Serdica J. Computing **7** (2013), No 3, 257–270

Serdica
Journal of Computing

Bulgarian Academy of Sciences
Institute of Mathematics and Informatics

PLAINTEXT RECOVERY IN DES-LIKE CRYPTOSYSTEMS BASED ON S-BOXES WITH EMBEDDED PARITY CHECK

Vesela Angelova, Yuri Borissov

Abstract. We describe an approach for recovering the plaintext in block ciphers having a design structure similar to the Data Encryption Standard but with improperly constructed S-boxes. The experiments with a back-tracking search algorithm performing this kind of attack against modified DES/Triple-DES in ECB mode show that the unknown plaintext can be recovered with a small amount of uncertainty and this algorithm is highly efficient both in time and memory costs for plaintext sources with relatively low entropy. Our investigations demonstrate once again that modifications resulting to S-boxes which still satisfy some design criteria may lead to very weak ciphers.

1. Introduction. General block ciphers combine simple operations to construct a complex encryption transformation. This tradition has its roots in Shannon's 1949 paper [1] connecting cryptography with information theory. Shannon suggested building a strong cipher system out of simple, individually weak components that substantiate the so-called "confusion" and "diffusion" of data,

ACM Computing Classification System (1998): E.3, I.2.7, I.2.8.

Key words: DES-like cryptosystem, S-box, affine subspace, plaintext recovery, backtracking search, language model.

*This work was presented in part at the 1st International Conference *Bulgarian Cryptography Days* 2012, Sofia, Bulgaria, 20–21 September 2012.

and applying these components iteratively in a number of rounds. Later (in the 1970s) this approach culminated in two main design methodologies: Feistel ciphers and substitution-permutation networks developed by researchers from the IBM cryptographic team.

The most prominent example of a Feistel type cipher is probably the Data Encryption Standard (DES). Recall that in addition to a Feistel network, DES uses an initial permutation ($\mathcal{IP}$) on the input 64-bit plaintext block, a final permutation (the inverse of $\mathcal{IP}$) on the last sixteenth block, and has a secret key size of 56 bits. The block cipher Lucifer, a predecessor of DES developed by the IBM crypto-group in the early 70s, belongs to the same family but has plaintext block and key size of 128 bits [2]. More recent proposals of Feistel type ciphers (some of them in a wider sense) are: Triple-DES, the Soviet GOST 28147-89, Blowfish, CAST-128, etc.

The security of a block cipher depends in a crucial way on the so-called S-boxes, the non-linear mappings at the core of the cipher. From the cryptanalyst's point of view some S-boxes could be much more malleable to carry out a particular kind of analysis and to perform the corresponding attacks. Of course, S-boxes, like affine ones, which are extremely vulnerable to linear cryptanalysis, are easily distinguishable. However, there are some classes of S-boxes looking innocent at the first sight but still having a hidden trap-door structure. The reader is referred to [3] for details of a more general probabilistic approach to constructing such S-boxes, and to [4] for a description of an attack on Feistel ciphers with a non-surjective round function such as the CAST cipher family and LOKI91. Another more recent contribution to this topic is given in [5].

The goal of this paper is to re-warn the developers and customers about the existence of such phenomena and at the same time to make clear in an explicit manner how dangerous it can be in practice for the case of DES-like cryptographic systems. For example, such trouble may occur when system of this kind implemented in software (and downloaded from an unreliable WEB site) has been deliberately damaged by an adversary unbeknownst to the ordinary users. A similar situation is considered in the context of symmetric block ciphers based on a Feistel network with secret S-boxes installed as an additional parameter [6] (as in case of GOST 28147-89) but the approach presented there leads only to a limited reduction in the key space per block.

The remaining of the paper is organized as follows. In the next section, we recall some background needed to present our results. Then in Section 3, we describe a ciphertext-only attack on DES-like cryptographic systems with a special kind of non-surjective S-boxes. In Section 4, a backtracking search algorithm for carrying out such an attack against proper modifications of DES/Triple-DES used for enciphering of a typical English plaintext source is presented. Finally, in the last section some conclusions are drawn.

