Security of the AES with a Secret S-box

Tyge Tiessen, Lars R. Knudsen, Stefan Kölbl, and Martin M. Lauridsen
{tyti,lrkn,stek,mmeh}@dtu.dk

DTU Compute, Technical University of Denmark, Denmark

Abstract. How does the security of the AES change when the S-box is replaced by a secret S-box, about which the adversary has no knowledge? Would it be safe to reduce the number of encryption rounds?
In this paper, we demonstrate attacks based on integral cryptanalysis which allow to recover both the secret key and the secret S-box for respectively four, five, and six rounds of the AES. Despite the significantly larger amount of secret information which an adversary needs to recover, the attacks are very efficient with time/data complexities of $2^{17}/2^{16}$, $2^{38}/2^{40}$ and $2^{90}/2^{64}$, respectively.
Another interesting aspect of our attack is that it works both as chosen plaintext and as chosen ciphertext attack. Surprisingly, the chosen ciphertext variant has a significantly lower time complexity in the attacks on four and five round, compared to the respective chosen plaintext attacks.

Keywords: AES, integral cryptanalysis, secret S-box

1 Introduction

The Advanced Encryption Standard (AES) [10] is an iterated block cipher using 10, 12, or 14 rounds depending on the key size of 128, 192, or 256 bits. These variants are named AES-128, AES-192, and AES-256.

In this paper we consider the cipher that is derived from the AES by replacing the S-box with a secret 8-bit S-box while keeping everything else unchanged. If the choice of S-box is made uniformly at random from all 8-bit S-boxes, the size of the secret information increases from $128 - 256$ bits, the key size in the AES, to $1812 - 1940$ bits. Clearly the security level of such a cipher could be very high, thus the question is: Could the number of rounds of this cipher be reduced to fewer than 10 rounds (as in AES-128)?

The AES was designed in order to achieve good resistance against differential and linear cryptanalysis, and this includes the choice of the S-box. Nonetheless a randomly chosen S-box is very likely to be highly resistant against these attacks as well.

The method that is most successful in attacking AES for up to 6 rounds is integral cryptanalysis. Somewhat surprisingly, a variant of this attack also applies to the AES variant with a secret S-box with up to 6 rounds, and although the complexity of the attack is larger than for the attack on the original AES, the time complexity is still less than exhaustive search of a 128-bit key.

Related Work. The idea of integral cryptanalysis was conceived as a dedicated attack against the block cipher SQUARE [3]. This attack is able to break up to six rounds of AES-128. Biryukov and Shamir applied integral cryptanalysis to a generalised SPN structure denoted SASAS [1], which consists of three substitution layers separated by two affine layers. In their paper, the attacker is assumed not to have any knowledge about the linear layer or the S-boxes which are all allowed to be chosen independently at random. The SASAS attack recovers an equivalent representation of this SPN and thus allows decryption of any ciphertext. The attack allows to break the equivalent of three rounds of AES. It does *not*, however, recover neither the key nor the S-box.

The case of the AES with a secret S-box, which we consider in this paper, lies in between two cases: The original SQUARE attack on one hand can not be directly applied to the case with the secret S-box as it requires knowledge of the S-box to peel off the last layer after guessing some key bits. The SASAS attack, on the other hand, can be used to attack three rounds of this cipher. However, it is not very effective, as the extra knowledge of the linear layer and the equality of all S-boxes remains unused.

The security of PRESENT with a secret S-box was studied by Borghoff et al. in [2] and allows an attack on 28 out of 31 rounds using slightly less than 2^{64} plaintexts. This attack was further improved by Liu et al. in [8]. As the attack depends on the weakness of some randomly chosen 4-bit S-boxes, it seems hard to apply it to the 8-bit S-boxes used in the AES.

Furthermore there are various block cipher designs based on using a secret, key-dependent substitutions like Khufu [9], Blowfish [14], Twofish [15] or Maya [7]. The attack also bears some resemblance to so-called SCARE (Side-Channel Analysis for Reverse Engineering) attacks in which side-channel information is used to recover unknown parts of cipher implementations (see for example [13]).

Our Contributions. We demonstrate that despite the increased size of the secret information in the cipher, we are able to recover both the secret key and the S-box for the 4-round, 5-round and 6-round versions of AES-128 by building up on techniques from integral cryptanalysis. Our attacks on four and five rounds are practical and achieve almost the same complexity as previous attacks which do not need to recover a secret S-box. The 6-round attack has a complexity of 2^{90} which is already much less than exhaustive search of the key, let alone of the S-box.

Table 1 compares the complexities for our attacks with those of previous integral attacks on AES-128 and the SASAS attack. Interestingly, the time complexities of the 4-round and 5-round attacks are lower by a factor of 2^{11} and 2^{16} respectively in the chosen ciphertext variant as compared to the chosen plaintext variant.

Organisation. This paper is organised as follows. In §2 the notation and a specification of the AES is given. In §3 we analyse the security of the AES

Table 1. Results of integral cryptanalysis on AES-128 with a secret S-box, AES-128 and SASAS with AES-like parameters. The time complexity is given in encryption equivalents, the data complexity is given in number of plaintexts/ciphertexts (16 bytes), the memory complexity is given in bytes. We assume that one round of encryption corresponds to 2^5 table lookups.

