Structural Cryptanalysis of SASAS

Alex Biryukov and Adi Shamir

Computer Science department
The Weizmann Institute
Rehovot 76100, Israel.

Abstract. In this paper we consider the security of block ciphers which contain alternate layers of invertible S-boxes and affine mappings (there are many popular cryptosystems which use this structure, including the winner of the AES competition, Rijndael). We show that a five layer scheme with 128 bit plaintexts and 8 bit S-boxes is surprisingly weak even when all the S-boxes and affine mappings are key dependent (and thus completely unknown to the attacker). We tested the attack with an actual implementation, which required just 2^{16} chosen plaintexts and a few seconds on a single PC to find the 2^{17} bits of information in all the unknown elements of the scheme.

Keywords: Cryptanalysis, Structural cryptanalysis, block ciphers, substitution permutation networks, substitution affine networks, Rijndael.

1 Introduction

Structural cryptanalysis is the branch of cryptology which studies the security of cryptosystems described by generic block diagrams. It analyses the syntactic interaction between the various blocks, but ignores their semantic definition as particular functions. Typical examples include meet in the middle attacks on double encryptions, the study of various chaining structures, and the properties of Feistel structures with a small number of rounds.

Structural attacks are often weaker than actual attacks on given cryptosystems, since they cannot exploit particular weaknesses (such as bad differential properties or weak avalanche effects) of concrete functions. The flip side of this is that they are applicable to large classes of cryptosystems, including those in which some of the internal functions are unknown or key dependent. Structural attacks often lead to deeper theoretical understanding of fundamental constructions, and thus they are very useful in establishing general design rules for strong cryptosystems.

The class of block ciphers considered in this paper are product ciphers which use alternate layers of invertible S-boxes and affine mappings. This structure is a generalization of substitution/permutation networks (in which the affine mapping is just a bit permutation), and a special case of Shannon's encryption paradigm which mixes complex local operations (called confusion) with simple global operations (called diffusion). There are many examples of substitution/affine ciphers in the literature, including Rijndael [4] which was recently

B. Pfitzmann (Ed.): EUROCRYPT 2001, LNCS 2045, pp. 394–405, 2001.
© Springer-Verlag Berlin Heidelberg 2001

selected as the winner of the Advanced Encryption Standard (AES) competition. Rijndael is likely to become one of the most important block ciphers in the next 20-30 years, and thus there is a great interest in understanding its security properties.

The best non-structural attack on Rijndael (and its predecessor Square [3]) is based on the *square attack* which exploits the knowledge of the S-box, the simplicity of the key schedule and the relatively slow avalanche of the sparse affine mapping (which linearly mixes bytes only along the rows and columns of some matrix and adds a subkey to the result). It can break versions with six S-box layers and six affine layers (a seventh layer can be added if the attacker is willing to guess its 128 bit subkey in a nonpractical attack).

In our structural attacks we do not know anything about the S-boxes, the affine mappings, or the key schedule, since they can all be defined in a complex key-dependent way. In particular, we have to assume that the avalanche is complete after a single layer of an unknown dense affine mapping, and that any attempt to guess even a small fraction of the key would require a nonpractical amount of time. Consequently, we cannot use the square attack (even though we are influenced by some of its underlying ideas) and we have to consider a somewhat smaller number of layers.

In this paper we describe surprisingly efficient structural attacks on substitution/affine structures with five to seven layers. The main scheme we attack is the five layer scheme $S_3 A_2 S_2 A_1 S_1$ (see Figure 1) in which each S layer contains k invertible S-boxes which map m bits to m bits, and each A layer contains an invertible affine mapping of vectors of $n = km$ bits over $GF(2)$:

$$A_i(x) = L_i x \oplus B_i$$

The only information available to the attacker is the fact that the block cipher has this general structure, and the values of k and m. Since all the S-boxes and affine mappings are assumed to be different and secret, the effective key length of this five layer scheme is [1]:

$$\log(2^m!)^{3 \cdot \frac{n}{m}} + 2\log(0.29 \cdot 2^{n^2}) \approx 3 \cdot 2^m (m - 1.44) \cdot \frac{n}{m} + 2n^2.$$