2. Background. A Feistel cipher (network), named after the cryptographer Horst Feistel, is an iterative block cipher, which splits the input block at the ith round into two parts L_{i-1} and R_{i-1}. Then R_{i-1} will form the left part of the next block, while the left part L_{i-1} will be combined with the output of the encryption function f of key K_i applied to R_{i-1}. More precisely,

$$\begin{aligned} L_i &= R_{i-1} \\ R_i &= L_{i-1} \oplus f_i(R_{i-1}), \end{aligned} \tag{1}$$

where $f_i = f_{K_i}(.)$ is the round function, and the sub-keys K_i are all derived by the key scheduling from the secret key K. Decryption is done using

$$\begin{aligned} R_{i-1} &= L_i \\ L_{i-1} &= R_i \oplus f_i(L_i) \end{aligned}$$

regardless of what the round function looks like (can be quite complex and not necessarily invertible).

We recall that a cryptographic $m \times n$ S-box can be regarded as a mapping $S : {\mathbb{F}_2}^m \mapsto {\mathbb{F}_2}^n$, where $\mathbb{F}_2 = \mathbf{GF}(2)$, $m \geq n$, which satisfies additional criteria (see, e.g., [7][Ch. 9]) that make the cipher resistant to the known kinds of cryptanalysis (linear, differential, algebraic, etc.). Note also that in Feistel ciphers, the S-boxes are embedded in the round function and are the only non-linear part of the cipher. For instance, in DES, the round function looks as follows:

$$f_i(R) = \mathcal{P}(\mathcal{S}(\mathcal{E}(R) \oplus K_i)), \tag{2}$$

where $\mathcal{P}$ is a fixed round permutation acting on a 32−bit block; $\mathcal{S} : {\mathbb{F}_2}^{48} \mapsto {\mathbb{F}_2}^{32}$ is a mapping produced by concatenating the outputs of the eight DES S-boxes; $\mathcal{E} : {\mathbb{F}_2}^{32} \mapsto {\mathbb{F}_2}^{48}$ is the expansion function of DES. (All these components are set in concrete specifications of the former U.S. standard.)

Also, we assume the reader has some background in Coding Theory (see, e.g., [8]). Further on, we shall make use of the following simple:

Proposition 1. *Let $\mathcal{U}$ be a coset of a binary linear code $\mathcal{C}$, and $\mathbf{u}_1, \mathbf{u}_2, \dots \mathbf{u}_t$ be t vectors from $\mathcal{U}$. Then $\sum_{i=1}^{t} \mathbf{u}_i \in \mathcal{C}$ if and only if $t \equiv 0 \pmod 2$, otherwise $\sum_{i=1}^{t} \mathbf{u}_i \in \mathcal{U}$.*

3. Ciphertext-only attack on DES-like cryptosystems with special non-surjective S-boxes.

3.1. Attack rationale for a Feistel cipher with S-boxes of a special kind. First, consider a Feistel network with $2t$ rounds, and let the number of used S-boxes be s. We shall assume also that the round function is of the form representing by equation (2), and the outputs of all S-boxes are from $\mathbb{F}_2{}^n$. So, the round permutation $\mathcal{P}$ acts on an $s \times n$-bit block. For example, in the case of DES we have: $t = 8, s = 8$, and $n = 4$.

Using equation (1) by induction one can easily derive the following expressions for the output block of the last round:

$$\begin{aligned} L_{2t} &= L_0 \oplus \sum_{i=1}^{t} f_{2i-1}(R_{2i-2}) \\ R_{2t} &= R_0 \oplus \sum_{i=1}^{t} f_{2i}(R_{2i-1}), \end{aligned} \tag{3}$$

where $(L_0 R_0)$ is the input block of length $2 \times sn$ bits, and the sub-keys K_i (respectively the secret key K) are present implicitly.

Now, let us focus on the two XOR sums:

$$\begin{aligned} X_l &= \sum_{i=1}^{t} f_{2i-1}(R_{2i-2}) \\ X_r &= \sum_{i=1}^{t} f_{2i}(R_{2i-1}), \end{aligned} \tag{4}$$

from the right side of equation (3). Rewriting it as follows

$$\begin{aligned} L_{2t} &= L_0 \oplus X_l \\ R_{2t} &= R_0 \oplus X_r \end{aligned} \tag{5}$$

one can interpret the considered Feistel cipher as a binary additive stream cipher. (Notice that the keystream depends on plaintext, thus it is a self-synchronizing stream cipher.) Of course, the above can be done always but the expressions for X_l and X_r (see, equation (4)) are very important in some cases of interest as we will show bellow.