		Complexity			
Cipher	Rounds	Time	Data	Memory	Reference
SASAS	3	2^{21}	2^{16}	2^{20}	[1]
AES-128 (secret S-box)	4	2^{17}	2^{16}	2^{16}	*This work*
AES-128	4	2^{14}	2^{9}	–	[4]
AES-128 (secret S-box)	5	2^{38}	2^{40}	2^{40}	*This work*
AES-128	5	2^{38}	2^{33}	–	[4]
AES-128 (secret S-box)	6	2^{90}	2^{64}	2^{69}	*This work*
AES-128	6	2^{44}	2^{34}	2^{36}	[6]

with a secret S-box with respect to statistical and integral attacks. §4 holds the concluding remarks.

2 AES Specification

The AES [10] is an iterated block cipher that operates on 128-bit blocks and comes in three variants: AES-128, AES-192, and AES-256, which have key sizes of 128, 192 and 256 bits, respectively. The number of rounds T is 10, 12, and 14 respectively. The AES uses the four operations `SubBytes`, `ShiftRows`, `MixColumns`, and `AddRoundKey` which are detailed below. We use R_i, $1 \leq i \leq T$, to denote the round function which takes a 128-bit block as input and provides a 128-bit block as output. The ith round is defined as

$$R_i = \begin{cases} \texttt{AddRoundKey}_i \circ \texttt{MixColumns} \circ \texttt{ShiftRows} \circ \texttt{SubBytes} & , \ i < T \\ \texttt{AddRoundKey}_i \circ \texttt{ShiftRows} \circ \texttt{SubBytes} & , \ i = T \end{cases}.$$

Before the first round, a pre-whitening key is used in a step $\texttt{AddRoundKey}_0$, so the T-round encryption with master key K is denoted as

$$E_K = R_T \circ \cdots \circ R_1 \circ \texttt{AddRoundKey}_0.$$

Each of the four operations operate on a 128-bit block arranged in a 4×4 byte matrix:

$$\begin{pmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_1 & s_5 & s_9 & s_{13} \\ s_2 & s_6 & s_{10} & s_{14} \\ s_3 & s_7 & s_{11} & s_{15} \end{pmatrix}.$$

The bytes are regarded as elements of what is called the *Rijndael finite field* $\mathbb{F}_{256} = \mathbb{F}_2[x]/(x^8 + x^4 + x^3 + x + 1)$. In the Rijndael finite field, an element is represented by a single byte $a = (a_7 a_6 \cdots a_1 a_0)$ with $a_i \in \mathbb{F}_2$, which in turn represents the field element

$$a(x) = a_7 x^7 + a_6 x^6 + \cdots + a_1 x + a_0.$$

We use hexadecimal notation in typewriter font to write byte values. As such $a = \texttt{01}$ represents $a(x) = 1$, $a = \texttt{02}$ represents $a(x) = x$, and so on. In the following, we briefly describe the four operations used in AES.

2.1 SubBytes

In the `SubBytes` operation, each of the 16 bytes in the state matrix is replaced by another value according to an 8-bit S-box. In the standard AES, the AES S-box is used whose full description is available to the adversary. However, in our analysis we will assume that the S-box is secret and thus unknown to the adversary.

2.2 ShiftRows

In the `ShiftRows` step, the ith row of the state, $0 \leq i \leq 3$, is rotated to the left by i positions. As such,

$$\texttt{ShiftRows}\left(\begin{pmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_1 & s_5 & s_9 & s_{13} \\ s_2 & s_6 & s_{10} & s_{14} \\ s_3 & s_7 & s_{11} & s_{15} \end{pmatrix}\right) = \left(\begin{pmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_5 & s_9 & s_{13} & s_1 \\ s_{10} & s_{14} & s_2 & s_6 \\ s_{15} & s_3 & s_7 & s_{11} \end{pmatrix}\right) .$$

2.3 MixColumns

In this step, each of the four columns of the state matrix are multiplied from the right onto an invertible matrix M over the Rijndael finite field. The matrix M and its inverse are

$$M = \begin{pmatrix} \texttt{02} & \texttt{03} & \texttt{01} & \texttt{01} \\ \texttt{01} & \texttt{02} & \texttt{03} & \texttt{01} \\ \texttt{01} & \texttt{01} & \texttt{02} & \texttt{03} \\ \texttt{03} & \texttt{01} & \texttt{01} & \texttt{02} \end{pmatrix} \quad \text{and} \quad M^{-1} = \begin{pmatrix} \texttt{0e} & \texttt{0b} & \texttt{0d} & \texttt{09} \\ \texttt{09} & \texttt{0e} & \texttt{0b} & \texttt{0d} \\ \texttt{0d} & \texttt{09} & \texttt{0e} & \texttt{0b} \\ \texttt{0b} & \texttt{0d} & \texttt{09} & \texttt{0e} \end{pmatrix}.$$

2.4 AddRoundKey

In this step, a 128-bit round key is added to the state using the XOR operation. The $T + 1$ round keys, denoted $RK_0, \ldots, RK_T$ are generated using the AES key schedule. A brief description of the AES key schedule can be found in Appendix A.