The new attack is applicable to any choice of m and n, but to simplify the analysis we concentrate on the Rijndael-like parameters of $m = 8$ bit S-boxes and $n = 128$ bit plaintexts. The effective key length of this version is about $3 \cdot 2^{12} \cdot 6.56 + 2^{15} \approx 113{,}000 \approx 2^{17}$ bits, and thus exhaustive search or meet in the middle attacks are completely impractical. Our attack requires only 2^{16}

[1] The probability that m randomly chosen linear equations in m unknowns are linearly independent over $GF(2)$ is:

$$\left(\frac{2^m - 1}{2^m}\right)\left(\frac{2^m - 2}{2^m}\right)\left(\frac{2^m - 2^2}{2^m}\right)\cdots\left(\frac{2^m - 2^{m-1}}{2^m}\right) = \prod_{l=1}^{m}\left(1 - \frac{1}{2^l}\right) > 0.288788. \quad (1)$$

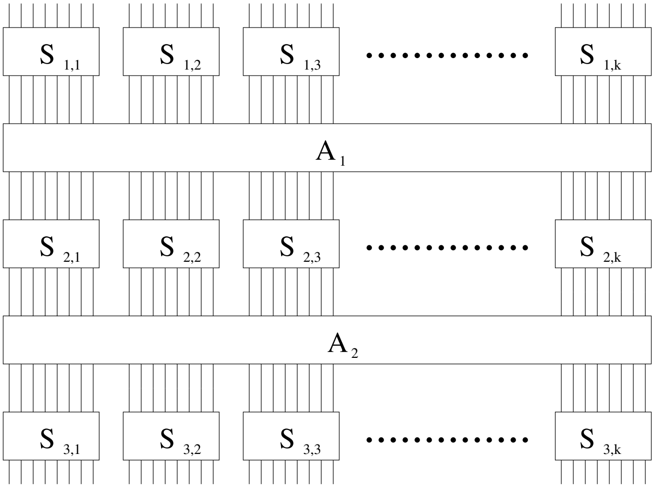

Fig. 1. Five-layer scheme.

chosen plaintexts and 2^{28} time to find all the unknown elements. This is quite close to the information bound since the 2^{16} given ciphertexts contain at most 2^{23} bits of information about the 2^{17} key bits.

It is important to note that not all the information about the S-boxes and the affine mappings can be extracted from the scheme, since there are many equivalent keys which yield the same mapping from plaintexts to ciphertexts. For example, we can change the order of the various S-boxes in a single layer and compensate for it by changing the definition of the adjacent affine mapping. In a similar way, we can move the additive constants in the affine mappings into the definition of the adjacent S-boxes. Our attack finds an equivalent representation of all the elements in the scheme which makes it possible to encrypt and decrypt arbitrary texts, but it may be different from the original definition of these elements.

A related structural attack on a five layer substitution/affine structure was recently published by Biham [2]. He attacked the slightly different structure $A_3S_2A_2S_1A_1$ (with two S-box layers and three affine layers) which was proposed by Patarin as a new algebraic public key cryptosystem called $2R$. However, in Patarin's scheme the S-boxes are implemented by multivariate quadratic polynomials, which are non-bijective due to design constraints. The starting point of Biham's attack is the existence of random collisions created by such S-boxes, and its time and data complexities were forced by the birthday paradox to be at least 2^{60}. Biham's attack is thus inapplicable to substitution/affine structures with invertible operations which have no collisions, and has higher complexity than our attack.