Assume now that the image of each S-box is some fixed proper affine subspace of the binary vector space $\mathbb{F}_2{}^n$, i.e., a coset of some proper linear code in $\mathbb{F}_2{}^n$. Since as a permutation $\mathcal{P}$ is linear (preserves the XOR operation and can be taken "out of the brackets") it follows from equation (4) that both X_l and X_r, belong to a permuted version of the Cartesian product of the corresponding subspaces. The latter being by Proposition 1 either the images of S-boxes in case $t \equiv 1 \pmod 2$, or the linear codes in ${\mathbb{F}_2}^n$ whose cosets are these images, in case $t \equiv 0 \pmod 2$. Therefore both $Y_l = \mathcal{P}^{-1}(X_l)$ and $Y_r = \mathcal{P}^{-1}(X_r)$ can be regarded as sequences of random variables taking their values in proper subsets of the whole ${\mathbb{F}_2}^n$. This is the reason (the information-theoretical basis) for the feasibility of breaking the kind of Feistel cipher considered here, provided the plaintext source redundancy is sufficiently large (see, e.g., [9] or [10][p. 400] for more general discussion on the last topic).

Remark 1. We emphasize the fact that although the images of the S-boxes are assumed to be proper affine subspaces, by no means these S-boxes are affine. For instance, keeping three of the coordinate functions in an S-box of DES and setting the remaining to be the parity check of the others, the resulting new S-box is apparently not affine. But of course all S-boxes of this kind are non-surjective over ${\mathbb{F}_2}^n$.

Remark 2. Note as well that the described approach for attacking the specified kind of Feistel networks points out a possibility to recover the plaintext (input block (L_0R_0)) without actually finding the secret key K.

Remark 3. The restriction to an even number of rounds is not essential and the above considerations can be properly adjusted for an odd number of rounds making use of the defining equation (1).

3.2. Attack rationale for DES-like systems based on relevant Feistel network. Second, we shall take into account the initial permutation $\mathcal{IP}$ and final permutation $\mathcal{IP}^{-1}$ present in any DES-like cryptosystem.

Let $\mathbf{P}$ be the instant plaintext block, while $\mathbf{C}$ is the corresponding ciphertext block in the attacked cryptosystem. Then for the input block of the included Feistel network we have: $(L_0R_0) = \mathcal{IP}(\mathbf{P})$, while for the input block of the final permutation it holds: $(R_{2t}L_{2t}) = \mathcal{IP}(\mathbf{C})$ (see, e.g., Fig. 11 [10][p. 410] in case of DES). Therefore, $(L_{2t}R_{2t})$ equals to $swap(\mathcal{IP}(\mathbf{C}))$, and by equations (5) we get: $swap(\mathcal{IP}(\mathbf{C})) = ((L_0 \oplus X_l)(R_0 \oplus X_r)) = (L_0R_0) \oplus (X_lX_r) = \mathcal{IP}(\mathbf{P}) \oplus (X_lX_r)$. Finally, taking into account that $X_l = \mathcal{P}(Y_l)$ and $X_r = \mathcal{P}(Y_r)$ we obtain:

$$\mathcal{IP}^{-1}((\mathcal{P}(Y_l)\mathcal{P}(Y_r))) = \mathcal{IP}^{-1}(swap(\mathcal{IP}(\mathbf{C}))) \oplus \mathbf{P}. \tag{6}$$

Remark 4. The last equation is very useful if we attempt to solve the cryptogram by guessing the plaintext $\mathbf{P}$ since both Y_l and Y_r in its left side are constrained for the special type of Feistel cipher we consider, and $\mathcal{IP}^{-1}(swap(\mathcal{IP}(\mathbf{C})))$ can be computed only once in a preprocessing step of the algorithm.

