys is potentially quite vulnerable. Then in Table 26 page 120 we see that historical keys could quite be vulnerable to correlation attacks on S1 specifically while it seems harder to develop any sort of attack on S2 (except in contrived cases).

Most of these should be considered as either engineering mistakes or at least they define definite areas where the cipher is weaker than it could otherwise be. Many of these observations also open interesting possibilities to potentially misuse the cipher or make it weaker on purpose just by selecting a weaker sort of long-term key, in chosen IV attacks, etc.

28 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 from the same historical period such as RC2, DES, and 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 [35]. In T-310 bits from rounds 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 25.7, allows one to recover the 240-bit key of T-310 with about 2^{43} chosen IV / chosen ciphertext decryption queries, which need to be 2^{25} characters long. The time required is about 2^{65} CPU clocks to recover a 240-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 $s = 1$. It seems that long-term keys immune to correlations required by such attacks are quite rare, cf. Section 26, however such keys definitely exist, cf. Section 21.5.

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 12. For example with $\beta = 2^{-8}$ for key 27 the attacker can recover the full 240-bit encryption key in a time of $2^{8'}$ given about 2^{45} characters of encrypted data in the ciphertext-only scenario. A similar attack can be mounted against SKS V/1 cipher, cf. Section 20.3. It is extremely rare to see a ciphertext-only attack on a real-life government cipher[64].

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. At the same time in this paper we show that both KT1 class, cf. Thm. C.10.1 page 130 and KT2 class of long-term keys, cf. Thm. D.6.1 page 139, are mathematically proven secure against this sort of potentially devastating ciphertext-only attacks.

Finally we show that if the attacker can select the long-term key freely, the security of T-310 collapses very badly, cf. Section 22.10 and just a few thousand characters of encrypted data can be used to decrypt communications encrypted with T-310 in the ciphertext only scenario.

[64] 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. [70, 88].

References

1. Xavier Ameil, Jean-Pierre Vasseur, Gilles Ruggiu; *Histoire de la machine Myosotis,* Actes du septième Colloque sur l'Histoire de l'Informatique et des Transmissions, pp 95-125, 269 pages, editor IRISA-INRIA-Rennes, 2004, ISBN-13: 978-2726112816.
2. 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
3. Om Bhallamudi, Matteo Scarlata, Killian Davitt, *Python Tool for Test Vectors and Software Algebraic attacks on T-310 block cipher,* Software in development, 2017-2018, first version was developed by Om Bhallamudi and it can be found at http://www.nicolascourtois.com/software/codegen_417.zip. A new version by Killian Davitt is available here: https://github.com/KillianDavitt/T-310-AlgebraicAttack.
   A further update and our best version can be found at http://www.nicolascourtois.com/software/codegen_latest.zip.
4. Matteo Scarlata, Maria-Bristena Oprisanu, Killian Davitt, Kwok Cheng, Istvan Hoffer: *T-310 KT1, Strong and Weak Key Generation Python Tool,* 2017-2018, https://gist.github.com/gcky/ce83934101e0ccf0f8c47481c30597a9 and code by Istvan Hoffer here https://github.com/ihoffer/kt1_key. A further update is in preparation and will appear under http://www.nicolascourtois.com/software/keygen_latest.zip.
5. A. Biryukov, D.Wagner: *Slide Attacks,* In proceedings of FSE'99, LNCS 1636, pp. 245-259, Springer, 1999.
6. 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.
7. 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
8. 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
9. Cagdas Calik and Meltem Sonmez Turan and Rene Peralta: *The Multiplicative Complexity of 6-variable Boolean Functions,* https://eprint.iacr.org/2018/002.pdf
10. Don Coppersmith, *The development of DES, Invited Talk, Crypto'2000, August 2000.*
11. *Nicolas Courtois, Jörg Drobick and Klaus Schmeh: Feistel ciphers in East Germany in the communist era,* In Cryptologia March 2018, https://www.tandfonline.com/doi/full/10.1080/01611194.2018.1428835
12. 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/.
13. Nicolas Courtois, Michał Misztal: *Aggregated Differentials and Cryptanalysis of PP-1 and GOST,* In CECC 2011, 11th Central European Conference on Cryptology. In Periodica Mathematica Hungarica Vol. 65 (2), 2012, pp. 1126, Springer.

14. Nicolas Courtois, Guilhem Castagnos and Louis Goubin: *What do DES S-boxes Say to Each Other ?* Available on `eprint.iacr.org/2003/184/`.
15. 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.
16. 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`
17. Nicolas Courtois: *The Best Differential Characteristics and Subtleties of the Biham-Shamir Attacks on DES,* On `eprint.iacr.org/2005/202`.
18. Nicolas Courtois: *Feistel Schemes and Bi-Linear Cryptanalysis,* in Crypto 2004, LNCS 3152, pp. 23-40, Springer, 2004.
19. Nicolas Courtois, Theodosis Mourouzis, Anna Grocholewska-Czurylo and Jean-Jacques Quisquater: *On Optimal Size in Truncated Differential Attacks,* In post-proceeding of CECC 2014 conference, Studia Scientiarum Mathematicarum Hungarica.
20. 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 [21].
21. 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.
22. 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`
23. *Nicolas Courtois: Some algebraic cryptanalysis software by Nicolas T. Courtois,* `http://www.cryptosystem.net/aes/tools.html`.
24. *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.
25. 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.
26. 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.
27. Nicolas T. Courtois: *Low-Complexity Key Recovery Attacks on GOST Block Cipher,* In Cryptologia, vol. 37, issue 1, pp. 1-10, 2013.
28. 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`.
29. 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`.

30. Nicolas Courtois: *Algebraic Attacks on Combiners with Memory and Several Outputs,* ICISC 2004, LNCS 3506, pp. 3-20, Springer 2005.
31. Nicolas Courtois, Gregory V. Bard, David Wagner: *Algebraic and Slide Attacks on KeeLoq,* In FSE 2008, pp. 97-115, LNCS 5086, Springer, 2008.
32. 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.
33. 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.
34. 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
35. 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.
36. 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.
37. 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.
38. 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.
39. 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.
40. 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.
41. 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 [42].
42. 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
43. 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
44. 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.

45. 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.
46. 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.
47. Nicolas T. Courtois: *Decryption oracle slide attacks on T-310,* In Cryptologia, vol. 42, Iss. 3, 2018, pp. 191-204. http://www.tandfonline.com/doi/full/10.1080/01611194.2017.1362062
48. Nicolas T. Courtois, Maria-Bristena Oprisanu: *Ciphertext-only attacks and weak long-term keys in T-310,* in Cryptologia, vol 42, iss. 4, pp. 316-336, May 2018. http://www.tandfonline.com/doi/full/10.1080/01611194.2017.1362065.
49. Nicolas Courtois, Maria-Bristena Oprisanu and Klaus Schmeh: *Linear Cryptanalysis and Block Cipher Design in Eastern Germany in the 1970s,* will appear in Cryptologia in 2018.
50. Nicolas Courtois, Karsten Nohl, Sean O'Neil: *Algebraic Attacks on MiFare RFID Chips,* http://www.nicolascourtois.com/papers/mifar_rump_ec08.pdf.
51. Nicolas T. Courtois, Keith Jackson and David Ware: *Fault-Algebraic Attacks on Inner Rounds of DES* In eSmart 2010, European Smart Card Security Conference, 22-24 September 2010, Sophia Antipolis, French Riviera, with proceedings being published as a CD-ROM (slides presented). .
52. Nicolas T. Courtois and Louis Goubin: *An Algebraic Masking Method to Protect AES Against Power Attacks,* https://eprint.iacr.org/2005/204.pdf
53. Lauren De Meyer and Serge Vaudenay; *DES S-box generator,* In Cryptologia, vol. 41, iss. 2, 2017, pp. 153-171.
54. Itai Dinur, Adi Shamir: *Cube Attacks on Tweakable Black Box Polynomials,* In Eurocrypt 2009, LNCS 5479, pp. 278 - 299, Springer.
55. 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
56. Jörg Drobick: *T-310/50 ARGON,* a web page about T-310 cipher machines consulted 19 March 2017, http://scz.bplaced.net/t310.html
57. A web page which contains videos which show T-310 in operation and some examples of encrypted data: http://scz.bplaced.net/t310-fs.html
58. 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
59. Document MfS-Abt-XI-183, which is a documentation of SKS V/1 and contains a selection of pages extracted from a larger document known as MfS-020-Nr. 747/73, 1973.
60. Document MfS-Abt-XI-598, *T310 - LSZ - Technologie,* an original handwritten document about generation of long-term keys for T-310, 9 pages, written probably in 1981.
61. Arbeitsgebiet 113: *Sachstandbericht zur Arbeit am Chiffrieralgorithmus des Gerätes T 310/50,* MfS-020-XI/674/76, 51 pages, Berlin, 31 December 1976, also known as MfS-Abt-XI-532
62. ZCO: *Beschreibung der Chiffrieralgorithmen-Klasse ALPHA,* GVS 415/75, document not available to authors.
63. ZCO: *Charakterisierung der Booleschen Funktion Z,* handwritten document, MfS-020-XI/493/76, 24 pages, 1976.

64. 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
65. H. Feistel, W.A. Notz, J.L. Smith, *Cryptographic Techniques for Machine to Machine Data Communications,* Dec. 27, 1971, Report RC-3663, IBM T.J.Watson Research.
66. Horst Feistel: *Cryptography and computer privacy;* Scientific American, vol. 228, No. 5, pp. 15-23, May 1973.
67. William F. Friedman, *The index of coincidence and its applications in cryptology,* Department of Ciphers. Publ 22. Geneva, Illinois, USA: Riverbank Laboratories, 1922.
68. 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
69. 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.
70. J. J. Gillogly, *Ciphertext-only cryptanalysis of Enigma, In Cryptologia 19 (4):321413, 1995.*
71. *Jovan Golic, Cryptanalytic Attacks on MIFARE Classic Protocol,* In CT-RSA 2013, LNCS 7779, pp. 239–258, Springer, 2013.
72. Jovan Dj. Golic, Christophe Tymen: *Multiplicative Masking and Power Analysis of AES,* In CHES 2002, LNCS 2523, pp. 198-212, Springer, 2003.
73. 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.
74. Viet Tung Hoang, Phillip Rogaway: *On Generalized Feistel Networks,* In Crypto 2010, LNCS 6223, pp. 613-630, Springer 2010.
75. 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.
76. Lars R. Knudsen: *Truncated and Higher Order Differentials,* In FSE 1994, pp. 196-211, LNCS 1008, Springer.
77. 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
78. 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.
79. O.A. Logachev, A.A. Salnikov V. V. Yashchenko: *Boolean functions in coding theory and cryptography,* American Mathematical Society, 2012, Hardcover, 334 pages, translation of earlier book in Russian from 2004, translated by Svetla Nikova.
80. Jiqiang Lu, Jongsung Kim, Nathan Keller, Orr Dunkelman: *Differential and Rectangle Attacks on Reduced-Round SHACAL-1*, In INDOCRYPT 2006, pp. 1731, Springer 2006.
81. 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.
82. Mitsuru Matsui: *Linear Cryptanalysis Method for DES Cipher,* Eurocrypt'93, LNCS 765, Springer, pp. 386-397, 1993.

83. Theodosis Mourouzis: *Optimizations in Algebraic and Differential Cryptanalysis,* PhD thesis, under superivsion of Dr. Nicolas T. Courtois, University College London, January 2015, http://discovery.ucl.ac.uk/1462141/2/PhD_Thesis_Theodosis_Mourouzis.pdf.
84. 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.
85. Jacques Patarin, Valérie Nachef, Côme Berbain: *Generic Attacks on Unbalanced Feistel Schemes with Expanding Functions,* in Asiacrypt 2007: pp. 325-341, Springer, 2007.
86. Valérie Nachef, Emmanuel Volte, Jacques Patarin: *Differential Attacks on Generalized Feistel Schemes,* In CANS 2013: pp. 1–19.
87. Elisabeth Oswald, Stefan Mangard, Norbert Pramstaller: *Secure and Efficient Masking of AES - A Mission Impossible?,* eprint.iacr.org/2004/134
88. Olaf Ostwald, Frode Weierud: *Modern breaking of Enigma ciphertexts,* , In Cryptologia, vol. 41, iss. 5, pp. 395-421, 2017.
89. Ludovic Perret: *Gröbner bases techniques in Cryptography,* http://web.stevens.edu/algebraic/Files/SCPQ/SCPQ-2011-03-30-talk-Perret.pdf
90. 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.
91. J. Pieprzyk and G. Finkelstein: *Two effective nonlinear cryptosystem design,* IEE proceedings E - Computers and Digital Techniques, Vol. 135 Iss. 6, November 1988, pp. 325-335, ISSN 0143-7062.
92. 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.
93. Vincent Rijmen, Bart Preneel: *Cryptanalysis of McGuffin,* In FSE '94, pp. 353–358, Springer, 1994.
94. Matthieu Rivain: *On the Physical Security of Cryptographic Implementations,* PhD thesis, 22 September 2009, University of Luxembourg.
95. 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.
96. Klaus Schmeh: *The East German Encryption Machine T-310 and the Algorithm It Used,* In Cryptologia, 30: 3, pp. 251 257, 2006.
97. Maciej Skorski, *Shannon Entropy versus Renyi Entropy from a Cryptographic Viewpoint,* Eprint https://eprint.iacr.org/2014/967.pdf
98. Guangyan Song: *Optimization and Guess-then-Solve Attacks in Cryptanalysis,* PhD thesis, draft, will be presented at University College London in 2017.
99. 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.
100. Referat 11: *Kryptologische Analyse des Chiffriergerätes T-310/50. Central Cipher Organ, Ministry of State Security of the GDR, document referenced as 'ZCO 402/80', a.k.a. MfS-Abt-XI-594, 123 pages, Berlin, 1980.*
101. *T. Siegenthaler: Correlation-immunity of nonlinear combining functions for cryptographic applications,* In IEEE Trans. on Inf. Th. 30(5): (1984).
102. Ko Stoffelen: *Optimizing S-box Implementations for Several Criteria using SAT Solvers,* In FSE 2016, cf. also https://eprint.iacr.org/2016/198

103. VEB Steremat "Hermann Schlimme", *Gerätesystem SKS V/1, Gerät DE1*Zeichnungs-Nr. 310017, Band 2, also known as MfS-Abt-XI-415, and a.k.a. B 86/1-31/77, Berlin, 1976.
104. Michael Vielhaber: *Breaking ONE.FIVIUM by AIDA an Algebraic IV Differential Attack,* In https://eprint.iacr.org/2007/413.
105. Wikipedia article: *Birthday problem,* consulted 12 March 2017, https://en.wikipedia.org/wiki/Birthday_problem.
106. Wikipedia article: *Standard Deviation,* consulted 12 March 2017, http://en.wikipedia.org/wiki/Standard_deviation.
107. Zhegalkin polynomial, Wikipedia entry, https://en.wikipedia.org/wiki/Zhegalkin_polynomial

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.

BStU = Bundesbeauftragter für die Stasi-Unterlagen	Stasi Records Agency
MfS = Ministerium für Staatssicherheit	GDR Ministry of State Security (Stasi)
ZCO = Zentrales Chiffrierorgan der DDR	Central Cipher Organ of the GDR
ZCO = MfS Abteilung XI	ZCO is a.k.a. Stasi Department XI
GVS = Geheime Verschlußsache	Secret and confidential
Chiffrierverfahrens	encryption method
LZS, Langzeitschlüssel	long-term key (up to 94 bits)
Zeitschlüssel	current secret key (240 bits)
SpS, Spruchschlüssel	spelling key, the IV (61 bits)
Synchronfolge SYF	synchronisation seq. SYF, contains the IV
bestehend aus 25 FS-zeichen	consisting of 25 teletype characters
FS, Fernschreiber	teleprinter
ZG, physikalischer Zufallsgenerator	RNG, random number generator
PZG, Pseudozufallsgenerator	PRNG, pseudo-random number generator
Verschlüsselungseinheit	encryption unit
CE = Chiffiereinheit	Encryption Unit
Eingabeeinheit	input unit
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 Abbildung	cryptographic mapping / function
Komplizierungsalgorithmus	complication algorithm
Komplizierungseinheit	complication unit
Zwischenfolge	sequence
Kryptologische Eingenschaften	cryptologic characteristics
Schieberegister	Shift register
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

We provide a complete[65] description of key class KT1 following page 58 and Section 2.2 of Annex 1 and pages 114-115, and also Section 4.1 page 117 in [100].