2 The Multiset Attack

2.1 Multiset Properties

In this section we develop a calculus of multiset properties, which makes it possible to characterize intermediate values deep in the encryption structure even though nothing is known about the actual functions in it. Each multiset can be represented as a list of (value, multiplicity) pairs (e.g., the multiset $\{1, 1, 1, 2, 2, 2, 2, 7\}$ can also be represented as $(1, 3), (2, 4), (7, 1))$. The size of the multiset is the sum of all its multiplicities (8 in this example). We now define several multiset properties:

Definition 1 *A multiset M of m-bit values has property C (constant) if it contains an arbitrary number of repetitions of a single value.*

Definition 2 *A multiset M of m-bit values has property P (permutation) if it contains exactly once each one of the 2^m possible values.*

Definition 3 *A multiset M of m-bit values has property E (even) if each value occurs an even number of times (including no occurrences at all).*

Definition 4 *A multiset M of m-bit values has property B (balanced) if the XOR of all the values (taken with their multiplicities) is the zero vector 0^m.*

Definition 5 *A multiset M of m-bit values has property D (dual) if it has either property P or property E.*

We will consider now the issue of how the multiset properties defined above are transformed by various mappings. In general if a bijective function is applied to a multiset we get a new multiset with possibly new values, but the same collection of multiplicities. If a non-bijective function is applied to a multiset, then the multiplicities of several distinct input values that are mapped to a common output value are added. The following observations are easy to prove:

Lemma 1
1. *Any multiset with either property E or property P (when $m > 1$) also has property B.*
2. *The E and C properties are preserved by arbitrary functions over m-bit values.*
3. *The P property is preserved by arbitrary bijective functions over m-bit values.*
4. *The B property is preserved by an arbitrary linear mapping from m bits to n bits when $m > 1$. It is preserved by arbitrary affine mappings when the size of the multiset is even.*

Let us consider now blocks of larger size $n = k \cdot m$ with mixed multiset properties. For example, we denote by $C^{i-1}PC^{k-i}$ a multiset with the property that when we decompose each n bit value into k consecutive blocks of m contiguous bits, $k - 1$ of the blocks contain (possibly different) constants across the multiset, and the i-th block contains exactly once each one of the 2^m possible m-bit values.

Similarly, we denote by D^k a multiset that decomposes into k multisets each one of which has property D. This decomposition should be understood not as a cross product of k multisets but as a collection of k projections of n bit to m bit values. Note that this decomposition operation is usually nonreversible, since we lose the order in which the values in the various blocks are combined. For example the multiset decomposition

$$\{0,1,2,3\}\{1,1,2,2\}\{1,1,1,1\}$$

(which has the multiset property PEC for $m = 2$) can be derived from several different multisets such as $\{(011),(111),(221),(321)\}$ or $\{(021),(121),(211),(311)\}$.

Let us consider now how these extended multiset properties are transformed by layers of S-boxes and affine mappings:

Lemma 2 *1. Property $C^{i-1}PC^{k-i}$ is preserved by a layer of arbitrary S-boxes provided that the i-th S-box is bijective.*

2. Property D^k is transformed into property D^k by a layer of bijective S-boxes.

3. Property D^k is transformed into B^k by an arbitrary linear mapping on n bits, and by an arbitrary affine mapping when the size of the multiset is even.

4. Property $C^{i-1}PC^{k-i}$ is transformed into property D^k by an arbitrary affine mapping when the size of the multiset is even.

Proof

The only non-trivial claims are 3 and 4. Let us show why claim 3 holds. Denote by

$$y_j = \sum_{i=1}^{n} d_{ji} x_i$$

a bit y_j at the output of the linear mapping. Property B holds since for each j, the sum (mod 2) of y_j bits over the 2^m elements of the multiset is zero:

$$\sum_{s=1}^{2^m} y_j^s = \sum_{s=1}^{2^m} \sum_{i=1}^{n} d_{ji} x_i^s = \sum_{i=1}^{n} d_{ji} \sum_{s=1}^{2^m} x_i^s = 0.$$

The last expression is zero since by Lemma 1, claim 1, both P and E (and thus D) imply the B-property. The result remains true even when we replace the linear mapping by an affine mapping if we XOR the additive constant an even number of times.