3 Cryptanalysis of the AES with a Secret S-box

3.1 Differential and Linear Cryptanalysis

First, we consider the security of the AES with a secret S-box which is chosen uniformly at random against the two most commonly used attacks vectors for block ciphers: differential cryptanalysis and linear cryptanalysis. The original AES was designed to resist these two attacks.

It has been shown that for mappings chosen uniformly at random from the set of all m-bit bijective mappings, the expected value of the highest probability of a (non-trivial) differential characteristic is at most $\frac{2m}{2^m}$ [11]. In our case where $m = 8$, this means that for a randomly chosen 8-bit S-box the expected maximum probability of a differential characteristic is $\frac{16}{2^8} = 2^{-4}$.

Since the number of active S-boxes for four rounds of the AES is at least 25 [4], one has an upper bound of the probability for any 4-round differential characteristic of 2^{-100}, and thus an upper bound for any 8-round differential characteristic of 2^{-200}. This is sufficient to conclude that differential cryptanalysis will not pose a threat to variants of the AES where the S-box is replaced by a randomly chosen 8-bit S-box.

It is possible to prove a similar result for linear cryptanalysis using the bounds of linear characteristics from [12].

3.2 Integral Cryptanalysis on Four Rounds

Summary. Before we go into the details of the attack, let us summarize it shortly. The attack splits the task of determining the secret S-box into consecutive steps that find increasingly better.

First we use the fact that we can create balanced sets of intermediate texts right after the first `SubBytes` step in round 1 by applying the SQUARE attack as a chosen ciphertext attack[1]. These balanced sets can be used to set up a system of linear equations which can be used to determine the secret S-box up to affine equivalence over $\mathbb{F}_2^8$ as is similarly done in the SASAS attack [1]. A representative from this equivalence class is already sufficient to determine the whitening key up to 256 variants.

The knowledge about the whitening key and the representative of the S-box equivalence class allow us now to determine the intermediate texts right before the `MixColumns` step in round 1 up to affine equivalence over $\mathbb{F}_2^8$. As a result of the SQUARE attack, the intermediate texts *after* the `MixColumns` step should take on each byte value in each byte position exactly once. This can be used to determine the secret S-box up to affine equivalence over $\mathbb{F}_{256}$. Finally, the secret S-box can be determined using knowledge of the key schedule.

[1] The reason for using a chosen ciphertext instead of a chosen plaintext attack will be explained later.

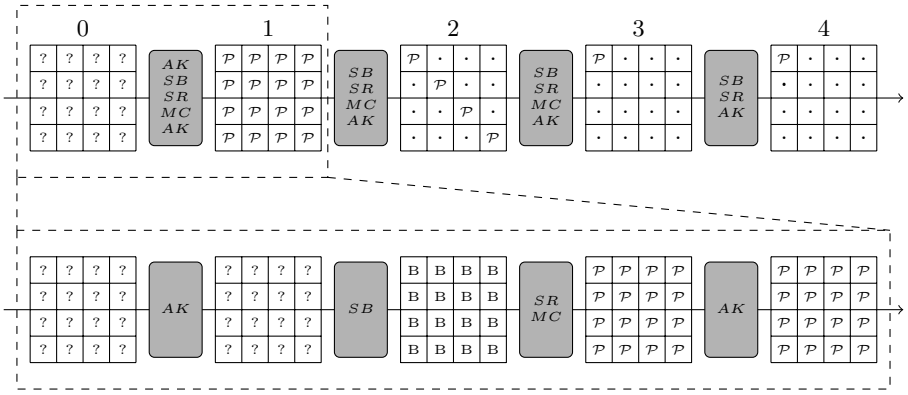

Fig. 1. Outline of the 4-round integral attack. The following notation is used: $\mathcal{P}$ takes each of the 256 values once, $\cdot$ is constant, B is balanced and the values ? are unknown.

Prerequisites. Before we start with the attack, let us clarify the notation. We assume that the last round, the fourth in this case, does not contain a `MixColumns` operation, as is the case for the last round of standard AES.

By a Λ-set, we mean a set of 256 messages that differ only in one byte but take for this byte all possible 256 values. Just as in the standard SQUARE [3] attack, when we decrypt a Λ-set with 4-round AES, we get intermediate texts right after the `SubBytes` step of round 1 that are balanced, i.e. the sum of all texts is equal to the text containing only zeroes, In particular, this set of messages is balanced in every byte.

Finding an Affine Equivalent of the Secret S-box over $\mathbb{F}_2^8$. Let $p_i, 0 \leq i < 256$, be the list of the first bytes of the 256 plaintexts, generated from the Λ-set of ciphertexts. Let k_0 be the first byte of the whitening key. We can now write the fact that the intermediate texts are balanced right after the first `SubBytes` step as

$$\bigoplus_{i=0}^{255} S(p_i \oplus k_0) = 0$$

where S is the secret S-box. Let $z_i := S(k_0 \oplus i)$. The above equation is then linear in the z_{p_i} and can be written as

$$z_{p_0} \oplus z_{p_1} \oplus \cdots \oplus z_{p_{255}} = 0. \tag{1}$$

As duplicate values in the p_i values will cancel out, only those p_i need to be taken into account that appear an odd number of times in the list.

