On some Weak Extensions of AES and BES

Jean Monnerat * and Serge Vaudenay

EPFL, Switzerland
http://lasecwww.epfl.ch

Abstract. In 2002, Murphy and Robshaw introduced an extension BES of AES and argued this could compromise the security of AES. We introduce here two block-ciphers CES and Big-BES that are some extensions of the AES and BES respectively in the spirit of Hensel lifting extensions. They are defined similarly to the AES respectively BES except that every operations are performed in a ring structure including the field $GF(2^8)$. We show that the AES and BES can be embedded in their extensions. More precisely, by restricting these extensions on a given subset, we obtain a fully equivalent description of the AES and BES. Furthermore, we show that these natural extensions are trivially weak by describing a cryptanalysis of them despite it leads to no consequence about the security of AES or BES. This shows that (except the nice mathematical construction) the Murphy-Robshaw extension might be pointless.

Key words: AES, BES, Rijndael.

1 Introduction

Since the publication of the Advanced Encryption Standard (AES), many attempts have been performed to find some security flaws in its design. It is well-known that the AES resists to some classical attacks such as the linear cryptanalysis [9] and the differential cryptanalysis [3]. Recently, the attention of some researchers has focused on some new ideas based on algebraic concepts [1, 2, 5, 7, 11]. This is particularly due to the fact that the S-box of the AES is algebraic.

The block-cipher BES has been proposed by Sean Murphy and Matthew J.B. Robshaw in Crypto 2002 [11]. BES has a 128 bytes message space and key space and can be regarded as an extension of the 128 bits version of the AES. Namely, by restricting BES on a special subset of the state space, we obtain a cipher that is fully equivalent to the AES. One of the advantages of BES is to describe the AES with basic operations in $GF(2^8)$.

Recently, people investigated the implication of an other extension type (the Hensel lifting) in public key cryptography (Okamoto-Uchimaya [13], Catalano et al. [4]). It was demonstrated that some extensions are indeed weak. Similar ideas were used in order to solve the Discrete Logarithm Problem on the elliptic

* Supported in part by a grant of the Swiss National Science Foundation, 200021-101453/1.

curves of trace one, see Smart [15] and Satoh [14]. Here we demonstrate that symmetric-key cryptography have similar properties.

In this paper, we introduce an extension of BES called Big-BES by replacing the underlying field $GF(2^8)$ by a commutative ring R in which $GF(2^8)$ can be embedded in a very natural way. We show also that restricting Big-BES on some subsets of the state vector space provides a cipher fully equivalent to BES. Using some linear properties occurring in Big-BES, we can cryptanalyze it quite efficiently. We also apply a similar construction to the AES by defining the block-cipher CES and show that CES can be broken by an identical cryptanalysis. From this work, we can deduce first that replacing the underlying field by a similar structure can have a destructive impact on the security of a block-cipher even if this one is an extension. Secondly, despite of an efficient cryptanalysis, this leads to no consequence about the security of BES and AES. So, it seems that a natural extension of AES such as BES may be weak without compromising the security of AES. This comes from the strong properties required to the extensions of a given cipher e.g. AES.

In section 2 we recall the descriptions of AES and BES. Section 3 is devoted to the introduction of Big-BES that we cryptanalyze in section 4. We adapt this work to AES in section 5 by defining a new block-cipher called CES. Then, section 6 contains a discussion of the choice of the extension of the field $GF(2^8)$. Finally, section 7 concludes this article.

2 Background on AES and BES

The operations involved in AES and BES are essentially performed in the finite field $\mathbb{F} = GF(2^8) = GF(2)[X]/(p(X))$, where p is the following irreducible polynomial $p(X) = X^8+X^4+X^3+X+1$. A byte will be then considered as an element of $\mathbb{F}$ and a plaintext of AES resp. BES will be an element of $\mathbb{F}^{16}$ resp. $\mathbb{F}^{128}$.