Let us now show why claim 4 holds. Any affine mapping over $GF(2)$ can be divided into k distinct n to m-bit projections. Since $(k-1)m$ of the input bits are constant, we will be interested only in restrictions of these affine mappings to new affine mappings that map the i-th block of m bits (the one which has the P property) into some other m-bit block in the output:

$$y = A_{ij}(x) = L_{ij} \cdot x \oplus B_j,\ j = 1, \dots k.$$

Here L_{ij} is an arbitrary $m \times m$ (not necessarily invertible) binary matrix and $B_j \in \{0,1\}^m$. We can again ignore B_j since it is XOR'ed an even number of times. If L_{ij} is invertible over $GF(2)$, then $L_{ij} \cdot x$ is a 1-1 transform and thus $L_{ij} \cdot x$ gets all the 2^m possible values when x ranges over all the 2^m possible inputs, so it has property P.

Thus we are left with the case of non-invertible L_{ij}. Suppose that

$$rank(L_{ij}) = r < m.$$

The kernel is defined as the set of solutions of the homogeneous linear equation $L_{ij} \cdot x = 0$. Let x_0 be some solution of the non-homogeneous equation $L_{ij} \cdot x = y$. Then all the solutions of the non-homogeneous equation have the form $x_0 \oplus v_0$, where v_0 is any vector from the kernel. The size of the kernel is 2^{m-r}, and thus each y has either no preimages or exactly 2^{m-r} preimages. Since $r < m$ by assumption, 2^{m-r} is even, and thus the multiset of m-bit results has property E. Consequently each block of m bits of the output has either property P or property E, and thus the n bit output has property D^k, as claimed.

2.2 Recovering Layers S_1 and S_3.

The first phase of the attack finds the two outermost layers S_1 and S_3, in order to "peel them off" and attack the inner layers.

Consider a multiset of chosen plaintexts with property $C^{i-1}PC^{k-i}$. The key observations behind the attack are:

1. The given multiset is transformed by layer S_1 into a multiset with property $C^{i-1}PC^{k-i}$ by Lemma 2, claim 1.
2. The multiset $C^{i-1}PC^{k-i}$ is transformed by the affine mapping A_1 into a multiset with property D^k by Lemma 2, claim 4.
3. The multiset property D^k is preserved by layer S_2, and thus the output multiset is also D^k, by Lemma 2, claim 2.
4. The multiset property D^k is not necessarily preserved by the affine mapping A_2, but the weaker property B^k is preserved.
5. We can now express the fact that the collection of inputs to each S-box in S_3 satisfies property B by a homogeneous linear equation. We will operate with m-bit quantities at once as if working over $GF(2^m)$ (XOR and ADD are the same in this field). Variable z_i represents the m-bit input to the S-box which produces i as an output (i.e., the variables describe S^{-1}, which is well defined since S is invertible), and we use 2^m separate variables for each S-box in S_3. When we are given a collection of actual ciphertexts, we can use their m-bit projections as indices to the variables, and equate the XOR of the indexed variables to 0^m. Different collections of chosen plaintexts are likely to generate linear equations with different random looking subsets of variables (in which repetitions are cancelled in pairs). When sufficiently many linear equations are obtained we can solve the system by Gaussian elimination in order to recover all the S-boxes in S_3 in parallel.

Unfortunately, we cannot get a system of equations with a full rank of 2^m. Consider the truth table of the inverted S-box as a $2^m \times m$-bit matrix. Since the

S-box is bijective, the columns of this matrix are m linearly independent 2^m-bit vectors. Any linear combination of the S-box input bits (which are outputs of the inverted S-box) is also a possible solution, and thus the solution space must have a dimension of at least m. Moreover, since all our equations are XOR's of an even number (2^m) of variables, the bit complement of any solution is also a solution. Since the system of linear equations has a kernel of dimension at least $m+1$, there are at most $2^m - m - 1$ linearly independent equations in our system. When we tested this issue in an actual implementation of the attack for $m = 8$, we always got a linear system of rank 247 in 256 variables, as expected from the formula.