Remark 5. So far we have not given an explicit definition of DES-like cryptosystem. In this paper, we mean that DES-like system is a superposition of an initial permutation, Feistel network, and final (inverse of the initial) permutation preceded by a swap. In this sense Triple-DES is not truly a DES-like system. But one can easily see that equation (6) is still valid since the permutations and the two additional swaps on the borders between the consecutive stages of this cryptalgorithm are canceled in the process of its deduction. So, in general the attack against a triple application of one DES-like system proceeds along the same lines as against the single one when using the last equation (in particular, this is true for Triple-DES and DES themselves).

3.3. A representative example: DES with S-boxes having parity check. We will exemplify the described attack in case of DES/Triple-DES (ECB mode) with suitably modified S-boxes.

To obtain the S-box of interest, it suffices to change an S-box of DES in the following way: Set one chosen coordinate function to be an even (or odd) parity check of the other three which are kept unchanged.

The image of a S-box modified in this way is either the even weight code $\mathcal{E}_4$ of length 4 or its unique nontrivial coset $\mathcal{O}_4$ consisting of all 4-bit tuples having odd weights. Therefore, for the image $\Im(S)$ of any such S-box: $|\Im(S)| = |\mathcal{E}_4| = 8$.

However, these S-boxes are non-affine and even satisfy other desirable properties of S-boxes (e.g., the criteria **C2a.** and **C2b.** from [7][p. 301]). Hereinafter, for the reader's convenience we recall those criteria:

- **C2a.** Changing one bit in the input of an S-box results in at least two output bits changing;
- **C2b.** If two inputs to an S-box differ in the middle two bits, their outputs must be different by at least two bits [11].

The reason that the above criteria are automatically satisfied is the following: If two outputs of the original S-box of DES differ in at least two positions they are replaced by different vectors in the modified S-box, therefore as these vectors belong to a coset of $\mathcal{E}_4$ the Hamming distance between them is 2 or 4. This means that properties **C2a.** and **C2b.** are preserved.

Moreover, that modification provides an opportunity to introduce the so-called customers's key by specifying the position and kind of the parity bits. (Recall that customer's keys are those keys which separate distinct user communication groups.) So, in case of DES there are $8^8 = 2^{24}$ possibilities for such a key. Additional diversity is achieved by XORing the outputs of any S-box thus modified with a fixed 4-bit tuple.

The 32-bit blocks Y_l and Y_r defined above will look as follows:

$$Y_l = (S'_1 S'_2 \dots S'_8)$$
$$Y_r = (S''_1 S''_2 \dots S''_8),$$

where S'_i and S''_i, $1 \leq i \leq 8$, are 4-bit tuples corresponding to the ith modified DES S-box. Since DES has 16 rounds ($t = 8$), by the general considerations from Section 3.1 this implies that all S'_i and S''_i, $1 \leq i \leq 8$ are of even weight. We shall refer to this property as a condition for even parity of a given S-box, or as **EP** condition for short.

Denote by $P_1, P_2, \dots, P_8$ the consecutive bytes of the plaintext block **P**. Equation (6) determines how the bytes P_j, $1 \leq j \leq 8$ influence S'_i, $1 \leq i \leq 8$, (respectively S''_i, $1 \leq i \leq 8$), through the initial permutation $\mathcal{IP}$ and the inverse round permutation $\mathcal{P}$ of DES. This dependence is summarized in Table 1. The presence of "x" in a cell of that table means that the corresponding byte affects the output of the corresponding S-boxes with one bit, while "xx" means the same but with two bits. For instance, looking at Table 1 one can see that P_2 influences S'_1 and S''_1 with two of its bits. The exact influence refined by a detailed analysis of permutations specified in FIPS-46 (see, e.g., [12]) is presented in the Appendix.

Table 1. The influence of plaintext bytes on outputs of S-boxes through equation (6)

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8
S'_1/S''_1		xx						xx
S'_2/S''_2				x	x		xx	
S'_3/S''_3	xx		xx					
S'_4/S''_4					x		xx	x
S'_5/S''_5	x		x			x		x
S'_6/S''_6				x	x	xx		
S'_7/S''_7	x	x	x		x			
S'_8/S''_8		x		xx		x		

There are two possibilities to exploit equation (6):

- Exhaust all possible "even parity" blocks ($Y_l Y_r$) storing only those plaintexts **P**s (obtained by the equation) which are "meaningful".
- Generate in some way the "meaningful" **P**s storing those of them satisfying the **EP** condition (checked by the equation) for all S-boxes.