Inversion. In the two ciphers, we use an inversion map that is defined as the normal inversion in $\mathbb{F}$ for non-zero elements and that maps zero to itself, i.e. for $a \in \mathbb{F}$ we have

$$a^{(-1)} := a^{254} = \begin{cases} a^{-1} & \text{if } a \neq 0 \\ 0 & \text{if } a = 0. \end{cases}$$

For a vector $\mathbf{b}$ of a space $\mathbb{F}^n$, $\mathbf{b}^{(-1)}$ means a componentwise inversion, i.e.

$$\mathbf{b}^{(-1)} := (b_1^{(-1)}, \ldots, b_n^{(-1)}).$$

2.1 The AES Structure

In this subsection, we recall roughly the structure of the AES and we will provide a description in which all operations are performed in the field $\mathbb{F}$. We consider here the 128 bits version with 10 rounds and we omit the key schedule. For a detailed version of the AES, we refer the reader to FIPS-197 [12] and the book of Daemen and Rijmen [6].

AES is a block cipher that consists in an iteration of some round transformations on the plaintext. The plaintext, the subkeys and the ciphertext are some elements of the state space $A := \mathbb{F}^{16}$ of the AES. Except for an initial subkey addition and the last round, all the rounds are of the following form:

Round of AES. We denote the input of the round as $\mathbf{x} = (x_1, \cdots, x_{16}) \in \mathbb{F}^{16}$. We describe the successive transformations performed in the round below.

1. **Inversion.** $\mathbf{x} \mapsto \mathbf{y} = \mathbf{x}^{(-1)}$ the componentwise inversion in each byte of the state vector.
2. **GF(2)-linear function.** We regard each byte y_i of the intermediate vector $\mathbf{y}$ as a vector on $GF(2)$. Then, we compute $y_i \mapsto L_A \cdot y_i$ for $1 \leq i \leq 16$, where L_A is a fixed 8×8-matrix with $GF(2)$ elements. In [11, 6], it is shown that we can express L_A in $\mathbb{F}$ with the linearized polynomial

$$q(t) = \mathtt{05} \cdot t + \mathtt{09} \cdot t^2 + \mathtt{F9} \cdot t^{2^2} + \mathtt{25} \cdot t^{2^3} + \mathtt{F4} \cdot t^{2^4} + \mathtt{01} \cdot t^{2^5} + \mathtt{B5} \cdot t^{2^6} + \mathtt{8F} \cdot t^{2^7}. \quad (1)$$

3. **ShiftRows and MixColumns.** The vector state is transformed by two functions called `ShiftRows` resp. `MixColumns`. In [11], it is shown that each of these operations corresponds to a matrix multiplication. Indeed, this step consists in the following computation $\mathbf{z} \mapsto \mathrm{Mix}_A \cdot R_A \cdot \mathbf{z}$, where $\mathbf{z} \in \mathbb{F}^{16}$ is the input of this step and Mix_A, R_A are two fixed (16×16) $\mathbb{F}$-matrices. For more details we refer again to [11].
4. **AddRoundKey.** Finally, the last operation of the round is simply a subkey addition, i.e. an addition in $\mathbb{F}^{16}$.

Remark. We notice that the AES S-Box is composed of the step 1 and 2. This transformation performed in the S-Box is called `SubBytes`. We have omitted a constant addition in the S-Box for sake of simplicity because it can be incorporated in a modified key schedule (see [11, 10]).
We also remark that the structure of the AES is composed of simple operations in $\mathbb{F}$ (such as inversions, linear operations) except for the evaluation of the linearized polynomial q. By extending the state space such that the conjugates can be included, the polynomial q can be represented by a matrix on $\mathbb{F}$. Using this fact, S. Murphy and M. Robshaw [11] introduced the cipher BES in order to express the AES in a very simple form.

2.2 The BES Structure

The block-cipher BES is defined on the state space $B := \mathbb{F}^{128}$ and has a similar structure to the AES one. We describe it below.