Taking different Λ-sets of ciphertexts, we can now try to generate enough linear equations to be able to determine S uniquely. Unfortunately, we encounter two problems now. Firstly, we do not know the value of k_0. We can thus only

hope to determine $S(k_0 \oplus \cdot)$. Secondly, the above equations are invariant under affine transformations: Let A be an affine transformation from $\mathbb{F}_2^8$ to $\mathbb{F}_2^8$. Then

$$A(z_{p_0}) \oplus A(z_{p_1}) \oplus \cdots \oplus A(z_{c_{255}}) = 0$$

is also true for any set of p_i that fulfills equation (1) and has an even number of summands. We can thus at best determine $S(k_0 \oplus \cdot)$ up to 2^{72} affine equivalent variants. Using the fact that the affine mapping needs to be invertible, we can thus at best determine the set

$$\{A \circ S(k_0 \oplus \cdot) \mid A : \mathbb{F}_2^8 \to \mathbb{F}_2^8 \text{ is invertible}\}$$

which is of size $2^{70.2}$.

As each linear equation like equation 1 gives us one byte of information and as we can only determine the S-box up to $2^{72} = 2^{9 \cdot 8}$ variants, there can at most be $256 - 9 = 247$ linearly independent equations like equation (1). We found that using 256 different Λ-sets suffices in most cases to generate a set of equations with rank 247.

Given such a set of equations, it is now easy to determine one representative from the set of affine equivalents to $S(k_0 \oplus \cdot)$. Let this representative be denoted as S', i.e. $S' = A \circ S(k_0 \oplus \cdot)$ for some invertible affine $A : \mathbb{F}_2^8 \to \mathbb{F}_2^8$ and unknown k_0.

Determining the Whitening Key. Let now $p_{i,j}$ with $0 \leq i < 256$ and $0 \leq j < 8$ be byte j of the plaintext i in one of the Λ-sets and let k_j be byte j of the whitening key. We then have for $a \in \mathbb{F}_2^8$:

$$a = k_j \quad \Rightarrow \quad 0 = \bigoplus_{i=0}^{255} S(a \oplus p_{i,j}),$$

which is generally not true for $a \neq k_j$, a fact the standard SQUARE attack is based on as well. For invertible affine $A : \mathbb{F}_2^8 \to \mathbb{F}_2^8$, we also have the equivalence

$$0 = \bigoplus_{i=0}^{255} S(a \oplus p_{i,j}) \quad \Leftrightarrow \quad 0 = \bigoplus_{i=0}^{255} A \circ S(a \oplus p_{i,j}).$$

We can thus for each byte j with $1 \leq j < 8$ find $k_j \oplus k_0$ by trying out for which of the 256 possible values of a we have

$$\bigoplus_{i=0}^{255} S'(a \oplus p_{i,j}) = 0$$

for all Λ-sets. This allows us to determine the whitening key up to 256 variants, depending on the value of k_0. Let us set $k' = (0, k_1 \oplus k_0, k_2 \oplus k_0, \ldots, k_{15} \oplus k_0)$. Then when using k' as the whitening key and S' as the S-box for encryption, the intermediate texts after the `ShiftRows` step in round 1 will correspond to the correct intermediate texts up to a fixed affine transformation on each byte.

Finding an Affine Equivalent of the Secret S-box over $\mathbb{F}_{256}$. When we decrypt a Λ-set, the set of intermediate texts that we get after the `MixColumns` step in round 1 will take all 256 possible values in each of the 16 state bytes (see Figure 1). The key idea here is to use this property to filter out wrong candidates for the secret S-box.

For a set of 256 bytes, we say that it has the $\mathcal{P}$ property if it contains every possible value exactly once. Let V be a set of 256 byte vectors. We will likewise say that V has the $\mathcal{P}$ property if V has this property in every byte position.

If V is now the set of intermediate texts after the `MixColumns` operation in round 1, that is the result of the decryption of a Λ-set, we know from the SQUARE attack that V has the $\mathcal{P}$ property. Let now D be the corresponding set of intermediate texts directly before the `MixColumns` step. We can test our candidate S' for S, by constructing the corresponding candidate set D' for the intermediate texts after the `ShiftRows` step in round 1 with our acquired knowledge of the whitening key, and applying the `MixColumns` operation on this set D' to see whether we obtain a set with the $\mathcal{P}$ property.

For how many of the 2^{72} candidates for S' do we expect this to hold? Let A be the affine transformation by which S' deviates from S. Then the byte vectors in D' also deviate by this transformation from the true set D. Clearly, if A consists only of an addition, the $\mathcal{P}$ property of MD' is preserved where M is the `MixColumns` matrix. We can thus restrict A to linear transformations.

In the case, that A corresponds to an invertible linear mapping over $\mathbb{F}_{256}$, i.e. a multiplication with some element from $\mathbb{F}_{256}^*$, the set of intermediate texts after the `MixColumns` step will still have the $\mathcal{P}$ property as well since the linear transformation commutes with the multiplication within the `MixColumns` matrix M and the application of the invertible linear transformation A on the set MD leaves the $\mathcal{P}$ property untouched:

$$MD' = MAD = AMD.$$

Opposed to this, when A does not commute with the multiplication in $\mathbb{F}_{256}$, the $\mathcal{P}$ property of MD' is in general not preserved. As is shown in Appendix B, if A commutes with a primitive element of $\mathbb{F}_{256}$, A corresponds to multiplication with an element of $\mathbb{F}_{256}$. As `03` is a primitive element of the Rijndael field and is an entry in every row and column of M, the only class of affine transformations that preserve the $\mathcal{P}$ property of MD' is exactly the affine transformations over $\mathbb{F}_{256}$.