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

$$\begin{cases} D \text{ and } P \text{ are injective} \\ P(3) = 33, P(7) = 5, P(9) = 9, P(15) = 21, P(18) = 25, P(24) = 29 \\ \text{Let } \ W = \{5, 9, 21, 25, 29, 33\} \\ \forall_{1 \geq i \geq 9} \ D(i) \notin W \\ \alpha \notin W \ \ \text{(note: cf. also Fig. 9.16 page 34)} \\ \text{Let } \ T = (\{0, 1, \ldots, 12\} \backslash W) \cap (\{P(1), P(2), \ldots, P(24)\} \cup \{D(4), D(5), \ldots, D(9)\} \cup \{\alpha\}) \\ \text{Let } \ U = (\{13, \ldots, 36\} \backslash W) \cap (\{P(26), P(27)\} \cup \{D(1), D(2), D(3)\}) \\ |T \backslash \{P(25)\}| + |U \backslash \{P(25)\}| \leq 12 \\ D(1) = 0 \\ \text{There exist } \ \{j_1, j_2, \ldots, j_7, j_8\} \text{ a permutation of } \{2, 3, \ldots, 9\} \ \text{ which} \\ \quad \text{defines } D(i) \text{ for every } i \in \{2, 3, \ldots, 9\} \ \text{ as follows:} \\ \quad D(j_1) = 4, D(j_2) = 4j_1, D(j_3) = 4j_2, \ldots, D(j_8) = 4j_7 \\ P(20) = 4j_8 \ \ \text{(note: this value is not any of the } \ D(i)) \\ (D(5), D(6)) \in \{8, 12, 16\} \times \{20, 28, 32\} \ \cup \ \{24, 28, 32\} \times \{8, 12, 16\} \\ P(6) = D(8), P(13) = D(7) \\ P(27) \neq 0 \bmod 4 \\ \forall_{1 \geq l \geq 9} \exists_{1 \geq i \geq 26} P(i) = 4 \cdot l \\ D(3) \in \{P(1), P(2), P(4), P(5)\} \\ D(4) \ \notin \ \{P(14), P(16), P(17), P(19)\} \\ \{P(8), P(10), P(11), P(12)\} \cap \{D(4), D(5), D(6)\} = \emptyset \end{cases}$$

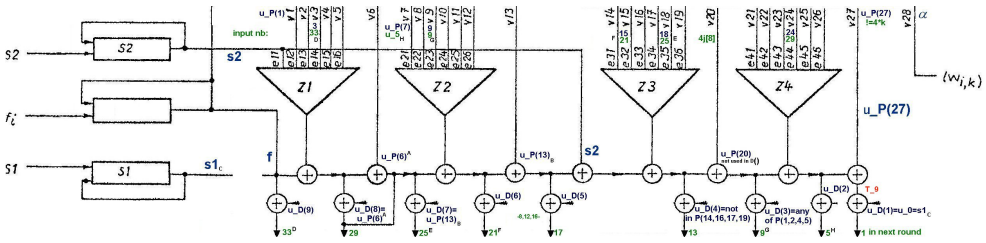


Fig. 2.27. Some observations about internal dependencies inside one encryption round ϕ, which hold for all KT1 keys, cf. also Fig. 9.16 page 34.

There exists approximately $2^{83.2}$ KT1 different keys total, cf. Section 8.6.

KT1 keys can be seen as a method to make the cipher have desired properties, an in principle are quite strong cf. later Thm. C.10.1 page 130, but not always, see for example Section 21.10, Section B.3 and Section B.5.

[65] An incomplete description which only included the conditions in page 58 of [100] was given in [96].

B.1 Observations About KT1 Keys

We have the following result:

Theorem B.1.1 (KT1 Cycling Theorem). For every key in the class KT1 if we replace the first value $d[1] = 0$ by $P(20)$ and we divide all values by 4, we obtain a permutation of the set $\{1, \ldots, 9\}$ with exactly one cycle.

Proof: We start with a real-life example. In key 14 we get 8,28,24,12,16,32,36,4,20. If we divide by 4 we get 2,7,6,3,4,8,9,1,5. This permutation of the set $\{1, \ldots, 9\}$ forms one single cycle: starting from 1: we have 1,2,7,9,5,4,3,6,8,1.
In the general case, and following the definition of KT1 in Section B, there exist $\{j_1, j_2, \ldots, j_7, j_8\}$ a permutation of $\{2, 3, \ldots, 9\}$ such that $D(j_1) = 4, D(j_2) = 4j_1, D(j_3) = 4j_2, \ldots, D(j_8) = 4j_7$ and $P(20) = 4j_8$ We claim that then the following permutation with 1 cycle [in order] is what we are looking for: $1, j_8$, $j_7, \ldots, j_2, j_1, 1$ which closes the cycle. Indeed 1 is mapped to j_8 due to $P(20) = 4j_8$, then we have $D(j_8) = 4j_7$ which implies that j_7 must follow position j_8 etc, finally j_1 is mapped to 1 due to $D(j_1) = 4$.

Corollary B.1.2. The number of possible sequences $\{j_1, j_2, \ldots, j_7, j_8\}$ with numbers 2-9 which can occur in KT1 keys is 8!=40320.

B.2 Origins and Consequences of the Cycling Property

KT1 keys "imitate" a sort of serial connection in SKS V/1:

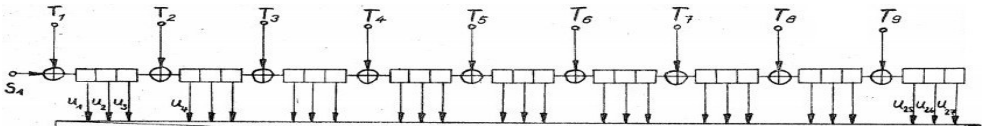


Fig. 2.28. A cascade of shift registers in an intermediate design which was neither SKS V/1 not T-310 cipher, from [61], cf. Fig. 6.39 page 145. This early version of T-310 KT1 keys had 3 bit blocks, later changed to 4 bit blocks.

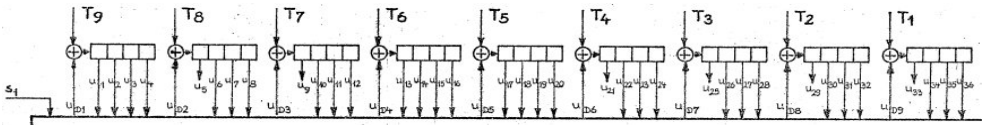


Fig. 2.29. Shift registers in T-310: here the fact that $D(x) = 4k$ in KT1 keys will insure a serial connection such as in Fig. 2.28 above with as starting point with S1 due to $D(1) = 0$ and ending with $P(20)$, cf. also Thm. B.1.1 page 119.

Moreover it is possible to specify in which exact order this serial connection of all 4-bit shift registers at Fig. 2.29 is handled:

Theorem B.2.1 (KT1 S1 Ordering Theorem). For every key in the class KT1, and for the key bit S1 which is used in encryption round $t \geq 1$,
initially S1 is XORed to T9 in round t and it becomes U_1 or input $u_{4\cdot 0+1}$ of round $t+1$, then it becomes input $u_{4\cdot 0+4}$ of round $t+4$,
then it is XORed to T_{10-j_1} end of round t+4 and becomes bit U_{j_1} or input $u_{4\cdot j_1+1}$ of round $t+5$, then it becomes input $u_{4\cdot j_1+4}$ of round $t+8$,
then it is XORed to T_{10-j_2} end of round t+8, etc,
and finally it gets XORed to T_{10-j_7} end of round t+56, and becomes U_{j_7} or input $u_{4\cdot j_7+1}$ of round $t+57$, then it becomes input $u_{4\cdot j_7+4}$ of round $t+58$,
finally it will enter input 20 of T or input $u_{P(20)}$ of round $t+59$.

Table 26. Order of treatment of T_i in Thm. B.2.1

LZS nb	α	T_i order
14	30	$T_9T_2T_4T_7T_6T_5T_1T_3$
21	1	$T_9T_6T_5T_2T_1T_7T_3T_4$
26	4	$T_9T_7T_3T_8T_4T_5T_2T_6$
30	4	$T_9T_3T_6T_1T_8T_7T_5T_2$
31	2	$T_9T_7T_5T_1T_2T_3T_4T_6$
32	4	$T_9T_6T_3T_5T_8T_4T_7T_1$
33	3	$T_9T_1T_6T_4T_7T_8T_2T_3$

Immediately we see that:

Corollary B.2.2. For any bit u_i, $1 \leq i \leq 36$ after $j-th$ block cipher round it can be expressed as a sum of just **one** specific key bit $S1$ and a XOR of up to nine T_k from the preceding 58 rounds (T_9 always coming first).

This observation can potentially lead to attacks of the following type:

Conjecture B.2.3. We conjecture that for some LZS the bit T9 can be approximated by a Boolean function of the key and IV bits in the last K rounds for some $K > 0$. Then by subtracting this Boolean function under a key guess the attacker can recover $S1$ key bits under a partial key guess and in a ciphertext-only scenario.

To illustrate this idea, in the table below we show example of keys which exhibit a bias for certain combinations of TiTj distant by exactly 4 rounds. It should be added that each of these two bits T_i is not biased individually.

Table 27. Examples of keys where a sum of T_i and T_j four rounds later is biased.

LZS nb	D	P	sequence	$\lvert P-1/2\rvert$
464	24,8,4,12,32,28,0,20,16	4,33,16,17,20,5,14,9,3,10,18,12,7,21,13,23,25,28,36,24,15,26,29,27,32,11	T_6, T_4	$2^{-6.9}$

B.3 On the Choice of α in KT1 Keys

Our Thm. B.2.1 shows that depending on the choice of α, the keystream bits are extracted potentially after XORing just T_9 to $S1$, or after XORing as many as nine different T_i. We believe that if this number is high, it may be particularly difficult to design a correlation attack. In Fig. 26 we show how this works for various historical keys. The result is that the oldest key 14, α was potentially the most secure choice. Here bit 30 is generated after using all of $T_9T_2T_4T_7T_6T_5T_1T_3$ in some 58 consecutive rounds cf. Thm. B.2.1.

In contrast, in all other historical keys, bit α is extracted as $S1 \oplus T_9$ in one of the few previous rounds. This is maybe a secure choice w.r.t. correlation attacks on $s2$ and the complexity of different bits of $T()$. However it seems to be the weakest possible choice for correlation or other attacks on $S1$, provided that some bias or weakness on T_9 bits would make such attacks possible in the KT1 case (or for some weaker keys cf. Table 27).

Remark. This bit 30 then enters $T()$ at specific wire v_{20} cf. Fig. 7.12.

B.4 Symmetries Which Preserve KT1 Keys

We have the following fact:

Theorem B.4.1 (KT1 Symmetry Property 1). For every key in the class KT1 if we permute in arbitrary way $\{P(1), P(2), P(4), P(5)\}$ we always get a valid KT1. The same applies to $\{P(14), P(16), P(17), P(19)\}$ and also to the set $\{P(8), P(10), P(11), P(12)\}$ and also to the pair $\{P(21), P(23)\}$.

Proof: By careful inspection we see that the definition never makes any distinction between these indices.

We also have:

Fact B.4.2 (KT1 Symmetry Property 2). For every key in the class KT1 if and for certain well chosen pairs a, b, if the numbers a, b appear at two locations $a = P(x)$ and $b = P(y)$ and we swap these pairs, we always obtain a valid KT1 key (this ignoring completely α or being KT1 for at least for some α). There are 112 such pairs a, b which are 1,[2-3,6-7,10-11];2,[3,6-7,10-11];3,[6-7,10-11];6,[7,10-11];7,[10-11];10,[11]; 13,[14-15,17-19,22-23,26-27,30-31,34-35];14,[15,17-19,22-23,26-27,30-31,34-35];15,[17-19,22-23,26-27,30-31,34-35]; 17,[18-19,22-23,26-27,30-31,34-35];18,[19,22-23,26-27,30-31,34-35];19,[22-23,26-27,30-31,34-35]; 22,[23,26-27,30-31,34-35];23,[26-27,30-31,34-35];26,[27,30-31,34-35];27,[30-31,34-35];30,[31,34-35];31,[34-35];34,[35];

Justification: By careful inspection we can see that the KT1 definition never makes any distinction between the two values for a large number of pairs above. For example if we look at set $13, \ldots, 36$ which explicitly appears in the KT1, and exclude all the other elements potential values of $P()$ which are explicitly used or name in the definition, such as all multiples of 4, 33, 21, 25, 29. We are left with 13,14,15,17,18,19,22,23,26,27 which can be swapped freely, which gives 45 pairs. Our computer simulations show that there exist 67 additional pairs and 112 total.

Definition B.4.3 (Normal form for the KT1 keys).
We say that a KT1 key is in a normal form if for each 4 sets of Thm. B.4.1 the values always appear in the increasing order inside P(). In addition for each of the 112 sets of Thm. B.4.2, the numbers x, y should be in an increasing order. For example if $P(x) = 13$ and $P(y) = 14$ we must have $x < y$.

Remark: This property allows to classify all KT1 keys in a reduced number of distinct equivalence classes with representatives being the keys in normal form.

B.5 Discussion KT1 Keys: Strong or Weak?

It is clear that KT1 keys prevents many strong attacks, see for example Thm. C.10.1 which has prevents the attacks of Section 20.2 and Appendix C.12-C.12 and Section 22.9.

Is KT1 class always secure? The answer is not always. In Section 21.10 we show numerous examples of keys which satisfies ALL the KT1 conditions above and which is pathological w.r.t Linear Cryptanalysis. A large proportion of all KT1 keys has this property. Following Section 21.9 about $1/8 \cdot 1/36 \approx 0.3\%$ of keys are such that $D(2) = 4$ and $P(27) = 6$ and therefore must be weak w.r.t. Linear Cryptanalysis. In a future update of this paper we will show that about 3.0% of all KT1 keys are weak w.r.t. Linear Cryptanalysis. In Section 22 we study the more general question of arbitrary weak LZS, not only in the KT1 case.

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 that it must be a bijection cf. pages 47 and 56 in [100]. It appears that if ϕ is required to always be bijective, this will be for security reasons, not for purely functional encryption reasons. Bijective ϕ allows to maintain high entropy of the state u_t at any time t. Moreover the opposite seems to lead **almost always** to insecure ciphers, see Conj. 22.14.1 page 94 and Sections 20 and 22.10.

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[66] to extract keys from SIM cards for certain mobile phone operators, and these attacks these keys exploit precisely vanishing differentials cf. [6, 7, 41, 34]. 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 [15, 45] 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 [14].

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. [14, 17] and they do not exist in GOST cf. slide 255 in [41]. These properties have been carefully engineered by the designers or DES, cf. slide 31 in [16] and [10, 14].

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

[66] 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. [6], and is also described in page 6-15 in [7] and in Section 13.1 slides 249-255 in [41]. Moreover there exist more efficient variants of this attack cf. [34] which we have developed ourselves, cf. slide 230 in [34] and [69], 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.

could be in the best case. However knowing that only up to 9 inputs of any of the four Boolean functions $Z()$ are repeated elsewhere (due to being an output of $D()$), it is hard to imagine an impossibility result [67] 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 42. 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 12.3.