Round of BES. One round of the BES is essentially an affine transformation except a componentwise inversion. We can describe it as follows (using notation of [11]) :

$$R_i : \mathbf{b} \longrightarrow M_B \cdot \mathbf{b}^{(-1)} + (\mathbf{k}_B)_i,$$

where $\mathbf{b} \in B$ denotes the state vector, $(\mathbf{k}_B)_i \in B$ denotes the i^{th}-subkey and M_B a 128×128-matrix on $\mathbb{F}$.

BES encryption. Let $\mathbf{p} \in B$ be the plaintext, $\mathbf{c} \in B$ the ciphertext, $\mathbf{w}_i \quad (0 \leq i \leq 10)$ the state vector after the i^{th} round and $(\mathbf{k}_B)_i \in B \quad (0 \leq i \leq 10)$ the 11 subkeys. The BES encryption can be described as follows:

$$\begin{aligned} \mathbf{w}_0 &= \mathbf{p} + (\mathbf{k}_B)_0 \\ \mathbf{w}_i &= R_i(\mathbf{w}_{i-1}) \text{ for } i = 1, \ldots, 9 \\ \mathbf{c} = \mathbf{w}_{10} &= M_B^* \cdot \mathbf{w}_9^{(-1)} + (\mathbf{k}_B)_{10}, \end{aligned}$$

where M_B^* is a 128×128-matrix on $\mathbb{F}$. Then, the encryption is composed only of the round described above except for an initial addition key and a different matrix in the last round.

The key motivation of BES is that a given vector space V of dimension 16 (on $\mathbb{F}$) is stable by BES as long as the key lies in a similar vector space and that the cipher BES restricted to V is isomorphic to AES.

3 Big-BES

3.1 Extension of $\mathbb{F}$

Our extension that we define below was inspired by the same way the ring of the p-adic integers extends $\mathbb{Z}/p\mathbb{Z}$. We consider here an analog extension in which we keep only the two first terms of the "$\mathbb{F}$-adic extension". To this end, we simply take $\mathbb{F} \times \mathbb{F}$ such that computations on the first coordinate corresponds to regular $\mathbb{F}$-operations. Hence, we choose the set $R := \mathbb{F} \times \mathbb{F}$ equipped with the componentwise addition

$$(x_1, y_1) + (x_2, y_2) = (x_1 + x_2, y_1 + y_2) \tag{2}$$

and the following multiplication:

$$(x_1, y_1) \cdot (x_2, y_2) = (x_1 \cdot x_2, x_1 \cdot y_2 + x_2 \cdot y_1) \tag{3}$$

From this definition, we notice that $(R, +, \cdot)$ is a commutative ring with the unit element $(1, 0)$ and that the multiplicative inverse of an invertible element $(x, y) \in R$ is given by

$$(x, y)^{-1} = (x^{-1}, yx^{-2}).$$

Thus, no element of the form $(0, y)$ is invertible and R is not a field. Since defining Big-BES requires to modify the inversion in a permutation on R, we use the following inversion map:

$$(x, y)^{(-1)} := \begin{cases} (x^{-1}, yx^{-2}) & \text{if } x \neq 0 \\ (0, y^{(-1)}) & \text{if } x = 0 \end{cases} \tag{4}$$

We notice that every operation performed on the first component of an element of R is the same as in $\mathbb{F}$. Hence, the field $\mathbb{F}$ is embedded in R via the first component. Moreover, R is a $\mathbb{F}$-vector space and its scalar multiplication by $\lambda \in \mathbb{F}$ corresponds to a multiplication in R by $(\lambda, 0)$.

Remark. If we fix the x-component in R and if $x \neq 0$, the inverse operation is linear with respect to the y-component.
Remark. R corresponds to $\mathbb{F}[X]/(X^2)$, i.e. polynomials over $\mathbb{F}$ truncated to the first two terms.

3.2 Definition of the Big-BES

Here we introduce a block cipher called Big-BES that is defined on $\mathfrak{B} := R^{128}$ and that is an extension of BES, i.e. in which BES is embedded. First, we will consider the most general case and then we will look for an appropriate way to embed BES in the Big-BES.