Fortunately, this rank deficiency is not a problem in our attack. When we pick any one of the non-zero solutions, we do not get the "true" S^{-1}, but $A(S^{-1})$, where A is an arbitrary invertible affine mapping over m-bits. By taking the inverse we obtain $S(A^{-1})$. This is the best we can hope for at this phase, since the arbitrarily chosen A^{-1} can be compensated for when we find $A(A_2) = A_2'$ instead of the "true" affine transform A_2, and thus the various solutions are simply equivalent keys which represent the same plaintext/ciphertext mapping.

A single collection of 2^m chosen plaintexts gives rise to one linear equation in the 2^m unknowns in each one of the k S-boxes in layer S_3. To get 2^m equations, we can use 2^{2m} (2^{16}) chosen plaintexts of the form (A, u, B, v, C), in which we place the P structures u and v at any two block locations, and choose A, B, C as arbitrary constants. For each fixed value of u, we get a single equation by varying v through all the possible 2^m values. However, we can get an additional equation by fixing v and varying u through all the 2^m possible values. Since we get $2 \cdot 2^m$ equations in 2^m unknowns, we can reduce the number of chosen plaintexts to $\frac{3}{4} \cdot 2^{2m}$ by eliminating the $\frac{1}{4}$ of the plaintexts in which u and v are simultaneously chosen in the top half of their range. The matrix of these (u, v) values has a missing top-right quarter, and we get half the equations we need from the full rows and half the equations we need from the full columns of this "L" shaped matrix.

Solving each system of linear equations by Gaussian elimination requires 2^{3m} steps, and thus we need $k2^{3m}$ steps to find all the S-boxes in S_3. For the Rijndael-like choice of parameters $n = 128$, $m = 8$ and $k = 16$, we get a very modest time complexity of 2^{28}.

To find the other external layer S_1, we can use the same attack in the reverse direction. However, the resultant attack requires both chosen plaintexts and chosen ciphertexts. In Section 3 we describe a slightly more complicated attack which requires only chosen plaintexts in all its phases.

2.3 Attacking the Inner Layers ASA

The second phase of the attack finds the middle three layers. We are left with a structure $A_2' S_2 A_1'$ – two (possibly modified) affine layers and an S-box layer in the middle. In order to recover the affine layers we use Biham's low rank detection technique from [2]. Consider an arbitrary pair of known plaintexts P_1 and P_2 with difference $P_1 \oplus P_2$. With probability $k/2^m$, after A_1' there will be no difference at the input to one of the k S-boxes in S_2. Thus there will also be no difference at the output of this S-box. Consider now the set of pairs $P_1 \oplus C_i$, $P_2 \oplus C_i$ for many randomly chosen n-bit constants C_i. Any pair in this set still has this property, and thus the set of all the obtained output differences after A'_2 will have a rank of at most $n - m$, which is highly unusual for random n dimensional vectors. Consequently, we can confirm the desired property of the original pair P_1 and P_2 by applying this low rank test with about n modifiers C_i.

We want to generate and test pairs with zero input differences at each one of the k S-boxes. We choose a pool of t random vectors P_j and another pool of n modifiers C_i, and encrypt all the nt combinations $P_j \oplus C_i$. We have about $t^2/2$ possible pairs of P_j's, each one of them has a probability of $k/2^m$ to have the desired property at one of the S-boxes, and we need about $k \cdot log(k)$ random successes to cover all the k S-boxes. The critical value of t thus satisfies $t^2/2 \cdot k/2^m = k \cdot log(k)$ and thus $t = \sqrt{2^{m+1}log(k)}$. For $n = 128$ $m = 8$ and $k = 16$ we get $t = 2^{5.5}$, and thus the total number of chosen plaintexts we need is $nt = 2^{12.5}$, which is much smaller than the number we used in the first phase of the attack.