Further Remarks. There exist also vanishing differentials if we are allowed to flip some key bits, cf. Thm. 12.4.1 page 44.

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. 12.6.1.

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 aperiodic 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 [67, 81]] 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 [100]. 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[68] large.

[67] 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 $\neq 0$.

[68] 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 [20, 21].

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

It is not sufficient to say that ϕ should be a bijection, cf. [100]. 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 E.6, 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 [100]. 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 (and also in [48]) 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. Then in Section D.6 we provide a proof for the KT2 case.

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. 3.31 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. 3.33 we will show it can always be done for all keys if type KT1.

If we put all the outputs of **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 9.6 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.

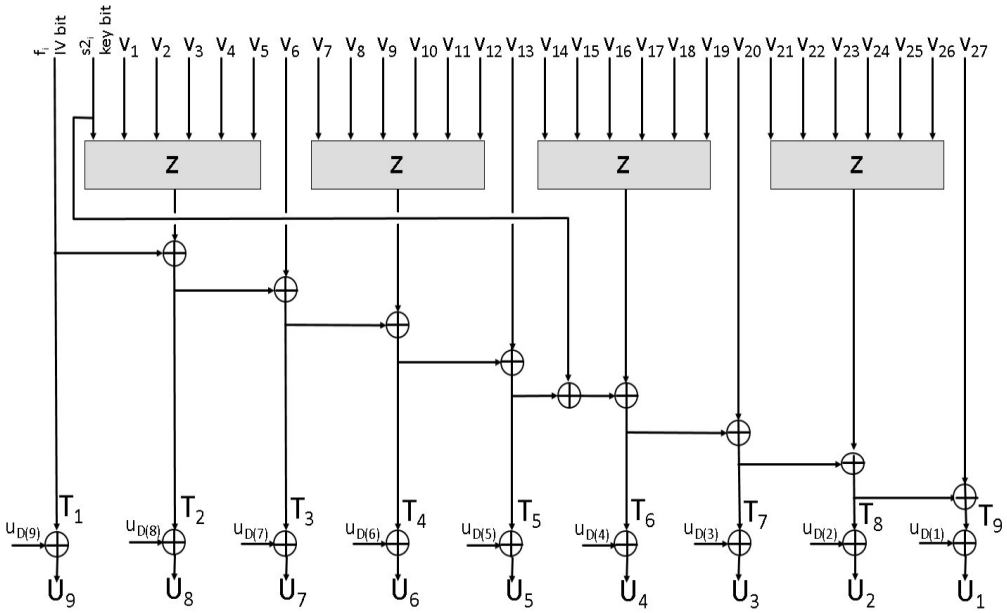


Fig. 3.30. The internal structure of $T()$ inside one round of T-310, cf. Fig. 9.17.

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. 3.31 below. We recall the necessary definitions and make some observations:

```
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
Z1 inputs are s2 and 8 4 33 16 31
Z2 inputs are 5 35 9 3 19 18
Z3 inputs are 7 21 13 23 25 28
Z4 inputs are 24 15 26 29 27 32
```

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 the KT1 keys which mandate that $D(8) = P(6)$.

We are now ready to explain how inversion can be performed. On Fig. 3.31 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 the KT1 keys cf. [96]) 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 the KT1 keys [96], 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}.

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. [96]. 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 $u_{36} = u_{D(3)} \oplus u_{D(4)} \oplus U_3 \oplus U_4$.

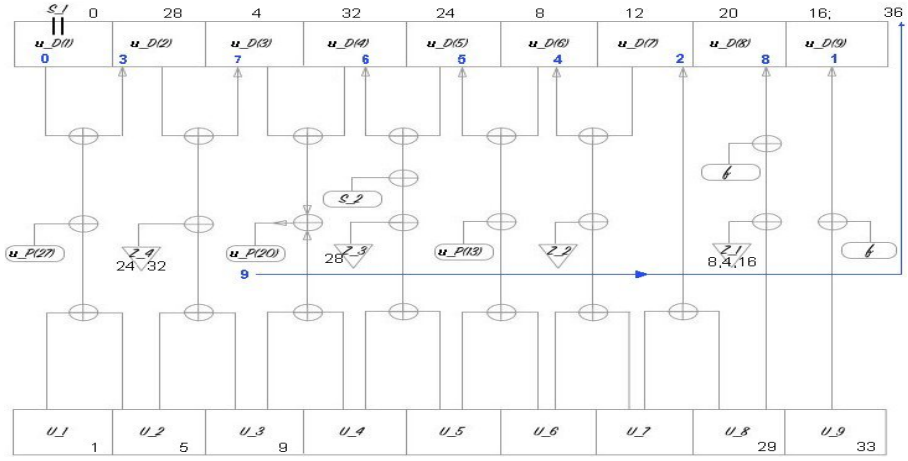


Fig. 3.31. Example of inversion of ϕ for key number 26.

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. 5.7, we recall the compact description of the ϕ function in Section 7.1.

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

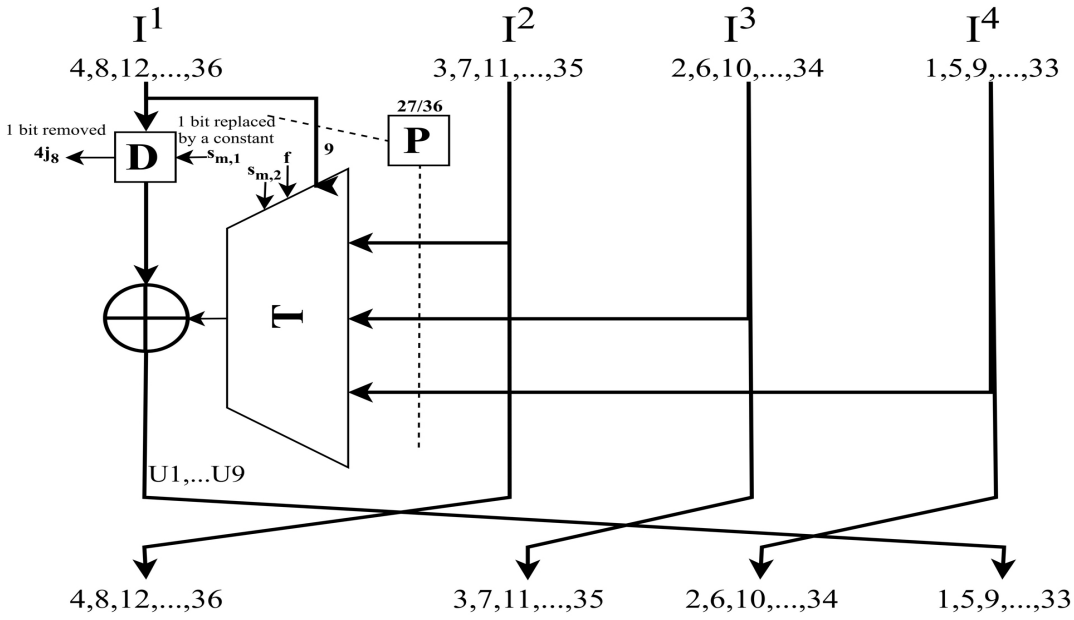


Fig. 3.32. T-310 for the KT1 keys.

Now we recall how all this translates into a set of multivariate equations when $D(1) = 0$, cf. Section 9.6 or Section C.8. We will number these equations (1-9).

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

We have the following result:

Theorem C.10.1 (KT1 Invertibility Theorem). For every key in the 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 the 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. 3.32 above. Thus we only need to show how to compute $u_{D(1-9)}$ and also $u_{P(20)}$ 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 the 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 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: $u_{D(9)} = U_9 \oplus f$.

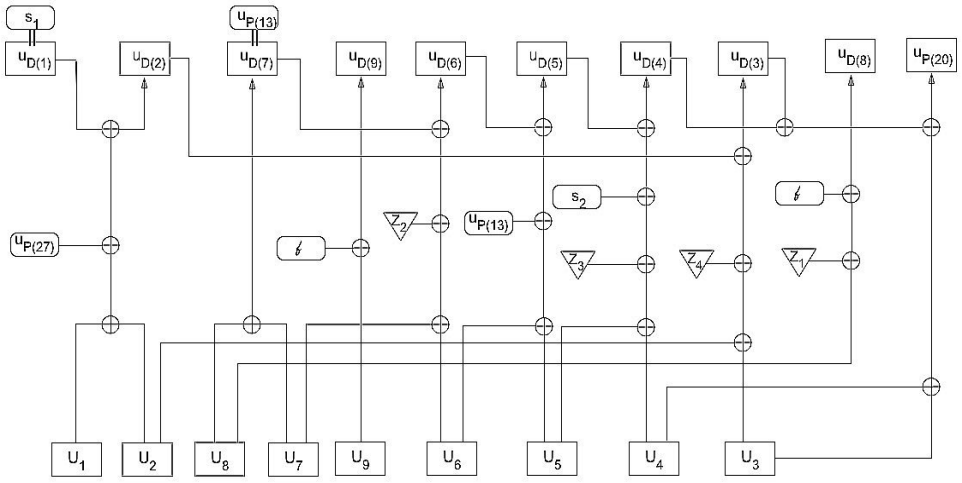


Fig. 3.33. A method for inverting ϕ which works for ANY key of type KT1.

`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 there are 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 the 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): $u_{D(6)} = U_6 \oplus U_7 \oplus u_{D(7)} \oplus Z_2(u_{P(7-12)})$.

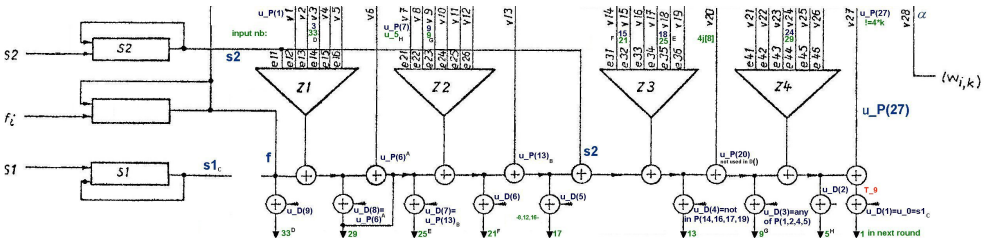


Fig. 3.34. A view of one round of T-310 encryption in the KT1 case.

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 there are 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, $P(20), P(6)$ due 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 the 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 the 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 no, because $D(8) = P(6)$ and P is 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, as described in Sections 18 and 20.

C.11 Post-Scriptum to Our Proof - Linear Ordering on 4k Bits and Structural Properties with KT1 Keys

In our proof we have the following order of computing bits:

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

which ordering is a strong structural property of the KT1 keys.

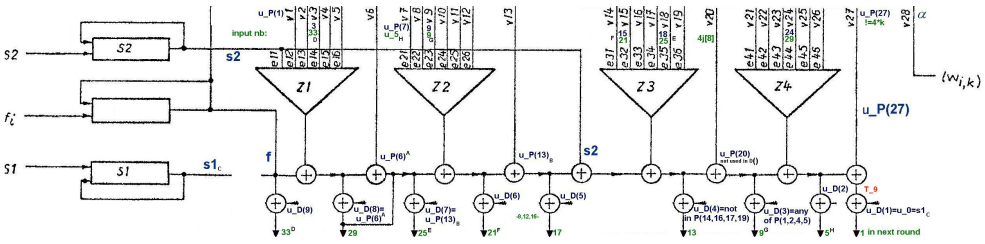


Fig. 3.35. A view of one round of T-310 encryption in the KT1 case.

If follow our proof, this inversion can also be seen as computing bits with numbers $4k, k = 0-9$ in a specific order, which are the only bit not immediately known in this process. Therefore the inversion imposes a linear order on bits $4k, k = 0-9$. For example going back to our example, key 26, we can rewrite the order in the proof paying attention only to numbers being multiples of 4 and without repetitions:

```
0 D1 P27 D9 D2 D7 P13 Z2 D6 D5 Z3 D4 Z4 D3 Z1 D8 P20
0 16 28 12 8 24 32 4 20 36
```

We can compare it to the following 2 orders which also worked for key 26, the solution is not completely unique:

```
0 16 12 28 8 24 32 4 20 36
0 28 12 16 8 24 32 4 20 36
```

In fact there exist another "ordering" structural property which concerns also numbers $4k, k = 0 - 9$ which is mandated for all KT1 keys and which is closely related yet not completely the same.
More precisely, we recall the following conditions from the definition of KT1 keys:

> There exist $\{j_1, j_2, \ldots, j_7, j_8\}$ a permutation of $\{2, 3, \ldots, 9\}$ which defines $D(i)$ for every $i \in \{2, 3, \ldots, 9\}$ as follows:
> $D(j_1) = 4, D(j_2) = 4j_1, D(j_3) = 4j_2, \ldots, D(j_8) = 4j_7$
> $P(20) = 4j_8$ (which value is not any of the $D(i)$).

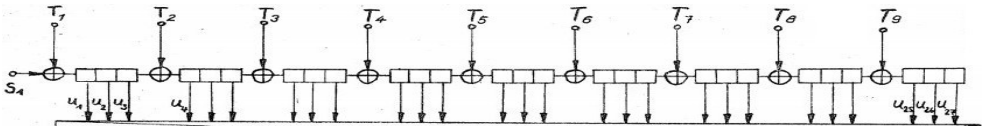


Fig. 3.36. A cascade of shift registers in an intermediate design which was neither SKS V/1 not T-310 cipher [61].

For example for key 26 we have $j_1 = 3, j_2 = 7, j_3 = 2, j_4 = 6, j_5 = 5, j_6 = 8, j_7 = 4, j_8 = 9$ and $4j_1, 4j_2, 4j_3, \ldots 4j_8$ is $12, 28, 8, 24, 20, 32, 16, 36$ which does not contain 4, but 4 could be added at the beginning or at the end to complete the cycle, cf. Thm. B.1.1 page 119. Overall we obtain:

```
12 28 8 24 20 32 16 36 4
```

which is QUITE SIMILAR but different than the 3 solutions obtained above, and there are substantial differences regarding numbers 16 and 4. The difference can be explained by the fact that our proof contains the $D(i)$ as a subsequence in the following order:

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

Discussion. Real-life T-310 keys impose two (or more) different ways to order the set $D(i)$ some of which are relevant in our proof and some aren't. Both ordering the $4k$ in the proof and the one due to the j_i are vaguely similar and seem related. It seems however that the conditions about the j_i are **not** at all required in our proof which simply does NOT use them. It is therefore not clear if imposing these strict conditions about the j_i is really important for the security of KT1 keys. This second linear order on $4k$ due to the j_i and a related [69] cyclic order seems to exist **not** exactly in order to allow inversion, but for reasons which are related to the history of T-310 which was built by modifying the earlier SKS V/1 cipher, see Section F.

[69] The ordering on j_i is further studied in Appendix B.1. In fact for KT1 keys, this total ordering can be modified to be cyclic, cf. Thm. B.1.1 page 119.

C.12 Our KT1 Proofs vs. Correlation Vulnerabilities

In contrast the conditions $P(6) = D(8)$ and $P(13) = D(7)$ are clearly somewhat needed or at least are used in the proof ot Thm. C.10.1, cf. also Fig. 2.27.

These two conditions are EXTREMELY important, see Section 22.9 to see that there is devastating ciphertext-only attack if just one of these is not satisfied. More precisely in Section 22.9 we see that different conditions of type rather $P(5) = D(8)$ or rather $P(6) = D(7)$ would lead to very strong correlations.

Properties such as studied above open avenues for further correlation attacks. For example if we look at the structure of Fig. 3.36 and related Thm. B.1.1 we see that even though S1 will be masked by one state bit at one step, this masking should be undone later by XOR with another state bit which could be correlated (cf. Table 27 for actual examples when this happens).