The first possibility requires 2^{48} trials (i.e., the number of "even parity" blocks), which is undesirable. The second one is more promising for some plaintext sources, but still some reductions have to be made.

4. Performing the attack.

4.1. A backtracking search algorithm in two variants. To accomplish the task from the end of the previous section we have adapted two search algorithms. Both make intensive use of Table 1 (the Appendix) and a list $\mathcal{L}$ of plausible four-grams for a particular plaintext source. According to Remark 4, both have a preprocessing step which computes $\mathcal{IP}^{-1}(swap(\mathcal{IP}(\mathbf{C})))$ accepting as input the ciphertext block $\mathbf{C}$.

Also, they are based on the so-called backtracking search strategy which gradually builds candidates and abandons each partial candidate that cannot be completed to a valid solution (see, e.g., [13]).

Hereinafter, the steps of the first algorithm $\mathcal{A}$, are briefly described:
Step 1. Replace the right half $(P_5P_6P_7P_8)$ of the guessed plaintext which affect the whole S'_4/S''_4 with a new four-gram from $\mathcal{L}$ (if the list is exhausted then STOP) and check for the **EP** condition. If it is not satisfied then continue the step with the next four-gram, otherwise go to *Step 2.*
Step 2. Repeat the following until the list $\mathcal{L}$ is exhausted:

Replace the left half $(P_1P_2P_3P_4)$ of the guessed plaintext by a four-gram from $\mathcal{L}$ (together with the temporary fixed right half they affect all S-boxes), and check the **EP** condition. If in some trial the **EP** condition is satisfied, store the plausible solution (candidate). When the list $\mathcal{L}$ is exhausted return to *Step 1.*

Note that $\mathcal{A}$ exploits essentially only the fact of dependence of the pair S'_4/S''_4 from the right half of the guessed plaintext.

The second algorithm $\mathcal{A}'$ is a derivative of $\mathcal{A}$, which exploits as well the dependence of the pair S'_3/S''_3 from the left half of the guessed plaintext (see Table 1). $\mathcal{A}'$ is more efficient from a computational point of view, but at the expense of using an additional amount of memory.

The two steps of $\mathcal{A}'$ are described as follows:
Step $1'$. Replace the left half $(P_1P_2P_3P_4)$ of the guessed plaintext by all four-grams from list $\mathcal{L}$, check the **EP** condition for S'_3/S''_3, and when it is satisfied store the corresponding four-gram in some (new) array $\mathcal{L}'$.
Step $2'$. Execute $\mathcal{A}$ substituting in its *Step* 2 the list $\mathcal{L}$ by the array $\mathcal{L}'$.

Actually, one could see that $\mathcal{A}'$ is an appearance of the so-called "meet-in-the-middle" strategy for attacking the cryptographic schemes since it explores the halves of the plaintext block independently of each other and then combines the results (see, e.g., [14] where this technique was described for the first time).

A plausible assumption that when guessing plaintext, the event "given S-box satisfies the **EP** condition" holds with probability $\frac{1}{2}$ and these events are independent of each other, implies that the probability that all S-boxes satisfy the condition simultaneously equals $\frac{1}{2^{16}}$. Since the above algorithms explore the plaintext space $\mathcal{L} \times \mathcal{L}$, it follows that the expected cardinality of the list of candidates (the number of "spurious decipherments" + the true one) is about

$$|\mathcal{L}|^2/2^{16}. \tag{7}$$

We shall see further that the estimate is remarkably confirmed by our experiments. (The interested reader is referred to [15] for a general study of this phenomena in pure information-theoretic settings.)

Also, under the above assumption the computational complexity of the algorithm $\mathcal{A}$ can be easily estimated. Namely, *Step 1* is passed successfully by about $\frac{1}{4}|\mathcal{L}|$ four-grams from the list $\mathcal{L}$, because the **EP** condition must be satisfied for two S-boxes. And for every such right plaintext-half, *Step 2* is executed $|\mathcal{L}|$ times giving a total of around $\frac{1}{4}|\mathcal{L}|^2 + |\mathcal{L}|$ trials per block.