Now we use linear algebra in order to find the structure of A'_2. Consider the representation of A'_2 as a set of n vectors $V_0, V_1, \ldots V_{n-1}$, $V_i \in \{0,1\}^n$, where A'_2 transforms an arbitrary binary vector $b = b_0, b_1, \ldots b_{n-1}$ by producing the linear combination:

$$A'_2(b) = \bigoplus_{i=0}^{n-1} b_i V_i.$$

(we can ignore the affine constants viewing them as part of the S-box). From the data pool we extract information about k different linear subspaces of dimension $n - m$ $(= 120)$. Then we calculate the intersection of any $k - 1 (= 15)$ of them. This intersection is an m-dimensional linear subspace which is generated by all the possible outputs from one of the S-boxes in layer S_2, after it is expanded from 8 bits to 128 bits by A'_2. We perform this operation for each S-box and by this we find a linear mapping A^*_2 which is equivalent to the original choice. The complexity of this phase is that of Gaussian elimination on a set of $O(n - m)$ equations.

After finding and discarding A'_2, we are left with the two layer structure $S_2 A'_1$. If we need to perform only decryption, we can recover this combined mapping by writing formal expressions for each bit, and then solving the linear equations with $k2^m$ (2^{12}) variables. If we also need to perform encryption this trick will not work, since the formal expressions will be huge. However, we can just repeat our attack in the reverse direction by using chosen ciphertexts and recover A^*_1. After that we can find the remaining layer S_1 with about 2^m known plaintexts. Again we will find not the real S-box layer S_2 but the equivalent one which corresponds to the modified A^*_1, A^*_2 that we have found in earlier phases.

Comment: for one of the mappings we need to know the order of the subspaces: we can assume arbitrary order of subspaces in A_2 together with arbitrary order of S-boxes in S_2, however at this point the order of subspaces in A_1 is no longer arbitrary. If after finding A_2 we mount the same attack on $S_2 A_1$ from the ciphertext direction, we can recover A_1' together with the correct ordering information.

The complete attack uses about 2^{2m} chosen plaintexts (2^{16}) and about $k2^{3m}$ ($16 \cdot 2^{24} = 2^{28}$) steps. We tested the attack with an actual implementation, and it always ended successfully after a few seconds of computation on a single PC. The attack remains practical even if we increase the size of the plaintexts from 128 to 1024 bits and replace the 8-bit S-boxes by 16-bit S-boxes, since with these parameters the attack requires 2^{32} chosen plaintexts and $64 \cdot 2^{3 \cdot 16} = 2^{54}$ time.

3 A Chosen Plaintext Attack on $ASAS$

In this section we show how to use a pure chosen plaintext attack, and avoid the less realistic chosen plaintext and chosen ciphertext attack. The modified attack has the same time and data complexities as the original attack.

After the first phase of the original attack we are left with a $A_2'S_2A_1S_1$ structure, since we can recover only one of the two external S-box layers. Since the inputs go through the additional S-box layer S_1, we can no longer argue that for any C_i, $P_1 \oplus C_i$ and $P_2 \oplus C_i$ will have a zero difference at the input to some S-box in S_2 whenever P_1 and P_2 have this property. We thus have to use a more structured set of modifiers which can be nonzero only at the inputs to the S-boxes in which P_1 and P_2 are identical.

For the sake of simplicity, we consider in this section only the standard parameters. We use 2^{16} chosen plaintexts with the multiset property PPC^{k-2} (the two P's could be placed anywhere, and we could reuse the chosen plaintexts from the first phase of the attack). There are 2^{15} different ways to choose a pair of values from the first P. For each such pair (a_1, a_2), we generate a group of 2^8 pairs of extensions of the form $(a_1, b_0, c, d, \ldots)$ and $(a_2, b_0, c, d, \ldots)$ where b_0 is any common element from the second P, and $c, d, \ldots$ are the constants from C^{k-2}. We claim that all these 2^8 pairs will have the same difference at the output of S_1, since the first S-box gets a fixed pair of values and the other S-boxes get identical inputs in each pair. We can now apply the low rank test since we have sufficiently many choices of (a_1, a_2) to get a zero difference at the input to each S-box in S_2 with high probability, and for any such (a_1, a_2) we have sufficiently many pairs with the same difference in order to reliably test the rank of the output vectors. Once we discover the partition of the output space into 16 different linear subspaces of dimension 120, we can again find the intersection of any 15 of them in order to find the 8 dimensional subspace generated by the outputs of each one of the 16 S-boxes. We fix A_2' by choosing any set of 8 arbitrary spanning vectors in each one of the 16 subspaces, and this is the best we can possibly do in order to characterize A_2' due to the existence of equivalent keys.