C.13 Some Basic Results on KT1 Keys

Our student Matteo Scarlata has proven the following result:

Theorem C.13.1 (KT1 Impossibility Result). For every KT1 key we have $D(2) \neq 8$. Moreover, for every $z > 1$ we have $D(z) \neq 4z$.

Proof: Following the definition is Section B there exist $\{j_1, j_2, \ldots, j_7, j_8\}$ a permutation of $\{2, 3, \ldots, 9\}$ which defines $D(i)$ for every $i \in \{2, 3, \ldots, 9\}$ as follows:

$$D(j_1) = 4, D(j_2) = 4j_1, D(j_3) = 4j_2, \ldots, D(j_8) = 4j_7$$

Now if $D(2) = 8$ then $\exists k \in \{2, \ldots 8\}$ such that $D(j_k) = 4j_{k-1}$ with $j_k = 2$ and $4j_{k-1} = 8$. Thus $j_k = j_{k-1}$ which contradicts the assumption that j is a permutation. More generally if $D(z) \neq 4z), z > 1$ then $\exists k \in \{2, \ldots 8\}$ such that $D(j_k) = 4j_{k-1}$ with $j_k = z$ and $4j_{k-1} = 4z$. Thus $j_k = z = j_{k-1}$ which contradicts the assumption that j is a permutation.

More we have the following result:

Fact C.13.2 (KT1 Exclusion Rules). We cannot have $D(2) = 4k + 1$ unless it is 13 or 17. Overall there are 168=8*21 possible values for pairs $(D(2), P(27))$, where we can have any combination of 8 case $D(2) = 4k$ and $D(2) \neq \{0, 8\}$, and $P(27)$ can take any of 21 permitted values between 1 and 36, which are defined by excluding nine multiples of 4 and six values in set $W = \{5, 9, 21, 25, 29, 33\}$.

Proof: We have $D(2) = 8$ not possible due to Thm. C.13.1 and we also recall that KT1 specifies $P(27) \neq 0 \bmod 4$, cf. Section B. It remains to see why six $\{4k + 1\}$ values 5,9,21,25,29,33 are forbidden, even though 1, 13 and 17 are actually allowed. This is because P is injective and all these 6 values are already taken due to $P(3) = 33, P(7) = 5, P(9) = 9, P(15) = 21, P(18) = 25, P(24) = 29$, cf. Section B. Computer simulations show that all the 168 cases do happen for actual valid KT1 keys.

D A Study of KT2 Keys

D.1 Definition of KT2 Keys

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

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 [100] is excessively complex, cf. pages 59-60,114-115 and 117 in [100] 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 40 and many other such results which can be found in [100]. 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 8.6.

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. 9.16 as this figure comes from the original specification of T-310 cipher in [100] and also those which avoid some particularly bizarre keys from [55] 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 of 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 the 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 [100].

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[70] page 60 in [100]. We recall that we have in the general case the following relations which can be seen as a standard compact way to write ϕ in a certain type of "triangular" order which makes the computations easy.

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

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 the 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 definition 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 the 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 $B = (b_{ij})$.

[70] 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 [100], 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 140 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 the 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 matrix 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. 3.31 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 the 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 Non-Standard Long-Term Keys

E.1 KT2 and Similar Keys vs. Chosen Long-Term Key Attacks

The known sources [100, 55] 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 28. Examples of keys which would be of type KT2 except for the matrix rank condition M_9.

LZS 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 65. Additional similar keys which satisfy less conditions can be found in Table 9 page 60.

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[71] 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 9.6 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.

[71] One particularity of this key is that it has $\underline{\mathbf{D}} : \mathbb{F}_2^{9+2} \rightarrow \mathbb{F}_2^9$, cf. Section 5.6.

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

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:

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

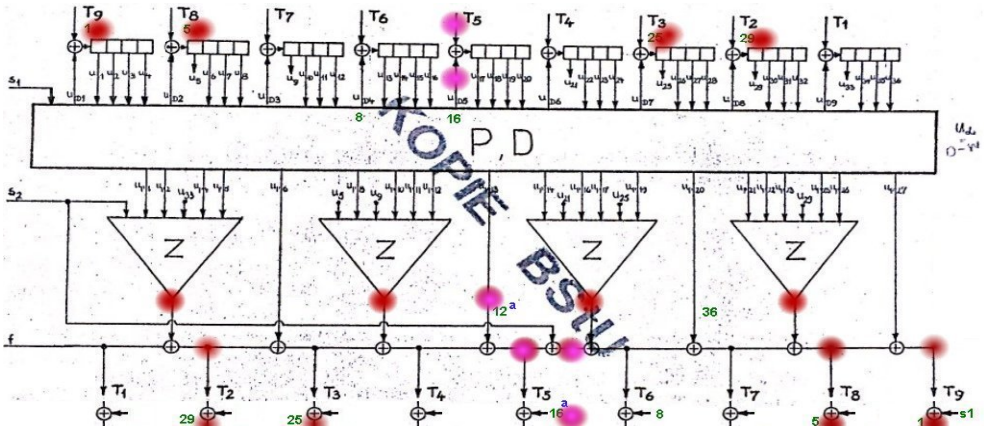
which is also shown in Fig. 5.37 below. Here we define $U^{(a)}$ as all bits being 0 except two $u_{D(5)} = u_{16} = 1$ and $u_{P(13)} = u_{12} = 1$. Then let $U^{(a')}$ is such that all bits are 0 except $u_{D(4)} = u_8 = 1$ and $u_{P(20)} = u_{36} = 1$. Finally let $U^{(b)} = \phi(0,0,0;U^{(a)})$. 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:

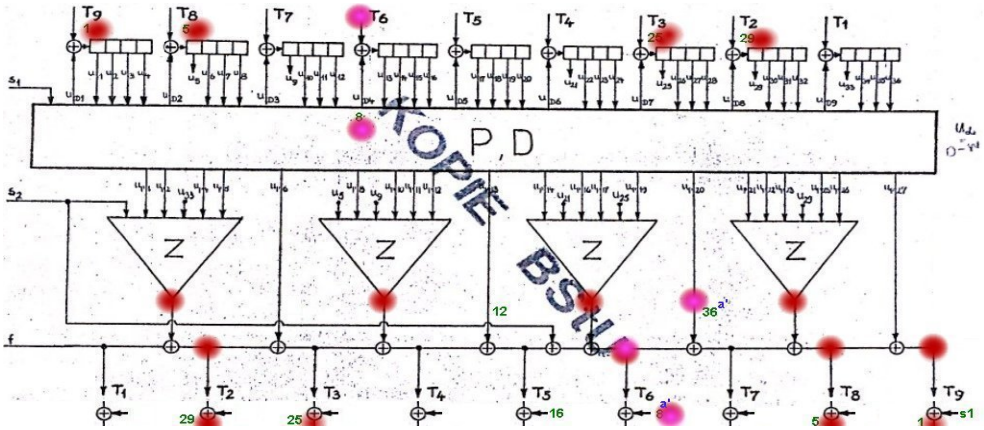


Fig. 5.37. A collision for ϕ with key 207: active wires at 1 are marked with color dots.

E.6 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 [96] is not quite correct and not compliant with the original document [100]. More precisely the author of [96] has forgotten to transcribe some additional conditions of Section 4.1 in Appendix 1 of [100] such as $P(3) = 33$ and few other conditions. In order to show that the spec of [96] 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 [96].

```
935: 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. At this moment it not clear if this key which we will call key 935, is weak in any way. The lecture of comments which appear in [55] just after key 33 suggests that this key may be weak for some cyclic properties such as repeated word test, which properties are however not clearly specified in [55].

E.7 Another Non-Standard Key

Here is another example of non-standard bijective key which is different than other keys and without trying to make it weak in any way. It has been generated to see if the complex conditions specified by the designers of T-310 cf. Appendix B and D.1 are necessary to obtain a secure cipher. Possibly most of these conditions are NOT needed. This key satisfies the first 4 conditions of KT2 keys of Section D.1 and we call such keys KT2f.