Regarding $\mathcal{A}'$, again *Step* $1'$ is passed successfully by the same amount of four-grams from $\mathcal{L}$ and thus the virtual memory for $\mathcal{L}'$ is of magnitude $\frac{1}{4}|\mathcal{L}|$. Further on (processing as above), we easily conclude that around $\frac{1}{16}|\mathcal{L}|^2 + 2|\mathcal{L}|$ trials per block are examined in the whole algorithm. Hence on the average, $\mathcal{A}'$ is four times faster than $\mathcal{A}$.

4.2. The language model for candidate's evaluation and ranking. The next stage of our attack consists of evaluating each potential solution from the obtained list and ranking them correspondingly.

To this end, we make use of the relevant plaintext language model assuming that the plaintext is a typical natural-language text, e.g., in English. Traditionally, in an n-gram language model the probability of a sentence $P_1, \ldots, P_m$, $m \geq n$ is assumed in the context history of the preceding $n-1$ letters, namely:

$$Prob(P_1, \ldots, P_m) \approx \prod_{i=1}^{m} Prob(P_i | P_{i-n+1} P_{i-n+2} \ldots P_{i-1})$$

The above is in connection to the so-called Markov model of natural languages (see, e.g., [16, Ch. 6]). For other interesting cryptanalytic applications of that model the reader is referred to [17] and [18].

In our case $n = 4$ and $m = 8$ (the block length). We use, as well, an approximation of the above product of probabilities by:

$$\prod_{i=1}^{8} \frac{\#(P_{i-3}P_{i-2}P_{i-1}P_i)}{\#(P_{i-3}P_{i-2}P_{i-1})}, \tag{8}$$

where $\#(.)$ means the frequency count of the corresponding n-gram in a sufficiently large representative text from the plaintext source. For the sake of completeness, we note that trigram $P_{-2}P_{-1}P_0$ is taken either from an already solved preceding plaintext block, or if the current plaintext block should start with a new word (as in the case of initial plaintext block) that trigram is replaced by the triply repeated symbol "space". We will refer to $P_{-2}P_{-1}P_0$ as prefix.

In fact, in our software program a logarithmic scale is used in order to avoid rounding errors, and the multiplications/divisions are replaced by the corresponding additions/subtractions. Also, a smoothing is done by setting the logarithms for missing four-grams to the "$-\infty$" value, i.e., a value smaller than all associated to those occurring.

Finally, we sort in descending order the list of candidates according to evaluations obtained by (8) in hope to get on the top the true (actually, maximum-likelihood) solution. In order not to skip that (correct) solution we store and display a certain part of the sorted list for further analysis such as human inspection.

4.3. Experiments with the implemented procedure and some comments. We have carried out our experiments under the assumption that the plaintext source is ordinary English with an alphabet consisting of 26 capital Roman letters augmented by "space", and all presented in the extended 8-bit ASCII code.

The "knowledge" about that source embedded in our computer program is the list $\mathcal{L}$ of the most frequent 4795 English four-grams (sorted by the frequency of their occurrence). Our statistics was made using two large English texts on social and natural topics. All letters were transformed to upper case, and the other characters are ignored. The statistics is slightly cropped down in order to avoid the selection of too many rare events, that is only four-grams with occurrences greater than 2 have been considered.

Our experiments show that the number of candidates for typical plaintext block is in the range $300 - 500$ which confirms the correctness of the estimate of its expected value (see (7)) because $|\mathcal{L}| = 4795$ is slightly over 2^{12}. Also, the time complexity per block of $\mathcal{A}$ is of order 2^{23}, while for $\mathcal{A}'$ it is of order 2^{21}.