Checking whether the $\mathcal{P}$ property holds for MD' allows us thus to find the correct S up to affine transformations over $\mathbb{F}_{256}$. Nevertheless, still $2^{72-16} = 2^{56}$ candidates need to be tested.

Complexity Reduction: Finding the Affine Equivalent Over $\mathbb{F}_{256}$. The specific structure of the `MixColumns` matrix M allows us to reduce the computational complexity of finding the correct affine representative amongst the 2^{56} possible candidates.

Let us define that a set of $2l$ vectors over $\mathbb{F}_2^n$ has the $\mathcal{R}$ property if both 1 and 0 appear in every bit position exactly l times. Note that the $\mathcal{P}$ property implies the $\mathcal{R}$ property and that the $\mathcal{R}$ property implies that the set of vectors is balanced but the opposite direction of implications is in generally false. As the $\mathcal{R}$ property, like the $\mathcal{P}$ property, is not preserved by the `MixColumns` layer, we still expect to find the correct representative by testing for the $\mathcal{R}$ property instead of the $\mathcal{P}$ property[2].

Let us take a closer look at the specific form of matrix M. When written as a linear function from F_{256}^4 to F_{256}^4, it has the form

$$M = \begin{pmatrix} \texttt{02 03 01 01} \\ \texttt{01 02 03 01} \\ \texttt{01 01 02 03} \\ \texttt{03 01 01 02} \end{pmatrix}. \tag{2}$$

If we associate the multiplication with `01`, `02`, and `03` with their respective linear mappings from $\mathbb{F}_2^8$ to $\mathbb{F}_2^8$, we get the following representations:

$$\texttt{01} = \begin{pmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \end{pmatrix} \texttt{02} = \begin{pmatrix} 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 1&0&0&0&1&0&0&0 \\ 1&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 1&0&0&0&0&0&0&1 \\ 1&0&0&0&0&0&0&0 \end{pmatrix} \texttt{03} = \begin{pmatrix} 1&1&0&0&0&0&0&0 \\ 0&1&1&0&0&0&0&0 \\ 0&0&1&1&0&0&0&0 \\ 1&0&0&1&1&0&0&0 \\ 1&0&0&0&1&1&0&0 \\ 0&0&0&0&0&1&1&0 \\ 1&0&0&0&0&0&1&1 \\ 1&0&0&0&0&0&0&1 \end{pmatrix}. \tag{3}$$

If we now write $a_0, a_1, \ldots, a_7$ for the rows of A we can write the first row of the 32×32 matrix MA over $\mathbb{F}_2$ as

$$v := (a_1, \quad a_0 \oplus a_1, \quad a_0, \quad a_0).$$

We see now that whether or not the first bit in the set MD' satisfies the $\mathcal{R}$ property relies solely on the rows a_0 and a_1 of the matrix A. As we only need to test matrices A that are not linearly equivalent over $\mathbb{F}_{256}$, we can fix one row of A to a non-zero constant. Let a_0 be fixed. Then we only need to try out all 2^8 possible values for a_2 to see which one gives us the $\mathcal{R}$ property in this bit.

After having determined a_1 (and fixed a_0), we can use the second row of MA to determine a_2 and continue on to determine A uniquely. In each step, we only need to test 2^8 possible values. We can thus split the task of trying of out all 2^{56} candidates for A, to trying out row by row which reduces the complexity to $7 \cdot 2^8 \approx 2^{11}$ steps.

Determining the Secret S-box. Without assuming anything about the key schedule, we can only determine the secret S-box up to an additive constant before and after the S-box, i.e. $S'(x) \sim a \oplus S(b \oplus x)$ since any additive constants can also be seen as part of the round keys. When not assuming anything about the key schedule, one can for example require that the first byte of the whitening key and the first round key is zero. It is straightforward then to find the correct

[2] This was indeed the case for all our test runs.

representative for S out of the 2^{16} options under these constraints. Using knowledge about the key schedule, one can also easily determine the correct variants for the round keys and adjust the representative for the S-box accordingly.

The Complexity of the Attack. The needed data consists of the decryption of 256 Λ-sets which corresponds to a data complexity of 2^{16} chosen ciphertexts. As most of these texts are only used to generate the linear system of equations in the first plaintext byte, most plaintext pairs can be discarded after the corresponding equation has been extracted. The memory complexity is thus $2^{8+8} = 2^{16}$ bytes.

Let us go through the steps to see what the time complexity is. Determining S' up to affine equivalence over $\mathbb{F}_2^8$ requires solving a system of linear equations in 2^8 variables. This requires $2^{3 \cdot 8} = 2^{24}$ steps where each step is comparable to a table lookup. Finding the whitening key requires trying out for each of the 16 key bytes all 2^8 possible solutions with one Λ-set of 2^8 values. It thus takes about $16 \cdot 2^8 \cdot 2^8 = 2^{20}$ table lookups.