```
731: P=3,26,33,31,32,4,5,34,9,18,14,28,27,7,21,2,30,25,
35,8,22,20,36,29,10,15,17 D=28,0,16,12,36,32,24,4,20
```

E.8 The Special Key 16 and SKS Cipher

This part was moved to later Appendix F.2.

F SKS V/1 Cipher: A Predecessor of T-310

Numerous sources indicate that T-310 is a descendant of an earlier cipher machine known as SKS V/1, see for example pages 10,11,22,41,47-48,120,122 in [100] and [58]. Both belong to a larger family of ciphers known as ALPHA, cf. [100]. In this section we summarize the main characteristics of SKS as opposed to T-310 following information obtained form Jörg Drobick, and from [58, 59, 100].

1. SKS V/1 is a sort of Feistel cipher with 3 branches cf. Fig 6.38 and later Fig 6.41 page operating on $27 = 3 \cdot 9$ bits, while T-310 has 4 branches and operates on $36 = 4 \cdot 9$ bits, cf. Fig. 5.7 page 19.

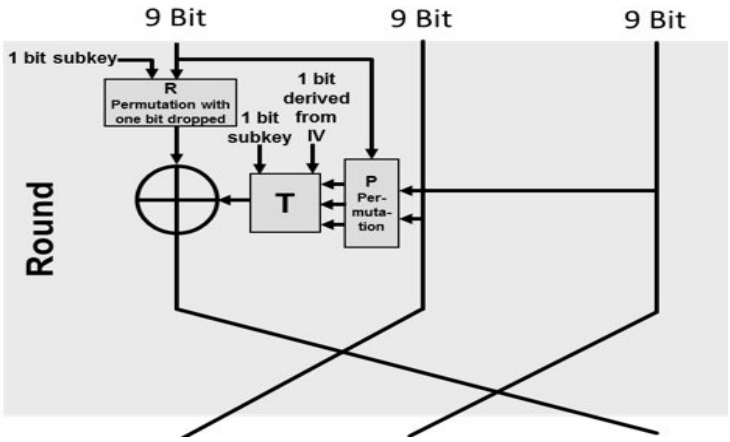


Fig. 6.38. Outline of SKS cipher with 3 branches and 27 bit block size.

2. In both cases 9 new bits are created in one step, and the overall cipher operation is extremely similar, cf. Fig. 6.40. Both designs have exactly the same $T()$ component which can be defined as $T : \{0,1\}^3 \times \{0,1\}^{27} \rightarrow \{0,1\}^9$ cf. Section 9.
3. The key $S1, S2$ in SKS has $208 = 2 \cdot 104$ bits, and in T-310 it has $240 = 2 \cdot 120$ bits (earlier version had 208 cf. [61]).
4. SKS also has a variable initial constant U_0 which is part of the long-term key in SKS and which is fixed once for all in T-310. It appears that U_0 has 7 inputs m_{1-7} and 27 outputs r_{1-27}, and that input m_1 is used in case of reset, cf. page 39 in [103].
5. SKS also has $P : \{1-28\} \rightarrow \{1-27\}$ instead of $P : \{1-27\} \rightarrow \{1-36\}$ in T-310.

6. SKS long-term keys have no D, instead it has a permutation on 9 or 10 wires called sometimes Q and sometimes R^{-1}.
7. More precisely, in all generality they operate on 10 wires Fig. 6.40 and in [58] and inputs would be called TR1-TR9, cf. page 12 or 100 in [61] and outputs are T1-9 which is NOT the same as later specifications of T-310 where T1-9 are not permuted by Q.
8. Both algorithms belong to a general class called ALPHA, and it appears that early versions of SKS V/1 from 1973-75 were known under another name of OPERATION, cf. for example the 1973 document [59] where Q and R had 10 wires.
9. Later T-310 also has something quite similar: see Thm. B.1.1 page 119.
10. Moreover there exists an early (not final) version of T-310 cipher from 1976 which is very much like SKS: it has 208 bit key it has 2 permutations P and Q on 27 bits and 9 bits respectively, cf. [61]. In this early version of T-310, is neither exactly SKS V/1 nor the actual T-310, we would have R9=1 and we have a permutation on 9 wires cf. Fig. 6.39 and [61].

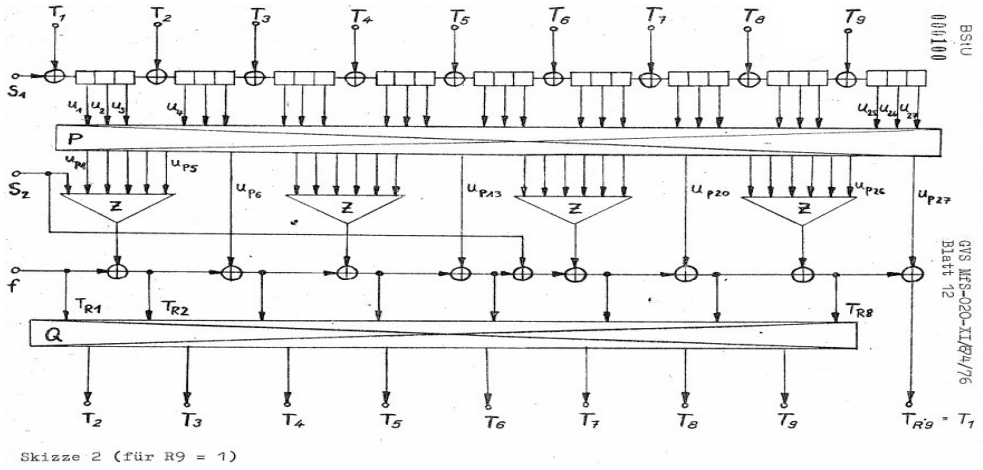


Fig. 6.39. An early original drawing of the encryption unit from [61] which is neither exactly SKS V/1 nor the actual T-310, but a sort of intermediate version.

11. The aperiodic sequence f_i is generated differently, the LFSR is a maximum-period LFSR with 52 bits with equation $f_i = f_{i-52} \oplus f_{i-49}$ instead of 61 in T-310. It is initialized to an initial value (IV) using 168 random bits obtained from a physical random number generator. Then at each clock i [which represents one round of the SKS block cipher] the f_i are expanded to 104 bits through duplication of certain bits as follows:

$$FU_{i,k} = f_{i+k+52\nu}$$

at step i with $k = 1 \ldots 104$ and $\nu = 0$ when $k = 1..51$, then $\nu = 1$ when $k = 52..103$, finally $\nu = 2$ for $k = 104$.

12. T-310 has a periodic key with period 120 which is different and relatively prime to 127 which is the period for extracting internal keystream bits. SKS has a stronger periodic structure: both the key bits repeat every 104 steps, and the extraction of 1 bit from the state of the block cipher happens every 104 steps after the starting interval of $104p$ rounds, where p in an integer known as "Startzeitpunkt".

$$w_{i,k} = \Psi(f_{i+k\nu*52}; S1_k; S2_k, U_0, U_i; k-1, P; Q; Z1; Z2; Z3; Z4)$$

where Ψ corresponds to extracting bit $P(28)$ from the cipher state.

13. The choice of α is replaced by the choice of $P(28)$.

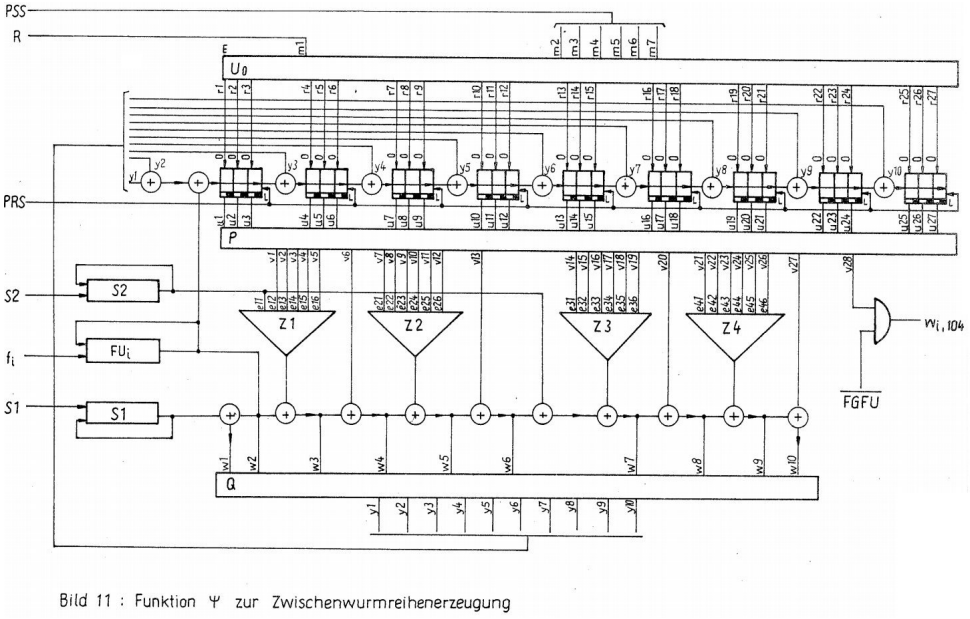


Fig. 6.40. An original drawing of the SKS encryption unit a.k.a. "Komplizierungseinheit" or a "complication unit", cf. [58].

14. SKS V/1 differs substantially from T-310 in handling the actual data/text encryption. In SKS, plaintext is divided in blocks of 41 bits expanded with 6 parity bits using a linear Error Correcting Code, and encrypted with a binary XOR with a keystream sequence W_i a.k.a. AR = "Additionsreihe". This is a lot simpler and also clearly weaker, than in T-310 which uses this peculiar "double one-time pad" method cf. Section 16. We have:

$$W_i = w_{i,104}$$

These bits are used in blocks of 47 to encrypt data blocks with parity.

F.1 Unorthodox Feistel Ciphers - Comparison of T-310 to the SKS Cipher

In the previous Section 4.5 we wrote that there is no evidence that a simple bijective permutation of wires **D** applied to I^1 would ever be used in a real-life cipher T-310 and nothing else. However in a more relaxed sense it was. If T-310 is used with a special long-term key 16 it will behave like an earlier SKS cipher, predecessor of T-310, see Appendix F and E.8. There are however two differences: SKS is a Feistel with 3, not 4 branches, and moreover SKS (already) ignores one bit of I^1 which is replaced by one bit of the key.

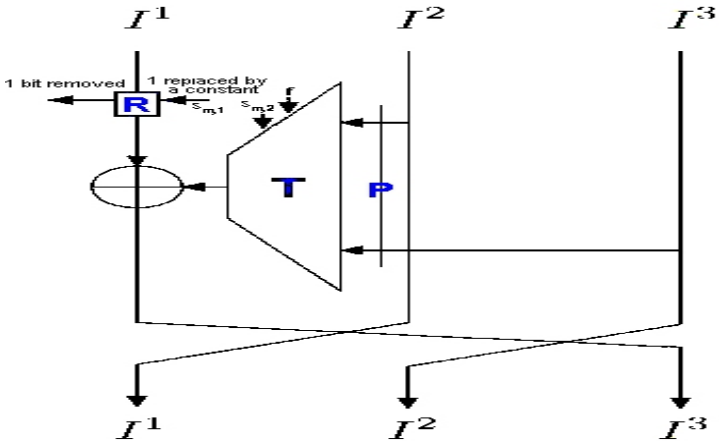


Fig. 6.41. SKS cipher is in every way similar to T-310 with KT1 keys except it has 3 branches and the state has 27 bits.

The fact the SKS systematically ignores one bit of I^1 is extremely significant, cf. Sections 5.2 and 5.4. As with T-310 a permutation is obtained due to careful use of 9 bits of I^1 in P, and would requires one to re-visit the bijectivity/security proofs Appendix C or SKS would be prone to some potentially quite strong ciphertext only attacks such as in Section 20.

F.2 The Special Key 16 and SKS Cipher

The following non-standard key is described in [55] as a key number 16 for a special version of T-310 cipher machine called T-310/51 SAGA instead of the usual T-310/50 ARGON. We ignore what the exact difference between these machines might be but in [55] 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"ussel 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 [100] 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[72].

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 42. At the same time it still has the basic 6 properties [73] regarding the set W of Section B such as $P(3) = 33$. This leads to the following situation which we depict in Fig. 6.42 below.

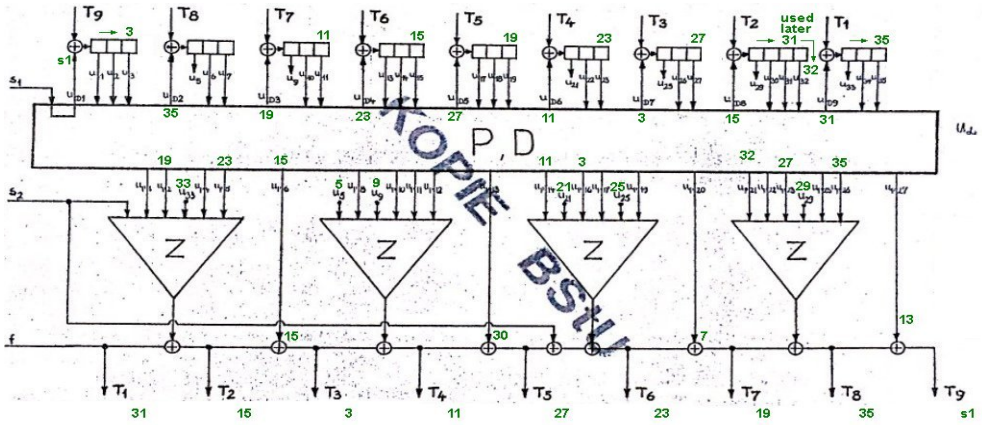


Fig. 6.42. T-310 round function based on page 119 in [100] with modifications due to the fact that most state bits of type $4k$ are no longer used except D(1)=0 and P(21)=32. We mark with green numbers bits 32, or those in W, and those XORed to the outputs $T1 - T9$ respectively, cf. Section 7.5.

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 [100] 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.

[72] SKS is also mentioned in other T-310 sources such as [55].

[73] These can be traced to Section 4.1 in Appendix 1 of [100] and they hold for all of KT1 keys, all of KT2 keys, and also for this key 16.

On page 48 in [100] 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 [100], which is expected to be a bijection on 27 bits, cf. Section F below.

F.4 A Sufficient Condition to Emulate SKS V/1

An interesting question is, what conditions should a long-term key satisfy to in order emulate a SKS V/1 cipher. A quick comparison of our later Fig. 6.41 page 148 to earlier Fig. 5.7 page 19 and of Fig. 7.11 page 26 to Fig. 6.39 page 145 and Fig. 6.40 page 146 suggest the following answer. The answer is that we need to prevent P from using any bits from the 4th branch, and force D to use exactly 8 out of these bits, 1 being s1, and without repetitions as Q/R is expected to be a bijection, and we need to make sure that first of these bits is $s1$ which is denoted as 0 in D.

We obtain the following set of "very minimalist" conditions KS0 which needless to say, are all satisfied by key 16:

$(P,D) \in KS0 \Leftrightarrow$ all of the following hold:

$$\begin{cases} \text{Let } \ W' = \{3,7,11,15,19,23,27,31,35\} \\ D(1) = 0 \\ \forall_{2 \geq i \geq 9} D(i) \in W' \quad \text{(one is inevitably missing)} \\ D \text{ is injective} \\ \text{at most 1 value } P(.) \text{ should be a multiple of 4} \end{cases}$$

As a bonus which we obtain "for free", key 16 satisfies also a well-known KT1-type property that $\forall_{1 \geq i \geq 9}\ D(i) \notin W$ where $W = \{5,9,21,25,29,33\}$ which is a simple consequence of W' property in KS0 above.

In addition we also list some stronger properties which key 16 also satisfies, but it is not clear if they are necessary for SKS V/1 to be secure, nor if these extra properties would be in any way mandatory or required for the designers of SKS V/1 cipher.

$(P,D) \in KS1 \Leftrightarrow$ all of the following hold:

$$\begin{cases} \text{all conditions of KS0 hold} \\ P \text{ is injective} \\ P(3) = 33, P(7) = 5, P(9) = 9, P(15) = 21, P(18) = 25, P(24) = 29 \\ \text{There exist } \ \{j_1, j_2, \ldots, j_7, j_8\} \text{ a permutation of } \{2,3,\ldots,9\} \ \text{ which} \\ \quad \text{defines } D(i) \text{ for every } i \in \{2,3,\ldots,9\} \ \text{ as follows:} \\ \quad D(j_1) = 3, D(j_2) = 4j_1 - 1, D(j_3) = 4j_2 - 1, \ldots, D(j_8) = 4j_7 - 1 \\ P(20) = 4j_8 - 1 \ \text{ (note: this value is not any of the } \ D(i)) \\ \text{Exactly one integer } m = 4k+3 \text{ is missing in } Im(P) \text{ and } P(21) = m+1 \text{ is present} \end{cases}$$

For example for LZS-16 we have $(j_1, j_2, \ldots, j_7, j_8) = (7, 5, 3, 6, 4, 8, 9, 2)$.

It follows from these conditions that the intersection of P and D must have 7 elements: all the eight non-zero elements of D except one which is 31, which is still present in a certain way in LZS-16: it replaced by 32 in P which means that bit 31 is used by P one clock later.

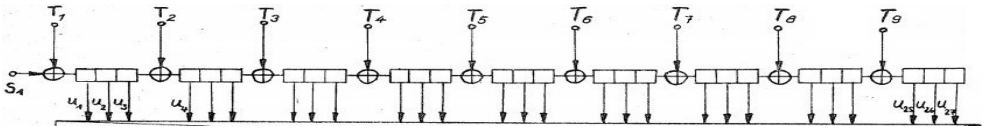


Fig. 6.43. A cascade of shift registers in an intermediate design which was neither SKS V/1 not T-310 cipher [61]. The key bit $s1$ enters the first register, then it propagates in a specific order. This order is precisely what is specified by a permutation of wires $\{2, 3, \ldots, 9\}$ which in different descriptions of SKS or T-310 emulating SKS, [100] will be denoted by R^{-1}, Q or j_{1-8}.

F.5 Security of SKS/1

It is possible to see that the security of SKS depends very strongly on the value of p: if p is small, the attacker will have access to keystream bits generated with just the first 104 rounds and the cipher should be broken by a variety of methods such as software algebraic attacks (cf. also Section 15) and also by slide attacks similar as in T-310 yet substantially different than in current paper, cf. for example Section 24 and Appendix G.

It is also possible that in the same way as in T-310, SKS V/1 will fall to very powerful ciphertext-only attacks if the long-term keys are faulty or not chosen carefully. For example in Section 20.3 we show one such attack in the case where the long-term key is such that ϕ is not bijective.

G An Advanced Birthday Paradox Sliding Key Recovery Attack on T-310 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 240 bits. The answer is that we need to combine several variants of the above attack and apply Thm. 25.2.1 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. 25.2.1.
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: $u_{120s} = u_0 = 0xC5A13E396$ 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[74] 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 240-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.

[74] For example, in Table 1, Section 9, page 25 in [40], 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 aperiodic which can be compared to other block ciphers where regular periodic structure is a source of numerous attacks, e.g. GOST [40]. 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 [29, 30, 37], 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 an aperiodic 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" [29, 30] which comes from LFSRs. These attacks were later improved/enhanced to tolerate a proportion of arbitrary non-linear components, cf. [30] and Fig. 8.44 below. We reproduce this picture here to show that it DOES apply to T-310.

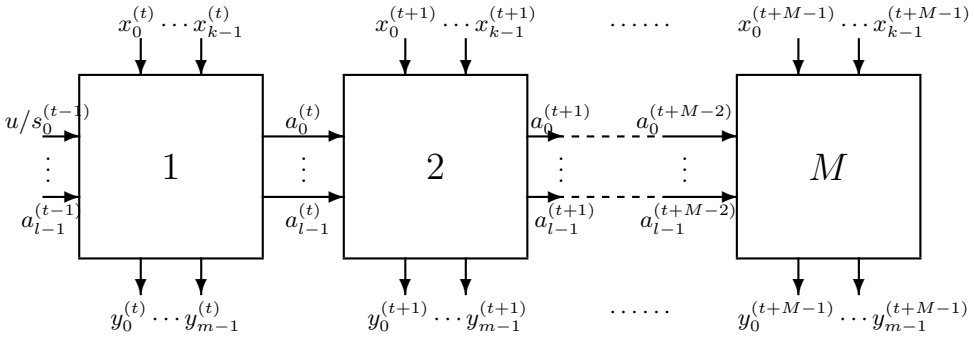


Fig. 8.44. M successive applications of a combiner with k inputs, m outputs and l bits of memory, a general setting in stream cipher cryptanalysis, cf. Fig. 2. in [30]

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. 8.44 and in [30], is now going to be a combination of the secret key and the 36-bit block state. Therefore the internal state will now be 276 bits: 240 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. 8.44 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. 8.44 contains 120iterations of ϕ and the key is on 240 bits and output identical to enter the next box, then we have $(k, m, l) = (120, \frac{120*10}{127*13}, 240 + 36)$.

In both cases the inputs come from one or several LFSRs, a linear component, exactly as in [30], which means that we can potentially apply the methods and ideas of [30]. This however will encounter very substantial difficulties: the main idea in [30] 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 [30] 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 [30] 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. 8.44 above and [30]. Overall the analysis of [30] can potentially be applied, at least in theory. The main point of [30], 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 [30]. 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 On Hardware and Software and Implementation Aspects of T-310

In this section we provide some software and tools which can help the reader to study the T-310 cipher.

I.1 Technical Information On Implementation of T-310 Encryption in T-310/50 and T-310/51 Cipher Machines

This section contains some important information about how T-310 encryption was implemented in practice in Eastern Germany. It is primarily based on informations provided by Jörg Drobick in [55, 56, 58] and on additional clarifications we have obtained from the author [in person and by email]. The actual machines can be seen in Harnekop NVA Museum in Prötzel, Germany, cf. `www.nva-harnekop.de`, some videos filmed at these premises can be found at [57].

1. The 240-bit key ZS is stored on punch cards.
2. On insertion followed by removal of the punchcard from the device, the key is stored in internal shift registers[75] which is similar to storing it in RAM and if the power supply is disconnected, the key would be lost.
3. There is a red 'panic' button on the operator console which allows to erase the key and produces an alarm.
4. Each T-310 encryption machine implements all the main internal state/keystream generation routines **twice**, cf. Section I.2 below.
5. Moreover, one single "complication unit" as depicted on Fig. 9.16 is implemented using 1.5 of a board of type GVS01 1103X the picture of which can be found at `http://scz.bplaced.net/t/t310lz.jpg` and additional explanations in [55].
6. Each board of type GVS01 1103X has a socket to insert a smaller board which is sometimes called 7905.XX which contains the long-term key (LZS).

I.2 Additional Notes and Remarks On Actual Hardware Implementation of T-310

The complication unit is implemented twice, probably in order to protect against fault injection attacks (a.k.a. DFA) [94, 51] and probably also to save operators some time dealing with faults and errors. The same is also true for SKS V/1 machines. If the internal keystream results are not the same, the encryption is stopped and nothing will be output. This can be viewed as protection against fault attacks which are in general quite powerful and could be combined with software algebraic attacks cf. [51] and Section I.10. Interestingly, the memory which stores the 240-bit key is NOT duplicated. If an error would occur in a circular shifting of the key, the faulty key would be kept in memory and it could be used to encrypt or decrypt further characters of the message (or further messages).

[75] We have twice 30 shift registers on 8 bits implemented using 74LS91 TTL ICs.

I.3 Coding of Short-Term Keys in T-310

In the section we give an example of a 240-bit short term key of T-310 and we explain how such keys are encoded. Detailed explanations were provided by Jörg Drobick.

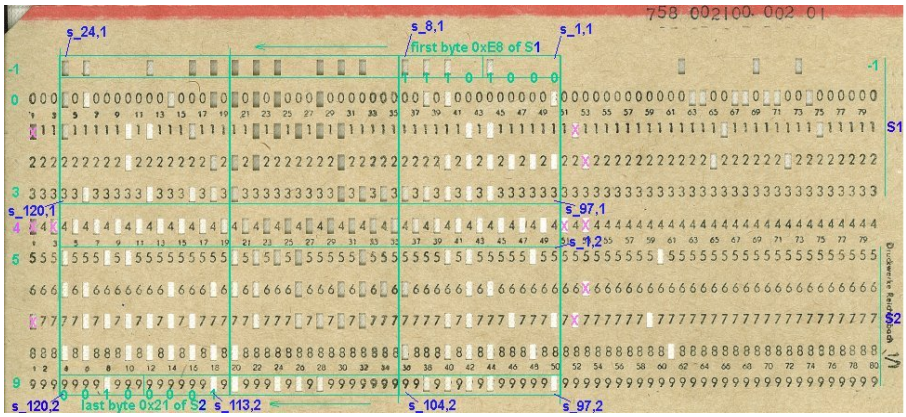


Fig. 9.45. ZS Key known as 758 002100 00X 01, an actual historical example of a key which can be found at [55] or `http://scz.bplaced.net/k/758-wochen.jpg`.

The key on punched car which is shown on this example should be read form right to left and from upper line -1 down to the lowest line 9, excluding the control line 4. This can be transcribed in hex as follows:

```
S1 = E8 EE CB 61 E8 C5 18 7C 9A 3F C4 11 F8 87 4B
S2 = 32 F2 B0 98 D7 C5 35 5E 7E BA 80 DF 79 B8 21
```

Here bytes are written from left to right, for example E8 is 1110 1000 in binary, corresponds to in order to bits $s_{8-5,1}, s_{4-1,1}$ with the notations of Section 13.2. Reading key bits in the opposite order $1, 2 \ldots 120$ for S1 and likewise for S2, we see that bits $s_{1-8,1}$ are in order 0001 0111 and at the end we have $s_{113-120,2}$ which are in order 1000 0100.

I.4 Some Test Vectors For T-310

We give here some test vectors for T-310. Notations are self-explanatory. Our first test vector details many internal values inside the cipher. The key is different than in previous section.

```
LZS 26 KT1=1
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,
S1[1..120] = 010100101001010011010101101000100111010010101100111101100111
111100001111011000010001000111101101011110001110001000101100
S2[1..120] = 110100110111100111011001000001010111000011010010101100010001
101011011011111001110100100011100101101001100110111100110001
IV[-60..0] = 000100000101111111110101001110100110011101000000000111111111
U0[1..36] = 011010011100011111001000010110100011
U1[1..36] = 001111000110101101101100101011011001
Keystream at alpha u_{i,alpha} i=1..127 =
110011101010001011111011101100111100111010010000100000100111
100000100111000110001100011011000001011101101111000010100010
1011010
keystream Ai generated [i=1..]
00000 1 11110 00 10101 1 11111 11 10101 0 00010 00 11001 1 00101 10
11001 0 10101 01 10101 0 01110 00 01110 1 00110 01 11100 1 00011 01
10011 1 00100 01 01100 0 00001 00
R_1/r_1 B_1 R_2/r_2 B_2 = 0/ 0 30 21/13 31
Plaintext chars ITA-2 12345 (A=24):
31 2 8 8 24 19 14 18 16 22
Plaintext chars:
LS CR LF LF A B C D E F
Ciphertext chars ITA-2:
1 11 25 24 24 11 12 6 26 17
Ciphertext chars:
T G W A A G I N J Z
```

I.5 Remark: KPA Attack for Free

We observe that the plaintext is pre-pended by the so called "MBF 2" or "Maschinenbefehlsfolge 2" which is LS CR LF LF. This is what happens in real life situations and these 4 characters will be encrypted and the corresponding plaintext is known to the attacker. Therefore in some sense, we always have a known plaintext attack with T-310 for the first 4 characters which corresponds to $127, \ldots 4 \cdot 13 \cdot 127$ iterations of the block cipher.

I.6 Padding, Formatting, Ini: Another Test Vector For T-310

This second test vector emphasises the padding, formatting and initialization questions: how exactly the machine needs to be synchronised at the receiving end and how the IV and the message are encoded and transmitted.

First we explain how plaintext is formatted and padded.

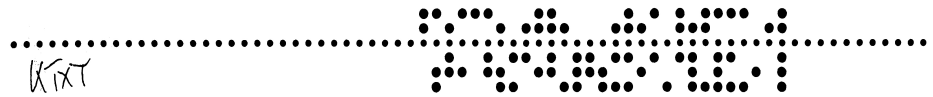


Fig. 9.46. Punched tape with the plaintext ABCD..YZ + final 3 characters.

1. In real-life encryption this plaintext would also be pre-pended with "MBF 2" as in Section I.4.
2. Then we have the plaintext which is ABCD..YZ here.
3. At the end if appended CR LF LS which marks the end of the plaintext.

Now we explain how encryption is performed and how data are padded.

Fig. 9.47. Punched tape with the initialization sequence and the ciphertext.

1. At the beginning of each transmission we have 1+3+4 characters which are always the same. The character LS which marks the beginning, then a sequence of 3 characters called "MBF 1" or "Maschinenbefehlsfolge 1" which is LS CR LF. This followed by letter 'b' repeated four times which is known as "BFF" or "Befehlsfolge".
2. Then we have 25 characters which conveys the initialization vector on 61 bits a.k.a. "SYF" or "Synchronfolge".
3. Finally we have 4 characters at 'k' repeated four times. This is called "BFG" or "Befehlsfolge".
4. The we have the proper ciphertext. The first 4 characters are the encryption of MBF 2. The last 3 character are the encryption of CR LF LS.

Finally we show the result of the decryption:

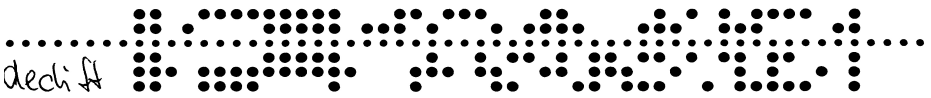


Fig. 9.48. Punched tape obtained from a decryption.

This tape contains several parts:

1. First we have LS.
2. Then MBF 1, i.e. LS CR LF.
3. Then 4 times b.
4. Then 4 times k.
5. Then MBF 2.
6. Then the plaintext ABCD..YZ
7. Finally the plaintext ending CR LF LS.

The key used in this example is the same as in Section I.3. The coding of IV will be specified in a future update of this paper.

I.7 More Test Vectors For T-310

Additional test vectors can be found in [57] and in [58] or more exactly at `http://scz.bplaced.net/ke-sks-t310.html#aufbau`.

I.8 A Reference Software 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.

```
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
};

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, allows to determines D uniquely for KT1 keys
int j[9]={-1,3,7,2,6,5,8,4,9};
//hard coded part of P,
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]
};
```

I.9 Software Simulators by Drobick

Software simulators which work under Windows are provided by Jörg Drobick. They operate in the same way as the original machines which is a bit tricky. This video shows how to use the simulator for T-310/51 SAGA:
https://www.youtube.com/watch?v=zh2pbsr3Kx4

I.10 Our Software Algebraic Attack CodeGen Tool

We have developed an open source software solution [3] for implementing software algebraic attacks on T-310 which we use in Section 15.3. a new final release is in preparation. 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 [23] for example a direct download link would be `http://www.nicolascourtois.com/software/codegen_latest.zip` or `http://www.nicolascourtois.com/software/ax64.exe` or `http://www.nicolascourtois.com/software/*.py`.

This software uses Python 2 (not Python 3). The basic command line reference is:

```
pip install Pexpect
pip install pypiwin32
python codegen.py Nr /fix115 /insX /xl /sat /T310set26
Nr = number of rounds
X=number of instances, 7 recommended
/T310set26 or /LZS26 = uses LZS-26
/fix115 could be replaced by /fix1/2 which will fix half of the 240 key bits
```

Some concrete examples are:

```
python codegen.py 100 /fix160 /ins7 /sat /LZS26
python codegen.py 120-121 /fix160 /ins7 /sat /LZS26
python codegen.py 100 /fix160..180 /ins7 /sat /LZS26
python codegen.py 100 /fix1/2 /ins7 /sat /LZS26
python codegen.py 100 /fix0.5 /ins7 /sat /LZS26
python codegen.py 100 /fix10,20,30 /ins7 /sat /LZS26
python codegen.py 100 /internalfix0.125 /ins7 /sat /LZS26
python codegen.py 40 /ins2 /sat /LZS31 /timeout2000
python codegen.py 40 /ins2 /sat /LZS31 /timeout20*Nr
python codegen.py 40 /ins2 /sat /LZS31 /seed1-2
```

I.11 Our KT1 Key Generation Weak/Strong Key Tool

We have developed another open source software solution [4] in order to generate and study KT1 keys in T-310 with particular focus on various classes of weak keys. Certain files necessary to work should be in the current directory and are the following:

```
keygen.py
```

The latest version can be obtained from [4] or [23], and a direct download link is
`http://www.nicolascourtois.com/software/keygen_latest.zip`.

The basic command line reference is:

```
python3 keygen.py --help
```

If both Python 2 and 3 are installed, we may need to specify the path of Python, for example

```
C:\Users\XYZTY\AppData\Local\Programs\Python\Python36\python keygen.py

usage: keygen.py [-h] [--method METHOD [METHOD ...]]
[--KT1symmetry1]
                 [--KT1symmetry2] [--KT1symmetry2pairs
KT1_SYMMETRY2_PAIRS]
                 [--KT1symmetry2f KT1_SYMMETRY2_WRONGPAIRSFILE] [--verify]
                 [--diffprop DIFFPROP] [--linprop
LINPROP]
                 [--clinprop CUSTOM_LINPROP]
[--save_linprop]
                 [--additional_masks ADDITIONAL_MASKS] [--count_j]
                 [--filter_j FILTER_J] [--count_d2p27] [--learn_cond]
                 [--forceD FILTERD] [--forceP FILTERP] [--forceDP FILTERDP]
                 [--force FILTER_GEN] [--key KT1_CL] [--print]
                 [--save_j SAVE_JS] [--debug]


KT1 stream generator and property checker

optional arguments:
  -h, --help            show this help message and exit
  --method METHOD [METHOD ...], -m METHOD [METHOD ...], /method METHOD
[METHOD ...]
                        KT1 key generation method used (default:
"german",
                        alt:
"normal")
  --KT1symmetry1, /KT1symmetry1
```

```
                        Use KT1 symmetry property to generate more
keys
  --KT1symmetry2, /KT1symmetry2
                        Use KT1 symmetry property to generate more
keys
  --KT1symmetry2pairs KT1_SYMMETRY2_PAIRS, /KT1symmetry2pairs
KT1_SYMMETRY2_PAIRS
                        Define pairs for KT1 symmetry property 2
  --KT1symmetry2f KT1_SYMMETRY2_WRONGPAIRSFILE,
                  /KT1symmetry2f KT1_SYMMETRY2_WRONGPAIRSFILE
                        File containing wrong pairs for KT1 symmetry
property 2
  --verify, -t, /verify
                        Verify validity of each generated KT1 key
  --diffprop DIFFPROP, /diffprop DIFFPROP
                        Filter keys according to the given Related Key
                        Differential Property
  --linprop LINPROP, /linprop LINPROP
                        Filter keys according to the given Linear Property
  --clinprop CUSTOM_LINPROP, /clinprop CUSTOM_LINPROP
                        Internal use only
  --save_linprop, /save_linprop
                        Create the file level2_lc.txt containing a log of
the
                        observed keys and the Z sets for the specified
linear
                        property
  --additional_masks ADDITIONAL_MASKS, --auxmasks ADDITIONAL_MASKS,
  /additional_masks ADDITIONAL_MASKS
                        additional linprop masks for Z detection only
  --count_j, /count_j   Count seen/missing Js: compute a J that satisfies
the
                        KT1 conditions for each generated key, then compare
to
                        the list of all possible Js and of a class of "weak"
                        Js
  --filter_j FILTER_J, /filter_j FILTER_J
                        Filter by given js
  --count_d2p27, /count_d2p27
                        Count seen/missing combination of D[2] and P[27]
  --learn_cond, /learn_cond
                        Intersect all generated keys, discovering if some
                        P[i],D[i] are constant (or if there exists a
P[i]=D[j]
  --forceD FILTERD, -D FILTERD, /forceD FILTERD
```

```
                        Restrict D values: D[i] = j -> i-j
  --forceP FILTERP, -P FILTERP, /forceP FILTERP
                        Restrict P values: P[i] = j -> i-j
  --forceDP FILTERDP, -DP FILTERDP, /forceDP FILTERDP
                        Restrict P and D values: D[i]=P[j] -> i-j
  --force FILTER_GEN, -f FILTER_GEN, /force FILTER_GEN
                        Restrict P and D values: D[i]=P[j], D[h]=k, P[l]=m
  --key KT1_CL, -k KT1_CL, /key KT1_CL
                        Specify a single key to run the software on
  --print, -p, /print   Print all genrated keys (to stderr)
  --save_j SAVE_JS, /save_j SAVE_JS
                        Save generated js to specified file
  --debug, -d, /debug   Enable debug mode
```

Key Generation Methods

We have three different methods for generating weak/strong KT1 keys implemented.

Some of those methods happen to generate keys which violate KT1 conditions: we call these keys "NKT1" keys, and their number appears in the first of the status columns when we run the software.

```
keygen.py -m=german
keygen.py -m=normal
keygen.py -m=js
```

What is called "german" method is based on [60] translated from German by UCL student Simon Boehm. Surprisingly, this original method also produces NKT1 keys.

The "normal" method generates keys based on a random J. Several optimizations are present to make the key generation faster, but from the simulations we ran it is not clear whether all of the key space is covered.

The js method allows to specify a list of Js allowed. For example:

```
python keygen.py  --method js 23456789,23456798 --count_j
```

The key generation process can be sped up using the KT1 symmetry properties, implemented by the `--KT1symmetry1` and `--KT1symmetry2` command line switches. For example, the following command line:

```
python keygen.py -m=german --KT1symmetry1 --KT1symmetry2
```

generates keys using the German method, then expands the generated keys using the symmetry properties 1 and 2, showing stats on how many keys are non-KT1 (NKT1) after each transformation.

Some tweaks on the KT1 symmetry property 2 are possible:

```
python keygen.py --KT1symmetry2 --KT1symmetry2pairs "1,[5-8];"
```

modifies the pair set used the property, for testing purposes.

Key Filtering

The tool allows to filter the generated keys according to several properties. Different filters can be stacked, but the order of the filters is hardcoded.

```
python keygen.py --count_j
```

Show stats on the j produced from the generated keys. Missing js could mean that the generation method does not cover the whole keyspace.