Table 2. Some experiments: rank of solution / all candidates

true plaintext	prefix	rank / candidates
CH WE NO	WHI	1/351
RITICISM	T C	1/415
SCIENTIF	IF	1/295
S REFORM	IOU	1/315
MENTION	TO	1/308
THE PAR	ING	1/372
CTION OF	FLE	1/332
QUESTION		1/370
HEART S	THE	6/544

In Table 2 we present summarized results of some experiments with the program. The last row of the table shows that the true plaintext is not always on the top (the reason in this particular case is that the list of candidates includes as well other very plausible phrases like "BASE OF", "S NOTION", "EASE ON"). So, in general the process of plaintext recovery requires an expert's decision to choose the suitable candidate (in most cases to confirm the first rank plaintext), and hence to provide a secure prefix for the next block.

In the worst possible case the correct plaintext may be absent from the list of potential candidates (from a certain block all decipherments look meaningless), for instance when the embedded "knowledge" is not compatible with that plaintext. But these very rare cases could be repaired by the expert guessing the right phrase. Another possible strategy to overcome this problem is "to jump over the gap" and try to break the next block using a relaxation variant of the procedure which ignores the influence of the prefix in (8). At a later stage the cryptanalyst would go back and fill up the gap on the basis of newly yielded context of the message.

5. Conclusion. In this paper, we have pointed out the feasibility of breaking DES-like cryptosystems in which the set of possible outputs of each S-box is a proper affine subspace of its ambient binary space, in particular the extreme case of those having S-boxes with a parity check embedded in them. Also, we have presented a very efficient backtracking search algorithm (in two variants) performing that plaintext recovery attack on a specific example of modified S-boxes in DES/Triple-DES (ECB mode) and verified it experimentally for a typical English plaintext source. But evidently this algorithm can be applied with the same strength to any plaintext source possessing a sufficiently large redundancy, for instance to high order ergodic Markov sources of relatively low entropy (which can be applied for approximating adequately the natural languages [16]). In this way, we show that while such S-boxes do not look very weak at first glance, and even satisfy several of the common design criteria, they make the cipher completely insecure, allowing to retrieve the plaintext in a ciphertext-only attack scenario.

Notice as well that plaintext block length of the aforementioned block ciphers (eight characters) allows the frequency tables of four-grams, which are relatively easy to collect, store and access, to be employed in a convenient and efficient way. However, for larger block lengths (as in the case of modified Lucifer, for instance) the described approach has no efficient straightforward extension.

This cryptanalytic algorithm can be easily adjusted to the other block cipher modes of operation, except the CFB and OFB modes with a partial block as feedback (see, e.g., [19]). And its working depends neither on the length of the secret key nor on the key itself (in the case of arbitrary DES-like system, of course, the number of rounds). However, the strength of our attack depends on the way the round and initial/final permutation act on each other, which leads to a conclusion (somehow in contrast to the common belief) that initial and final permutations in DES do have a cryptographic significance for some attack scenarios, such as that studied in this paper.

Acknowledgments. This work was partially supported under the Bulgarian NSF grant I01/0003. Both authors would like to thank Svetla Nikova and the anonymous reviewers for helpful comments which substantially improved the manuscript. Yuri Borissov wishes to thank FWO-Belgium for support during his visit at Katholieke Universiteit Leuven in October 2012.

REFERENCES

[1] Shannon C. E. Communication theory of secrecy systems. *Bell System Technical Journal*, **28** (1949), No 4, 656–715.

[2] Sorkin A. Lucifer, a cryptographic algorithm. *Cryptologia*, **8** (1984), No 1, 22–41.

[3] Rijmen, V., B. Preneel. A family of trapdoor ciphers. In: Proceedings of the FSE'97, Zurich, 1997, 139–148.

[4] Rijmen, V., B. Preneel, E. De Win. On weaknesses of non-surjective round functions. *Des. Codes Cryptography*, **12** (1997), No 3, 253–266.

[5] Paterson K. G. Imprimitive permutation groups and trapdoors in iterated block ciphers. In: Proceedings of the FSE'99, LNCS, Vol. **1636**, Springer, 1999, 201–214.

[6] Oliynykov R. Cryptanalysis of symmetric block ciphers based on the Feistel network with non-bijective S-boxes in the round function. Cryptology ePrint Archive, 2011, #685.

[7] Konheim A. G. Computer Security and Cryptography. John Wiley & Sons, Inc., New Jersey, 2007.

[8] Pless V. Introduction to the theory of error-correcting codes. John Wiley & Sons, Inc., New York, Third edition, 1998.