One possible problem with this compact collection of plaintexts is that the attack may fail for certain degenerate choices of affine mappings. For example, if both A_1 and A_2 are the identity mapping, the insufficiently mixed intermediate values always lead to very low output ranks. However, the attack was always successful when tested with randomly chosen affine mappings.

After peeling off the computed A'_2, we are now left with a $S'_2A_1S_1$ structure, which is different from the $A'_2S_2A'_1$ structure we faced in the original attack. We have already discovered in the previous part of the attack many groups of 256 pairs of plaintexts, where in each group we know that the XOR of each pair of inputs to any particular S-box in S'_2 is the same constant. We do not know the value of this constant, but we can express this property as a chain of homogeneous linear equations in terms of the values of the inverse S-box, which are indexed by the known outputs from the $S'_2A_1S_1$ structure. A typical example of the equations generated from one group is

$$S^{-1}(1) \oplus S^{-1}(72) = S^{-1}(255) \oplus S^{-1}(13) = S^{-1}(167) \oplus S^{-1}(217) = \dots$$

If we need additional equations, we simply use another one of the 2^{15} possible groups of pairs, which yields a different chain of equations (with a different unknown constant). Note that these sparse linear equations are completely different from the dense equations we got in the first phase of the attack, which expressed the B property by equating the XOR's of various random looking subsets of 256 variables to 0^m.

We are finally left with a simple A'_1S_1 structure. It can be attacked in a variety of ways, which are left as an exersise for the reader.

Comments:

- The attack works in exactly the same way if the affine mappings are over finite fields with even characteristic. In particular, it can be applied to Rijndael-like schemes in which the affine transforms are over $GF(2^8)$.
- The attack can be extended to the case where S_2 contains arbitrary random (not necessarily bijective) S-boxes with a small penalty in the number of chosen plaintexts. Direct application of our attack will not work, since the P property at the input to some S-box in layer S_2 may not result in a balanced output after S_2 if this particular S-box is non-bijective. In order to overcome this dificulty we can work with double-sized $2m$-bit S-boxes at layer S_1. We consider a projection mapping $PT1$ from $2m$ to m bits (in the affine mapping A_1) which necessarily has a non-zero kernel (and thus always has the E property which is preserved even by non-bijective S-boxes, and not the P property which is not preserved by non-bijective S-boxes). The attack works in exactly the same way with the exception that we pay a factor of 2^m in data and in the process of equation preparation (now each equation is the XOR of 2^{2m} variables instead of 2^m). The total complexity of the attack becomes 2^{3m} chosen plaintexts and $k2^{3m}$ steps.
- We can attack the scheme even if a sparse linear mapping (a bit permutation or a mapping that mixes small sets of bits like the Serpent [1] mappings) is added to the the input. The attack works as long as we can guess columns of the linear mapping that correspond to the inputs of one particular S-box in S_1. If we add an initial bit permutation with the standard parameters, we can guess which 8 plaintext bits enter this S-box, and construct the

$C^{i-1}PC^{k-i}$ structure we need to get each linear equation with just this knowledge. Note that to generate the P property we can choose these 8 bits in an unordered way, and to generate the other C^{k-1} property we don't care about the destination of the other bits under the bit permutation, and thus the number of cases we have to consider is at most $\binom{128}{8} \approx 2^{40}$. By increasing the time complexity of the attack by this number, we get a (barely practical) attack on this six layer scheme. By symmetry, we can also attack the scheme in which the additional bit permutation layer is added at the end, and with a somewhat higher complexity we can attack the seven layer scheme in which we add unknown bit permutations both at the beginning and at the end of the scheme. It is an open problem whether we can attack with reasonable complexity six layer schemes with a general affine mapping added either at the beggining or at the end.