```
python keygen.py  --method german --count_j --filter_j 23456789,23456798
     NKT1/total   Keygen Speed   FiltJ/tested        Hard Js        Weak Js
 122/  17742        1510k/s        2^-1.48        0/35771     3436/ 4549
```

A separate feature allows filtering by j, for example: In the above 23456789,23456798 could be replaced by a filename and a longer list could be inside a file.

Examples Relevant to Linear and Differential Attacks Some filters are aimed at testing particular linear or differential properties.

Here is a command line to generate KT1 keys with specific linear approximations:

```
python3
keygen.py --linprop "4x [11,15,27,31]" --print
```

Here is a command line to generate KT1 keys with specific constraints on P and D:

```
python3
keygen.py -m=german --force "D[6]=32 P[8]=1" --print
```

More specific examples of command line options related to Linear Cryptanalysis can be found in Section 21.23.

The tool can also generate keys respecting one specified related-key differential property:

```
keygen.py  --diffprop "s1>1>2>3>4,s1" --print
```

Check a related-key differential property on a specific key:

```
keygen.py --key
"P=16,6,33,11,20,24,5,13,9,7,31,19,36,12,21,30,34,25,17,32,23,28,4,29,26,8,3
D=0,4,16,28,12,20,36,24,8 []"  --diffprop "s1>1>2>3>4,s1"
```

If the property is satisfied, the program will output:

```
NKT1/total   Keygen Speed   DiffPr/total
0/      1           0k/s      1/      1
```

I.12 Our Illustration Tool

Matteo Scarlata have developed another open source software tool to visualize connections inside long-term keys. The main files are

```
create_key_graph.py
keygen.py
```

And the tool is included as part of the latest version of [4] and a direct download link is
`http://www.nicolascourtois.com/software/keygen_latest.zip`.

The basic command line reference is:

```
python3
python create_key_graph.py
"P=15,24,33,27,19,12,5,22,9,31,3,7,8,34,21,36,32,25,18,28,35,20,4,29,16,14,2
D=0,36,24,4,32,16,8,12,20 [1]"
--color "u2_21=red u1_12=blue u1_32=green"
-k 741 -r 1
```

This will generate a file "phi_diagram_741.xml", which can be opened and exported/saved as JPG PNG or other image format, or edited further at `http://draw.io` . We show an example result obtained with this tool.

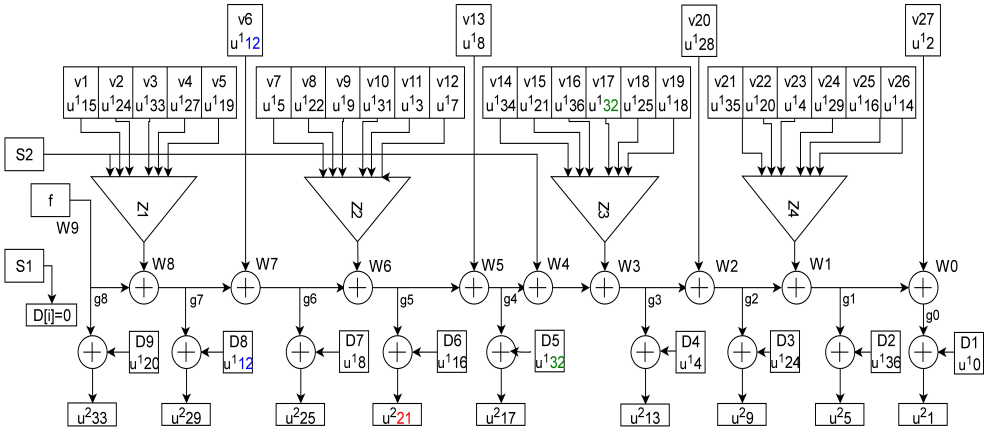


Fig. 9.49. One round of T-310 for key 741.

I.13 Short Documentation For Our DC Tool

Our Differential Cryptanalysis (DC) tool [95] was written by our student Matteo Scarlata and was used in Section 12.5. 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.
```

I.14 On Hardware Implementation of the Boolean Function *Z*

We have applied a free software logic optimizer in order to obtain a gate efficient implementation of $Z()$. The software tool we used is ABC by Berkeley Logic Synthesis and Verification Group. Here is our command line:

```
abc.exe
read_dsd !(a+e+f+(a*d)+(b*c)+(b*e)+(d*e)+(e*f)+(a*c*d)+(a*c*f)+(a*d*e)+(b*c*f)+
(b*d*f)+(c*e*f)+(a*b*c*d)+(a*b*c*e)+(a*b*e*f)+(b*c*d*f)+(a*b*c*d*e)+(a*c*d*e*f))
resyn
resyn2 cleanup fx dsd rewrite
resyn2 cleanup fx dsd
resyn2 cleanup fx dsd rewrite
resyn2 cleanup fx dsd rewrite
write_verilog o.verilog
print_stats
dsd           : i/o =    6/    1  lat =    0  and =     51  lev =  9
```

We see that this circuit can be implemented with 9 levels and using 51[76] AND gates [with some inputs negated]. Here is the result we obtained:

```
// Benchmark "dsd" written by ABC on Tue Jan 30 18:29:37 2018
module dsd (   a, b, c, d, e, f,    F  );
  input  a, b, c, d, e, f;   output F;
  wire n7, n8, n9, n10, n11, n12, n13, n14, n15, n16, n17, n18, n19, n20,
    n21, n22, n23, n24, n25, n26, n27, n28, n29, n30, n31, n32, n33, n34,
    n35, n36, n37, n38, n39, n40, n41, n42, n43, n44, n45, n46, n47, n48,
    n49, n50, n51, n52, n53, n54, n55, n56;
  assign n7 = b & ~c;  assign n8 = a & e;
  assign n9 = ~d & ~n8;  assign n10 = d & n8;
  assign n11 = n7 & ~n10;  assign n12 = ~n9 & n11;
  assign n13 = a & d;  assign n14 = ~b & e;
  assign n15 = b & ~e;  assign n16 = ~n14 & ~n15;
  assign n17 = c & ~n13;  assign n18 = ~n16 & n17;
  assign n19 = ~n12 & ~n18;  assign n20 = f & ~n19;
  assign n21 = b & c;  assign n22 = ~d & ~e;
  assign n23 = d & e;  assign n24 = f & ~n23;
  assign n25 = a & n21;  assign n26 = ~n22 & n25;
  assign n27 = ~n24 & n26;  assign n28 = ~n20 & ~n27;
  assign n29 = a & ~c;  assign n30 = ~d & n29;
  assign n31 = f & ~n21;  assign n32 = ~n30 & n31;
  assign n33 = ~a & ~b;  assign n34 = ~c & d;
  assign n35 = ~f & ~n33;  assign n36 = ~n34 & n35;
  assign n37 = ~e & ~n36;  assign n38 = ~n32 & n37;
  assign n39 = ~a & ~d;  assign n40 = ~n13 & ~n39;
  assign n41 = n7 & n40;  assign n42 = c & ~d;
  assign n43 = n40 & ~n42;  assign n44 = ~n7 & ~n43;
  assign n45 = e & ~n41;  assign n46 = ~n44 & n45;
  assign n47 = ~c & ~d;  assign n48 = ~a & ~n47;
  assign n49 = b & ~n48;  assign n50 = ~n8 & ~n49;
  assign n51 = ~f & ~n29;  assign n52 = ~n50 & n51;
  assign n53 = ~n38 & ~n52;  assign n54 = ~n46 & n53;
  assign n55 = n28 & n54;  assign n56 = ~n28 & ~n54;
  assign F = ~n55 & ~n56; endmodule
```

I.15 The Cost of Hardware Implementation of T-310

Now we can estimate very roughly the cost of hardware implementation of the whole T-310 cipher. We evaluate the cost of $Z()$ to be roughly 100 GE. One round of encryption requires 4 applications of Z and 9 XORs, cf. Section 9. We assume that one XOR costs 3.2 GE. Overall with some overheads we need maybe about 430 GE for 1 round of T-310. In order to encrypt one character we need $127 \cdot 13 = 1651$ rounds. Therefore we need about 700 KGE to encrypt

[76] A better implementation with 5 or 6 AND gates only (and only if we neglect the cost of negations) is expected to exist cf. Section I.16 and [9].

one 5-bit character, so the encryption cost per bit is about 140,000 GE. This is extremely expensive, several orders of magnitude more expensive than any modern block cipher we have ever heard of. For example one optimized variant of GOST requires only 650 GE per 64 encrypted bits [92].

I.16 On Multiplicative Complexity of *Z*

It remains an open problem to compute the multiplicative complexity of Z. Potentially this is already done in [9]. We have inspected the github files of [9] and found that functions with $MC = 5$ contain 91 Boolean functions which have the same Walsh spectrum stats and same algebraic degree 5 as our Z function, while the file for $MC = 6$ had 0 such cases. Furthermore we looked at auto-correlation spectra of these functions and found 6 functions in the file which are candidates for being affine equivalents of Z. This sort of "signature" does NOT identify functions uniquely, cf. page 16 of [9]. On the basis of Walsh spectrum, we should consider that Z is probably equivalent to one of the following functions from [9]:

```
f130158, f14045, f214, f145636, f6449, f31545
```

However the auto-correlation spectrum seems to indicate that Z is none of these candidates, therefore probably we have that $MC = 7$. To be confirmed in the next update of this paper.

I.17 An Alternative Boolean Function

A recent paper shows that Boolean functions with 6 variables and MC ≤ 6 are extremely rare, 200 times less than MC=5 which is the most common case, see Table 4 in [9]. The paper gives an example of a Boolean function with MC=6, here we recall the second example f_2 from page 15 of [9]:

$$f2 = e_4e_5+e_3e_4e_5+e_2e_5+e_2e_4+e_2e_4e_6+e_1e_5e_6+e_1e_4+e_1e_3+e_1e_2e_4e_5+e_1e_2e_3e_6$$

A possible implementation would be as follows:

```
a = e4e5
b = e1e6
c = e2e4
f2 = a + e3a + e2e5 + c + ce6 + be5 + e1e4 + e1e3 + ae1e2 + be2e3
```

This implementation is however sub-standard and uses more than 6 multiplications. An implementation with only 6 multiplications can be found at: https://github.com/usnistgov/Circuits/blob/master/slp/n6_slp_mc6.txt cf. f68424 in line 6271.

```
a0 = (x6+x3+x2+x1) * (x6+x3+x1)
a1 = (1+x6+x5+x3+x2+x1) * (x6+x5+x3+x2)
a2 = (x4+x3+x2+x1) * (x5+x4+x2)
a3 = (1+x5+x4+x2+x1) * (a0+a1+1+x6+x5+x3+x1)
a4 = (a2+1+x6+x5+x4) * (a0+a3+x6+x3+x2+x1)
a5 = (a0+a1+a2) * (a1+a3)
Z  = a4+a5+a1+a2+a3+x4+x3+x2
```

Remark: This Boolean function is implemented in our codegen software and the command line switch is /Z2.

J Some Detailed Proofs For Linear Cryptanalysis Parts

J.1 A Detailed General Result for 8 Rounds and Key 788

The key 788 (and also 782,789,784,787) is an example of key where we have a property for 8 rounds as follows:

[9,13]→ [10,14]→ [11,15]→ [12,16]→
[25,29,33]f→ [26,30,34]→ [27,31,35]→ [28,32,36]→ [9,13]

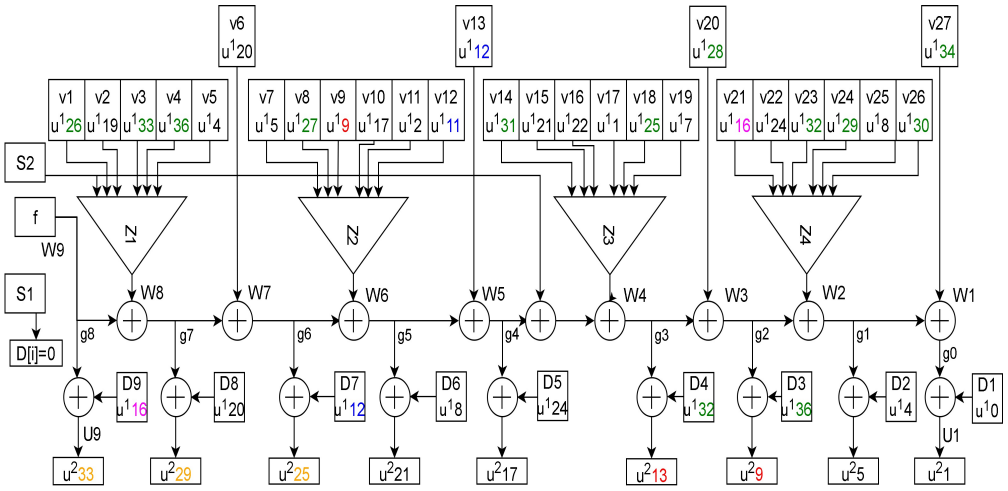


Fig. 10.50. One round of T-310 for key 788.

A General Result

More generally our student Marios Georgiou has proven the following result [extracted from his master thesis to be completed in 2018]:

Theorem J.1.1 (A class of 8R properties). For each long term KT1 key such that $\{D(7), D(9)\} \in \{12, 16\}$, $\{D(3)/D(4)\} = 32$, with the remaining of $\{D(3)/D(4)\} \in \{28, 36\}$ and finally the remaining of $\{28, 36\}$ becoming $P(20)$, and for any short term key on 240 bits, and for any initial state on 36 bits, we have the linear approximation $[9, 13] \rightarrow [9, 13]$ which is true with probability exactly 1.0 for 8 rounds.

Proof: We will show that the following holds:

rounds	input → output	bias
3	[9,13] → [12,16]	$2^{-1.0}$
1	[12,16] → [25,29,33]	$2^{-1.0}$
3	[25,29,33] → [28,32,36]	$2^{-1.0}$
1	[28,32,36] → [9,13]	$2^{-1.0}$

Let $X^{(i)}$ denote values inside round i.

First, we observe that [9] → [10] → [11] → [12] and [13] → [14] → [15] → [16]

for 3 rounds. So $u_9^{(1)} = u_{12}^{(3)}$ and $u_{13}^{(1)} = u_{16}^{(3)}$.

We recall a subset of equations from Appendix C.10.

$$U_3 \oplus u_{D(3)} = U_4 \oplus u_{D(4)} \oplus u_{P(20)} \tag{3}$$
$$U_7 \oplus u_{D(7)} = U_8 \oplus u_{D(8)} \oplus u_{P(6)} \tag{7}$$
$$U_9 \oplus u_{D(9)} = f \tag{9}$$

Now given that for all KT1 keys we have $P(6) = D(8)$, cf. Appendix B. Therefore equation (7) becomes

$$u_{21}^{(4)} \oplus u_{29}^{(4)} \oplus u_{25}^{(4)} = u_{D(7)}^{(3)}$$

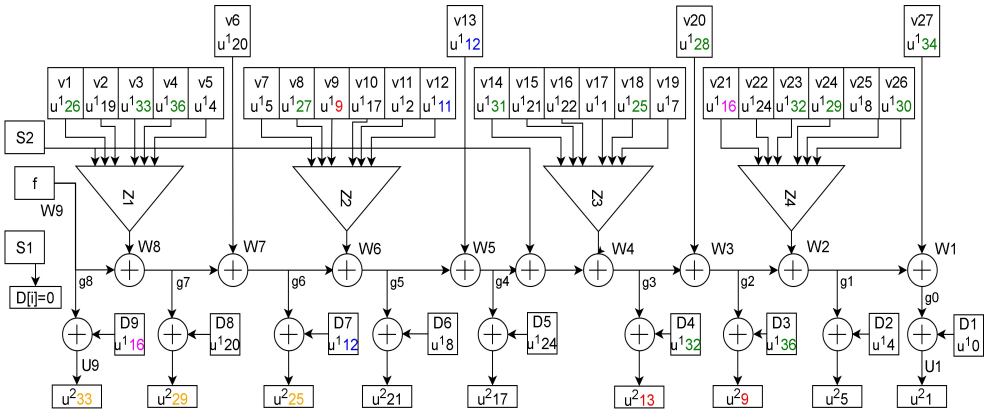


Fig. 10.51. Illustration repeated for better readability of the proof.