To determining S' up to affine equivalence over $\mathbb{F}_{256}$ using the $\mathcal{R}$ property, for each of the seven rows of A that have not been fixed we have to test 2^8 values, each with a Λ sets. Thus the total complexity of this step is $7 \cdot 2^8 \cdot 2^8 \approx 2^{19}$. A step here has about the same complexity as a table lookup.

The complexity of the attack is dominated by solving the linear system of equations, namely 2^{24} steps, which corresponds to 2^{17} encryptions when assuming a complexity of 2^5 table lookups per encryption round. We ran the attack 1000 times on the single core of an Intel Core i7-4600M CPU at 2.90GHz. It found both the correct S-box and the correct key each time and always ran in less than a second (including reading the input data).

3.3 Integral Cryptanalysis on Five Rounds

The attack on four rounds can be extended to five rounds using a technique by Ferguson et al. [6] that allowed to improve the SQUARE attack on six rounds. The underlying idea is to create sets of ciphertexts that form a Λ-set right before the `MixColumns` step of round 4. Unfortunately, even with key guessing, it is not possible to determine such a set without knowledge of the secret S-box. However, by taking all 2^{32} possible values for four bytes that are in the same column during the `MixColumns` step of round 4 and keeping all other bytes constant, we can generate a set of ciphertexts that will take all 2^{32} values in that column. This set can now be viewed as the union of 2^{24} Λ-sets (see Figure 2).

A set of ciphertexts that gives us a Λ-set in the `MixColumns` step of round 4 will generate a balanced set right after the `SubBytes` step of round 1. As a sum of balanced sets remains balanced, decrypting our 2^{32} ciphertexts, we get a balanced set of size 2^{32} after the `SubBytes` step of round 1. This set can now be used to mount the four round attack on five rounds as well.

Just as in the four round version, we use the fact when such a set is balanced, we can, by using 256 of them, create a system of linear equations that can be solved to find an S-box S' that is an affine equivalent to S over $\mathbb{F}_2^8$. We can use

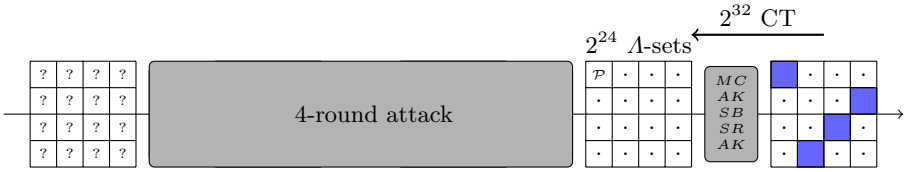

Fig. 2. The 2^{32} ciphertexts take all possible combinations in the blue bytes but constant values in the rest. The state before the `MixColumns` step in the round before can be seen as the union of 2^{24} Λ-sets as depicted here. It is then possible to apply the 4-round attack again.

the knowledge of S' again to determine the whitening key up to 256 variants. We can then again generate the corresponding intermediate texts after the first `SubBytes` step that are affinely equivalent over $\mathbb{F}_2^8$ to the true texts. With these texts we can now determine S up to affine equivalence over $\mathbb{F}_{256}$ by using the $\mathcal{R}$ property. Note that when using the $\mathcal{R}$ property here, we expect the correct set of texts to take in each bit the values 0 and 1 each exactly 2^{31} times as we are now working with the union of 2^{24} Λ-sets. Again to determine S exactly and finding the correct master key is straightforward from this point.

How do the complexities of the attack change as compared to the 4-round variant? As we need 256 sets of ciphertexts, each of size 2^{32}, this leaves us with a data complexity of 2^{40}, an increase by a factor of 2^{24} in comparison to the four round attack. The time complexity of solving the linear system of equations does not change (it is still a system of 256 equations in 256 variables). The complexity of the whitening key recovery increases with the size of the balanced sets, i.e. again by a factor of 2^{24}, leaving us with a complexity of 2^{44} table lookups. Likewise is the complexity of checking the $\mathcal{R}$ property increased by a factor of 2^{24} to a total complexity of 2^{43} steps of the same complexity as a table lookup. This leaves the total time complexity at roughly 2^{45} steps which corresponds to 2^{38} encryptions when assuming a complexity of 2^5 table lookups per encryption round.

The data complexity of 2^{40} chosen ciphertexts corresponds to 18 terabyte of data. But as most of the sets of 2^{32} plaintexts are each only used to generate one linear equation (in the 256 variables), apart from a few (16 suffice), most can be discarded during the generation of the linear system of equations, leaving us with at most 2^{40} bytes that need to be stored in memory at any point in time.

3.4 Integral Cryptanalysis on Six Rounds

The standard way of extending the Square attack to six rounds (in the case of a chosen ciphertext attack) is by guessing four bytes of the whitening key and peeling of the first round of encryption for one byte of intermediate text, thereby increasing the time complexity of the attack by a factor of 2^{32}. Unfortunately, this does not extend to the AES with a secret S-box as knowledge of the S-box is required to strip off the first round.

There is nonetheless a way to extend the five round attack to six rounds. Over one round of the AES, the four output bytes of one column only depend on four of the input bytes. Thus, it is possible to describe two rounds of AES with a secret S-box as the parallel application of four Super-boxes (see also [5]) with a linear transformation before and after. Such a Super-box consists of the parallel application of four S-boxes, a key addition a multiplication of the four bytes with the `MixColumns` matrix, again an application of four S-boxes in parallel and a final key addition.