[9] Massey J. L. An introduction to contemporary cryptology. *Proceedings of the IEEE*, **76** (1988), No 5, 533–549.

[10] Diffie W., M. Hellman. Privacy and authentication: an introduction to cryptography. *Proceedings of the IEEE*, **67**(1979), No 3, 397–427.

[11] Coppersmith D. The Data Encryption Standard (DES) and its strength against attacks. *IBM Journal of Research and Development*, **30** (1993), 243–250.

[12] Schneier B. Applied cryptography. John Wiley & Sons, Inc., New Jersey, second edition, 1996.

[13] Knuth D. E. The art of computing programming. Addison-Wesley, 1968.

[14] Diffie W., M. Hellman. Exhaustive cryptanalysis of the NBS Data Encryption Standard. *Computer*, **10** (1977), No 6, 74–84.

[15] Hellman M. An extension of the Shannon theory approach to cryptography. *IEEE Trans. on Information Theory*, **23** (1977), No 3, 289–294.

[16] Welsh D. Codes and cryptography. Oxford University Press Inc., New York, 1988.

[17] Griffin A. Solving XOR plaintext strings with the Viterbi algorithm. *Cryptologia*, **30** (2006), No 3, 258–265.

[18] Griffin A. Solving the running key cipher with the Viterbi algorithm. *Cryptologia*, **30** (2006), No 4, 361–367.

[19] `www.en.wikipedia.org/wiki/Block_cipher_mode_of_operation`

Vesela Angelova
Software Engineering Department
Institute of Mathematics
and Informatics
Bulgarian Academy of Sciences
Acad. G. Bontchev Str., Bl. 8
1113 Sofia, Bulgaria
e-mail: `vaa@math.bas.bg`

Yuri Borissov
Department Mathematical
Foundations of Informatics
Institute of Mathematics and Informatics
Bulgarian Academy of Sciences
Acad. G. Bontchev Str., Bl. 8
1113 Sofia, Bulgaria
e-mail: `youri@math.bas.bg`

Received October 16, 2013
Final Accepted January 22, 2014

Appendix. *Let $\mathcal{TP}$ denote a permutation acting on components of a 64-bit block as a composition of two permutations identical to $\mathcal{P}^{-1}$, each acting separately on the left and right halves of the block. Then the equation (6) can be rewritten as:*

$$(Y_lY_r) = \mathcal{TP}(swap(\mathcal{IP}(\mathbf{C}))) \oplus \mathcal{TP}(\mathcal{IP}(\mathbf{P})). \tag{9}$$

Using the concrete settings of the U.S. Federal Standard FIPS-46, it is easy to obtain the superposition $\mathcal{TP} \circ \mathcal{IP}$. Thus equation (9) which gives the way of constituting the bits of nibbles S'_i, and S''_i, $1 \leq i \leq 8$ by the corresponding bits of guessed plaintext $\mathbf{P} = (p_1, p_2, \dots p_{64})$ (with an accuracy up to a certain translation for the given ciphertext $\mathbf{C}$), implies the following:

$$\begin{aligned}
S'_1 &= (60, 62, 14, 16), & S''_1 &= (59, 61, 13, 15),\\
S'_2 &= (28, 40, 50, 54), & S''_2 &= (27, 39, 49, 53),\\
S'_3 &= (6, 4, 24, 18), & S''_3 &= (5, 3, 23, 17),\\
S'_4 &= (56, 38, 52, 58), & S''_4 &= (55, 37, 51, 57),\\
S'_5 &= (2, 20, 64, 42), & S''_5 &= (1, 19, 63, 41),\\
S'_6 &= (34, 32, 44, 46), & S''_6 &= (33, 31, 43, 45),\\
S'_7 &= (8, 36, 22, 10), & S''_7 &= (7, 35, 21, 9),\\
S'_8 &= (26, 48, 12, 30), & S''_8 &= (25, 47, 11, 29),
\end{aligned}$$

where for the sake of simplicity only the indices of the plaintext bits are present. Finally, taking into account the positions of plaintext bytes $P_1, P_2, \dots P_8$, Table 1 can be easily derived.