- We can attack the scheme even if the S-boxes have inputs of different sizes which are unknown to the attacker, since this information will be revealed by rank analysis.
- We can attack modified schemes which have various types of feedback connections between the S-boxes in the first and last rounds (see Figure 2 for one example). The idea is that we still have some control over multisets in such construction: We can cause the rightmost S-box to run through all the possible inputs (if the XORed feedback is a constant) and thus can force multisets to have the $C^{k-1}P$ property after S_1 even when the indicated feedback connections are added. The extraction of the S-boxes in the last layer S_3 has to be carried out sequentially from right to left, in order to take into account the effect of the feedbacks at the bottom.
- The attack stops working if S_3 contains non-bijective S-boxes. One can estimate the sizes of the equivalence (collision) classes of the outputs of the particular S-box. However even writing the linear equations does not seem possible: If we get the same output value twice in our structure, we cannot tell which variables should be used as the input of the S-box in each case.

Acknowledgements

We thank David Wagner for a very useful early exchange of ideas, and Anton Mityagin for implementing and testing the attack described in this paper.

References

1. R. Anderson, E. Biham, L. Knudsen, *Serpent: A Proposal for the AES*, 1st AES Conference, 1998.
2. E. Biham, *Cryptanalysis of Patarin's 2-Round Public Key System with S-boxes (2R)*, proceedings of EUROCRYPT'2000, LNCS 1807, pp.408–416, Springer-Verlag, 2000.
3. J. Daemen, L. Knudsen, V. Rijmen, *The Block Cipher Square*, proceedings of FSE'97, LNCS 1267, pp.147–165, Springer-Verlag, 1997.
4. V. Rijmen, J. Daemen, *AES Proposal: Rijndael*, 1st AES Conference, 1998.

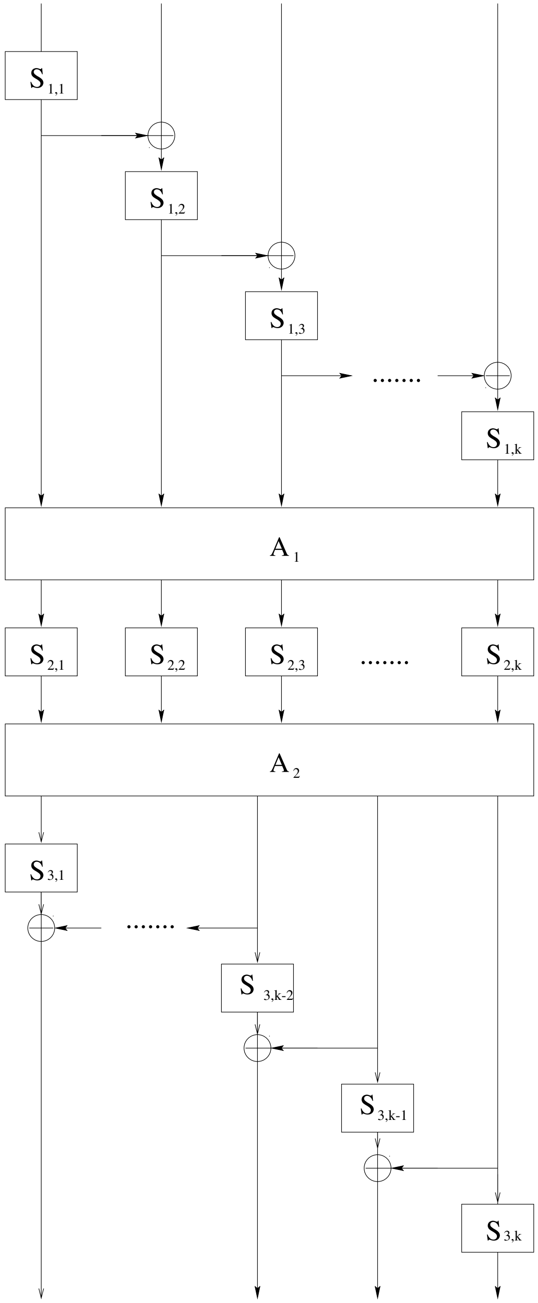

Fig. 2. Modified scheme with S-box feedbacks.