Just as in the 5-round attack, we can generate sets of texts that are balanced right after the `SubBytes` step in round 2 and we can hence use these texts to generate a system of linear equations that lets us determine the Super-boxes, just as it allowed us to determine the usual S-boxes in the attacks before. Unfortunately, the system of linear equations for one Super-box involves now not 2^8 variables but 2^{32} variables. This means that both the computational complexity as well as the data complexity increase. For the data complexity, when using the round extension as in the five round attack, we need now 2^{32} sets of each 2^{32} texts, leaving us with a data complexity of 2^{64} chosen ciphertexts. Just as with the attack on the normally sized S-box, the set of equations is not of full rank and lets us determine the Super-box only up to $2^{32 \cdot 32+32} = 2^{1056}$ affine equivalents – only slightly less when taking the necessary bijectivity of the affine transform into account.

The Super-box that we obtain will thus be of the form

$$A \circ \texttt{SubBytes} \circ \texttt{KeyAddition} \circ \texttt{MixColumns} \circ \texttt{SubBytes} \circ \texttt{KeyAddition}$$

where A is an unknown invertible affine mapping over $\mathbb{F}_2^{32}$ and where the other standard AES steps are truncated to operate on four bytes only. Despite our lack of knowledge of A, this form is already enough to extract from it the secret S-box and the involved key bytes up to 2^{16} variants, i.e. up to two additive constants applied before and after the S-box. After this, it is straightforward to uniquely determine the secret S-box and the key e.g. by guessing the two additive constants and applying standard 6-round SQUARE attack.

If we decrypt a Λ-set with our affinely transformed Super-box, we get a set that is balanced right after the first `SubBytes` step of the Super-box as described in the SASAS paper [1]. Note that it is necessary to assume that A distributes the 8 bits that are being varied in the Λ-set to at least two S-Boxes, an assumption that is true for almost all possible A. At this point we can thus simply apply again the same techniques as we did for the four round attack to determine the secret S-box and the involved key bytes, only that we mount the attack on the affine equivalent of the Super-box now instead of the whole cipher.

What is the complexity of this attack? As already mentioned above, the data complexity is 2^{64} chosen ciphertexts. The time complexity is dominated by the first step of solving the system of 2^{32} linear equations over 2^{32} variables. Using Gaussian elimination, this step consists of 2^{96} operations, each comparable in complexity to a table lookup. Thus, the time complexity corresponds to 2^{90} encryptions when assuming a complexity of 2^5 table lookups per encryption

round. The memory complexity of $2^{32} \cdot 2^{32} \cdot 32 = 2^{69}$ bytes is also dominated by the size of the system of equations.

3.5 A Note on Chosen Ciphertext vs. Chosen Plaintext

Due to the symmetry of the AES regarding encryption and decryption, the attacks described here principally work in both directions. Interestingly though, for the attacks on four and five rounds, the chosen ciphertext variant is considerably more effective than the chosen plaintext attack. This is because the `MixColumns` matrix is sufficiently sparser than its inverse, creating a difference of 2^{16} in the number of steps when applying the $\mathcal{R}$ property. This changes the time complexities of the 4-round and 5-round attacks to 2^{28} and 2^{54}. As the complexity of the 6-round attack is dominated by the solving of the linear system of equations, it does not make a difference in that attack scenario.

4 Conclusion

In this work, we studied the impact of replacing the S-box in the AES by a secret S-box unknown to the adversary. Despite the expected increase in difficulty of recovering the secret information, we were able to mount efficient attacks based on integral cryptanalysis combined with dedicated techniques.

We were able to show that AES-128 with a secret S-box, reduced to 4 and 5 rounds, is susceptible to attacks with practical complexity that successfully recover both the secret S-box and the key. Furthermore, we have shown an attack on a variant with 6 rounds with a time complexity of 2^{90}, which is much less effort than the time required to do exhaustive search of the key, let alone of the S-box.

Similarly to standard AES, it seems difficult to extend our attacks to more than 6 rounds. Also, the gap between the time complexities of integral attacks on standard AES and the AES with a secret S-box increases dramatically for the attack on 6 rounds. It is an open question whether this complexity can be further reduced.

Acknowledgements

The work in this paper has partially been funded by the Nasjonal sikkerhetsmyndighet (NSM).

References

1. A. Biryukov and A. Shamir. Structural cryptanalysis of SASAS. In B. Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 394–405, 2001.
2. J. Borghoff, L. R. Knudsen, G. Leander, and S. S. Thomsen. Cryptanalysis of PRESENT-like ciphers with secret S-boxes. In A. Joux, editor, *Fast Software Encryption, FSE 2011*, volume 6733 of *LNCS*, pages 270–289, 2011.