Then we have two cases:
Case A: If $D(7) = 12$ and $D(9) = 16$, then we have $u_{29}^{(4)} \oplus u_{25}^{(4)} = u_{12}^{(3)}$ and using (9) we have $u_{33}^{(4)} = f \oplus u_{16}^{(3)}$ which implies that $[12] \rightarrow [25, 29]$ and $[16] \rightarrow [33]$.
Case B: If $D(7) = 16$ and $D(9) = 12$, then we have $u_{29}^{(4)} \oplus u_{25}^{(4)} = u_{16}^{(3)}$ and using (9) we have $u_{33}^{(4)} = f \oplus u_{12}^{(3)}$ which implies that $[16] \rightarrow [25, 29]$ and $[12] \rightarrow [33]$.
Overall in both cases we have $[12, 16] \rightarrow [25, 29, 33]$.
Then we can easily observe that $[25] \rightarrow [28]$, $[29] \rightarrow [32]$ and $[33] \rightarrow [36]$ for 3 rounds.
From equation (3), the following holds

$$u_9^{(8)} \oplus u_{13}^{(8)} = u_{D(3)}^{(7)} \oplus u_{D(4)}^{(7)} \oplus u_{P(20)}^{(7)} \tag{10}$$

If we first have $\{D(3)/D(4)\} = 32$, with the remaining of $\{D(3)/D(4)\} \in \{28, 36\}$ and finally the remaining of $\{28, 36\} \in P(20)$, then clearly $\{D(3), D(4), P(20)\} \in \{28, 32, 36\}$. Hence, the following holds

$$u_9^{(8)} \oplus u_{13}^{(8)} = u_{28}^{(7)} \oplus u_{32}^{(7)} \oplus u_{36}^{(7)} \tag{11}$$

Therefore, we have successfully shown that $[28, 32, 36] \rightarrow [9, 13]$.
However, we cannot have $P(20) = 32$ and $\{D(3), D(4)\} \in \{28, 36\}$ because the following condition of KT1 keys of Appendix B. will no longer hold:
"There exist $\{j_1, j_2, \ldots, j_7, j_8\}$ a permutation of $\{2, 3, \ldots, 9\}$ which defines $D(i)$ for every $i \in \{2, 3, \ldots, 9\}$ as follows:
$D(j_1) = 4, D(j_2) = 4j_1, D(j_3) = 4j_2, \ldots, D(j_8) = 4j_7$"

J.2 Another Detailed Result for 8 Rounds and Key 868

Theorem J.2.1 (2nd class of 8R properties). For each long term KT1 key such that $D(2) = 36$, $D(9) = 4$, $\{D(5), D(6), D(7)\} \subset \{8, 20, 24\}$, $P(27) = 6$ and for any short term key on 240 bits, and for any initial state on 36 bits, we have the linear approximations $[1, 3, 5, 17, 21] \rightarrow [1, 3, 5, 17, 21]$ and $[3, 7, 19, 23, 33] \rightarrow [3, 7, 19, 23, 33]$ which are true with probability exactly 1.0 for 8 rounds.

Proof: We will show that the following holds:

rounds	input $\rightarrow$ output	bias
1	$[1, 3, 5, 17, 21] \rightarrow [2, 4, 6, 18, 22, f^{(2)}]$	$2^{-1.0}$
2	$[2, 4, 6, 18, 22, f^{(2)}] \rightarrow [4, 8, 20, 24, 34, f^{(4)}]$	$2^{-1.0}$
2	$[4, 8, 20, 24, 34, f^{(4)}] \rightarrow [18, 22, 34, 36, s_1^{(6)}]$	$2^{-1.0}$
2	$[18, 22, 34, 36, s_1^{(6)}] \rightarrow [2, 6, 8, 20, 24, 36, s_1^{(8)}]$	$2^{-1.0}$
1	$[2, 6, 8, 20, 24, 36, s_1^{(8)}] \rightarrow [1, 3, 5, 17, 21]$	$2^{-1.0}$

rounds	input $\rightarrow$ output	bias
1	$[3, 7, 19, 23, 33] \rightarrow [4, 8, 20, 24, 34, f^{(2)}]$	$2^{-1.0}$
2	$[4, 8, 20, 24, 34, f^{(2)}] \rightarrow [18, 22, 34, 36, s_1^{(4)}]$	$2^{-1.0}$
2	$[18, 22, 34, 36, s_1^{(4)}] \rightarrow [2, 6, 8, 20, 24, 36, s_1^{(6)}]$	$2^{-1.0}$
2	$[2, 6, 8, 20, 24, 36, s_1^{(6)}] \rightarrow [2, 4, 6, 18, 22, f^{(8)}]$	$2^{-1.0}$
1	$[2, 4, 6, 18, 22, f^{(8)}] \rightarrow [3, 7, 19, 23, 33]$	$2^{-1.0}$

In order to explain the results in the two above tables we will use the following facts:
Fact A: It is clear that $u_{4k+j}^{(i)} \rightarrow u_{4k+j+1}^{(i+1)}$ where $k \in \{0, \ldots, 8\}$ and $j \in \{1, 2, 3\}$, for some rounds i and $i + 1$.
Fact B: According to the theorem, $D(2) = 36$, $P(27) = 6$ and using equation (1) we have

$$u_1^{(i+1)} \oplus s_1^{(i)} = u_5^{(i+1)} \oplus u_{36}^{(i)} \oplus u_6^{(i)}$$

So we have $[6, 36, s_1^{(i)}] \rightarrow [1, 5]$.
Fact C: However we know that $u_6^{(i)} = u_7^{(i+1)}$ so the previous equation can also become

$$u_1^{(i+1)} \oplus u_5^{(i+1)} \oplus u_7^{(i+1)} = u_{36}^{(i)} \oplus s_1^{(i)}$$

Hence, $[36, s_1^{(i)}] \rightarrow [1, 5, 7]$ also holds.
Fact D: From the theorem, we also have $\{D(5), D(6), D(7)\} \subset \{8, 20, 24\}$. From

the description of KT1 keys we have that $P(13) = D(7)$ and ,thus, equation (5) becomes

$$u_{17}^{(i+1)} \oplus u_{21}^{(i+1)} = u_{8}^{(i)} \oplus u_{20}^{(i)} \oplus u_{24}^{(i)}$$

Hence, $[8, 20, 24] \rightarrow [17, 21]$.
Fact E: The last condition of the theorem is $D(9) = 4$ and equation (9) becomes

$$u_{33}^{(i+1)} \oplus u_{4}^{(i)} = f^{(i)}$$

Hence, $[4, f^{(i)}] \rightarrow [33]$.
Thus, combining some of the above facts in each round, we can prove that $[1, 3, 5, 17, 21] \rightarrow [1, 3, 5, 17, 21]$ and $[3, 7, 19, 23, 33] \rightarrow [3, 7, 19, 23, 33]$ for 8 rounds.

J.3 Another Detailed Result for 6 Rounds and Key 706

In a similar way we study the exact conditions which trigger the linear property of key 706 for 6 rounds (also found by Marios Georgiou):

Theorem J.3.1 (A class of 6R properties). For each long term KT1 key such that $D(7) = 16$, $\{D(3)/D(4), P(20)\} \subset \{4, 8, 36\}$, $P(27) = 10$ and finally $\{D(2), D(9)\} \subset \{28, 32\}$ and for any short term key on 240 bits, and for any initial state on 36 bits, we have the linear approximation $[1, 5, 15, 33, s_1^{(6)}, f^{(6)}] \rightarrow [1, 5, 15, 33]$ which is true with probability exactly 1.0 for 6 rounds.

Proof: We will show that the following holds:

rounds	input → output	probability
2	[1,5,15,33] → [3,7,25,29,35]	1.0
2	[3,7,25,29,35] → [9,13,27,31]	1.0
2	[9,13,27,31] → [1,5,15,33]	1.0

Let $X^{(i)}$ denote values inside round i. We recall a subset of equations from Appendix C.10.

$$U_1 \oplus s_1 = U_2 \oplus u_{D(2)} \oplus u_{P(27)} \quad (1)$$

$$U_3 \oplus u_{D(3)} = U_4 \oplus u_{D(4)} \oplus u_{P(20)} \quad (3)$$

$$U_7 \oplus u_{D(7)} = U_8 \oplus u_{D(8)} \oplus u_{P(6)} \quad (7)$$

$$U_9 \oplus u_{D(9)} = f \quad (9)$$

First of all, we observe that $[1] \rightarrow [3]$, $[5] \rightarrow [7]$ and $[33] \rightarrow [35]$ for 2 rounds. We also see that $[15] \rightarrow [16]$ for 1 round. So $u_1^{(1)} = u_3^{(3)}$, $u_5^{(1)} = u_7^{(3)}$, $u_{33}^{(1)} = u_{35}^{(3)}$ and $u_{15}^{(1)} = u_{16}^{(2)}$.

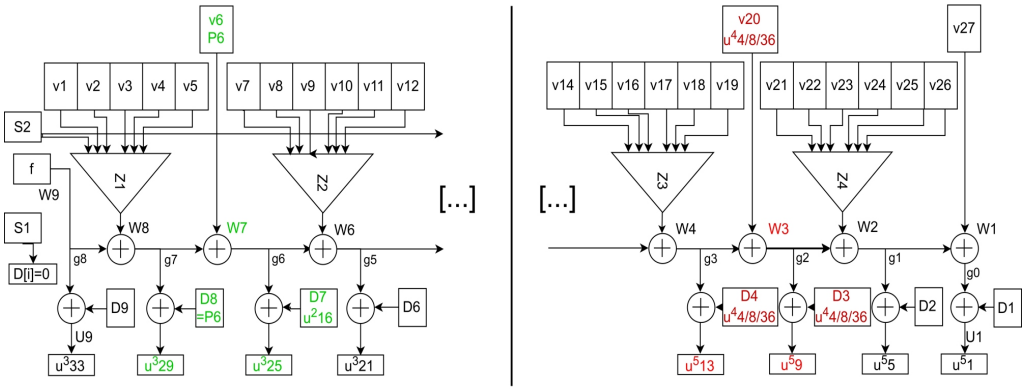


Fig. 10.52. Explanations for our proof for key 706.

From the KT1 properties in Appendix B we know that for all KT1 keys $P(6) = D(8)$. We also assumed $D(7) = 16$. Hence, equation (7) becomes

$$u_{25}^{(3)} \oplus u_{29}^{(3)} = u_{16}^{(2)}$$

Thus, we have $[16] \rightarrow [25, 29]$ for 1 round and, combining all the linear properties discussed so far, $[1, 5, 15, 33] \rightarrow [3, 7, 25, 29, 35]$ for 2 rounds.

Then we observe that $u_{25}^{(3)} = u_{27}^{(5)}$, $u_{29}^{(3)} = u_{31}^{(5)}$, $u_3^{(3)} = u_4^{(4)}$, $u_7^{(3)} = u_8^{(4)}$ and $u_{35}^{(3)} = u_{36}^{(4)}$. We assumed $\{D(3)/D(4), P(20)\} \subset \{4, 8, 36\}$. Therefore, equation (3) becomes

$$u_4^{(4)} \oplus u_8^{(4)} \oplus u_{36}^{(4)} = u_9^{(5)} \oplus u_{13}^{(5)}$$

Thus, we have shown that $[3, 7, 25, 29, 35] \rightarrow [9, 13, 27, 31]$ for 2 rounds.

We recall that our goal is to show the following sequence of linear equalities:
[1,5,15,33]s1f→[2,6,16,34]→[3,7,25,29,35]→
[4,8,26,30,36]→ [9,13,27,31]→[10,14,28,32]→[1,5,15,33]

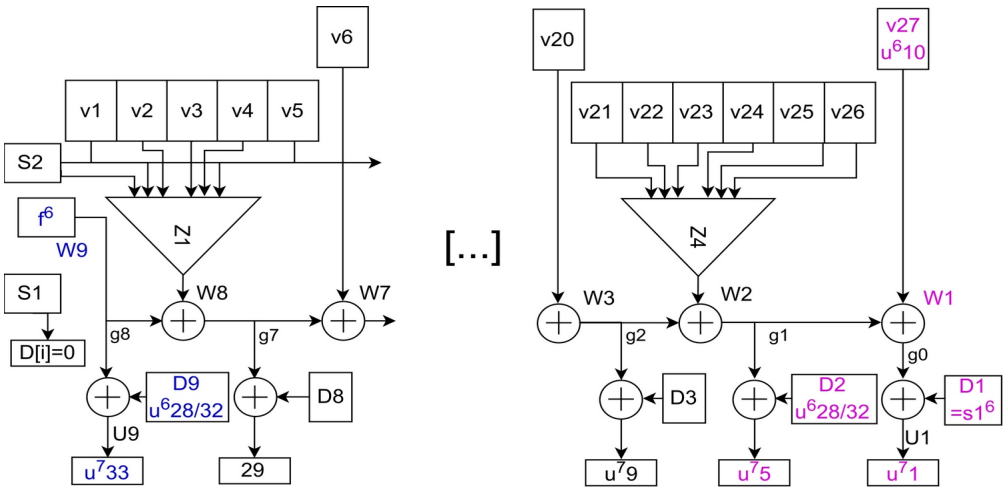


Fig. 10.53. Further illustration for our proof.

It is clear that $u_{13}^{(5)} = u_{15}^{(7)}$, $u_{9}^{(5)} = u_{10}^{(6)}$, $u_{27}^{(5)} = u_{28}^{(6)}$, and $u_{31}^{(5)} = u_{32}^{(6)}$. The remaining conditions from the theorem J.3.1 hypothesis are $P(27) = 10$ and $\{D(2), D(9)\} \subset \{28, 32\}$. Hence, equation (1) becomes

$$u_{10}^{(6)} \oplus s_1^{(6)} \oplus u_{D(2)}^{(6)} = u_1^{(7)} \oplus u_5^{(7)}$$

and equation (9) becomes

$$u_{D(9)}^{(6)} \oplus u_{33}^{(7)} = f^{(6)}$$

Finally

$$u_{10}^{(6)} \oplus u_{D(2)}^{(6)} \oplus u_{D(9)}^{(6)} \oplus s_1^{(6)} \oplus f^{(6)} = u_1^{(7)} \oplus u_5^{(7)} \oplus u_{33}^{(7)}$$

It follows that if $\{D(2), D(9)\} \subset \{28, 32\}$ we have $[10, 28, 32, s_1^{(6)}, f^{(6)}] \rightarrow [1, 5, 33]$ for 1 round. Finally, we have also shown that $[9, 13, 27, 31] \rightarrow [1, 5, 15, 33]$ for 2 rounds. This ends the proof that if the conditions of the theorem are satisfied, we have $[1, 5, 15, 33, s_1^{(6)}, f^{(6)}] \rightarrow [1, 5, 15, 33]$ for 6 rounds.

K On Cryptanalaysis of Modified Versions of T-310

K.1 On Parity Bits

Here is how the original documenation [100] specifies the parity bits on page 117:

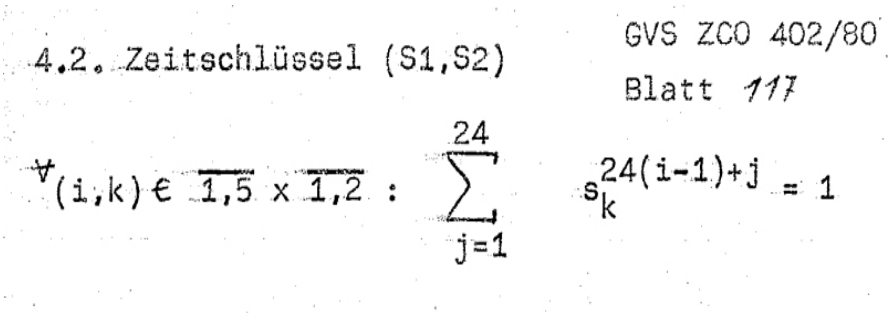


Fig. 11.54. The parity equations specified (but not used) in T-310 ciphers.

We expect that this will interact with LC-weak keys and slide attacks in many ways.

K.2 On Order of Bits Used For Encryption

We can make the cipher very substantially weaker in this way, this question is studied at the very end of Section 17.4.