3. J. Daemen, L. R. Knudsen, and V. Rijmen. The block cipher Square. In E. Biham, editor, *Fast Software Encryption, FSE '97*, volume 1267 of *LNCS*, pages 149–165. Springer, 1997.
4. J. Daemen and V. Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.
5. J. Daemen and V. Rijmen. Understanding two-round differentials in AES. In R. D. Prisco and M. Yung, editors, *Security and Cryptography for Networks (SCN) 2006*, volume 4116 of *LNCS*, pages 78–94, 2006.
6. N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner, and D. Whiting. Improved cryptanalysis of Rijndael. In B. Schneier, editor, *Fast Software Encryption, FSE 2000*, volume 1978 of *LNCS*, pages 213–230, 2000.
7. M. Gomathisankaran and R. B. Lee. Maya: A novel block encryption function. In *International Workshop on Coding and Cryptography 2009*, 2009.
8. G.-Q. Liu, C.-H. Jin, and C.-D. Qi. Improved slender-set linear cryptanalysis. In *Fast Software Encryption, FSE 2014*, 2014.
9. R. C. Merkle. Fast software encryption functions. In A. Menezes and S. A. Vanstone, editors, *CRYPTO '90*, volume 537 of *LNCS*, pages 476–501, 1991.
10. National Institute of Standards and Technology. Advanced Encryption Standard. Federal Information Processing Standard (FIPS), Publication 197, U.S. Department of Commerce, Washington D.C., November 2001.
11. L. O'Connor. On the Distribution of Characteristics in Bijective Mappings. In T. Helleseth, editor, *EUROCRYPT '93*, volume 765 of *LNCS*, pages 360–370. Springer, 1994.
12. L. O'Connor. Properties of linear approximation tables. In B. Preneel, editor, *Fast Software Encryption, FSE '94*, volume 1008 of *LNCS*, pages 131–136. Springer, 1995.
13. M. Rivain and T. Roche. SCARE of secret ciphers with SPN structures. In K. Sako and P. Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013*, volume 8269 of *LNCS*, pages 526–544, 2013.
14. B. Schneier. Description of a new variable-length key, 64-bit block cipher (Blowfish). In R. J. Anderson, editor, *Fast Software Encryption, FSE '93*, volume 809, pages 191–204, 1994.
15. B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson. Twofish: A 128-Bit Block Cipher.

A The AES Key Schedule

In the AES, we think of the round keys as matrices over the Rijndael finite field, just as the state matrix. The first pre-whitening key RK_0 is the n-bit master key itself, so $RK_0 = K$. The key schedule varies slightly across the three AES variants. Here, we describe it for AES-128 and refer to [4] for the other two cases. We consider the 4 columns of the two round keys as $RK_i = (RK_i^0 \| RK_i^1 \| RK_i^2 \| RK_i^3)$ and $RK_{i+1} = (RK_{i+1}^0 \| RK_{i+1}^1 \| RK_{i+1}^2 \| RK_{i+1}^3)$. To derive RK_{i+1} from RK_i, $0 \leq i < T$, we do the following

1. Let $RK_{i+1}^j = RK_i^j$ for $j = 0, 1, 2, 3$,
2. Rotate RK_{i+1}^3 such that the byte in the first row is moved to the bottom,
3. Substitute each byte in RK_{i+1}^3 by using the S-box from the `SubBytes` operation,

4. Update the byte in the first row of RK^3_{i+1} by adding $\mathtt{02}^{i-1}$ from the Rijndael finite field, and
5. Let $RK^j_{i+1} = RK^j_{i+1} \oplus RK^{j-1}_{i+1} \mod 4$ for $j = 0, 1, 2, 3$.

This procedure is repeated for $i = 1, \ldots, T$ to obtain the round keys RK_0 to RK_T.

B Lemma

Let $m \in \mathbb{N}^*$. As $\mathbb{F}_{2^m}$ is an m-dimensional $\mathbb{F}_2$-vector space, its elements can be represented as m-dimensional $\mathbb{F}_2$-vectors. But as the multiplication in $\mathbb{F}_{2^m}$ obeys the distributive law, the multiplication with an element of $\mathbb{F}_{2^m}$ corresponds to a linear mapping from $\mathbb{F}_2^m$ to $\mathbb{F}_2^m$, that is an $m \times m$ matrix over $\mathbb{F}_2$. For an element $a \in \mathbb{F}_{2^m}$, let L_a denote the corresponding $m \times m$ matrix. For $b \in \mathbb{F}_{2^m}$, we then have $a \cdot b = L_a b$.

Lemma 1. *Let a be a primitive element of $\mathbb{F}_{2^m}$. Let B be an $m \times m$ matrix over $\mathbb{F}_2$ which commutes with L_a. Then there exists $b \in \mathbb{F}_{2^m}$ such that $L_b = B$.*

Proof. Let c be any element from $\mathbb{F}^*_{2^m}$. As a is primitive, there exists $k \in \mathbb{N}^*$ such that $c = a^k$ and likewise $L_c = L_a^k$. As B commutes with L_a, by induction B also commutes with L_c. Clearly, B also commutes with L_0, so B commutes with all elements of $\mathbb{F}_{2^m}$.

Let now $b \in \mathbb{F}_{2^m}$ be the image of 1 under B, $b = B1$. We then have for any $c \in \mathbb{F}^*_{2^m}$:

$$Bc = L_1 Bc = L_c L_{c^{-1}} Bc = L_c B L_{c^{-1}} c = L_c B1 = L_c b = c \cdot b = b \cdot c = L_b c.$$

As this is true for any $c \in \mathbb{F}^*_{2^m}$ and clearly also for 0, we have $B = L_b$. □