The Big-BES is essentially obtained by replacing the field $\mathbb{F}$ by the ring $R = \mathbb{F} \times \mathbb{F}$ and using the trivial embedding $e : \mathbb{F} \to R$ defined by

$$e(x) = (x, 0). \tag{5}$$

The rounds of Big-BES are defined exactly as for BES except that the operations are performed in R and that we replace the elements of the matrix M_B by their images under the trivial embedding e, i.e.

$$(M_{Big})_{ij} := e((M_B)_{ij}) \text{ for } 1 \leq i, j \leq 128.$$

In a similar way, we define the matrix $M^*_{Big} \in R^{128 \times 128}$. So, the Big-BES is defined as follows:

$$\begin{aligned} \mathbf{w}_0 &= \mathbf{p} + (\mathbf{k}_{Big})_0 \\ \mathbf{w}_i &= M_{Big} \cdot \mathbf{w}_{i-1}^{(-1)} + (\mathbf{k}_{Big})_i \text{ for } i = 1, \ldots, 9 \\ \mathbf{c} = \mathbf{w}_{10} &= M^*_{Big} \cdot \mathbf{w}_9^{(-1)} + (\mathbf{k}_{Big})_{10}, \end{aligned}$$

where the plaintext $\mathbf{p}$, the ciphertext $\mathbf{c}$, the intermediate vectors $\mathbf{w}_i$'s and the subkeys $(\mathbf{k}_{Big})_j$'s are in $\mathfrak{B}$, the state space of Big-BES. Obviously, Big-BES is the natural extension of BES where $\mathbb{F}$ is replaced by R.

3.3 Embedding BES in Big-BES

Let $\Phi : B \to \mathfrak{B}$ that acts on each $\mathbb{F}$-component applying the function e. This function maps the subkeys and the plaintext of BES in a subset $\mathfrak{B}_E$ of $\mathfrak{B}$ such that a BES encryption can be described by a Big-BES encryption restricted on $\mathfrak{B}_E$. Hence, the following diagram

$$\begin{array}{ccc} B & \xrightarrow{\Phi} & \mathfrak{B}_E \subset \mathfrak{B} \\ {\scriptstyle (\mathbf{k})_i \to} \downarrow & \text{BES} \quad \text{Big-BES} & \downarrow {\scriptstyle \leftarrow \Phi((\mathbf{k})_i)} \\ B & \xleftarrow{\Phi^{-1}} & \mathfrak{B}_E \subset \mathfrak{B} \end{array} \tag{6}$$

commutes. This condition is necessary to ensure that a Big-BES encryption on $\mathfrak{B}_E$ really corresponds to a BES encryption. More generally, we can replace e by a function F defined by $F(x) = (x, c \cdot x)$, for a constant $c \in \mathbb{F}$. Even with this generalisation the diagram (6) still commutes. Thus, we have

$$\text{Big-BES}_{\Phi((\mathbf{k}_B)i)} \circ \Phi(\mathbf{b}) = \left(\text{BES}_{(\mathbf{k}_B)_i}(\mathbf{b}),\, c \cdot \text{BES}_{(\mathbf{k}_B)_i}(\mathbf{b})\right). \tag{7}$$

So, we have found a restriction of Big-BES that consists more or less to duplicate BES. It seems to be clear that such a restriction has the same security properties as BES.

4 Attack on Big-BES

4.1 A Detailed Description of Big-BES

In this subsection, we will describe Big-BES in more details by writing the ciphertext in an expression depending on the subkeys and the plaintext.

In order to simplify the notation, we will denote the subkeys as $\mathbf{k}_i \in \mathfrak{B}$ for $i = 0, \ldots, 10$ and the matrices M_B resp. M_B^* as M resp. M^*. An element of $\mathfrak{B} = R^{128}$ will be represented by two elements of $\mathbb{F}^{128}$, for instance $\mathbf{k}_i := ((\mathbf{k}_i)_1, (\mathbf{k}_i)_2) \in \mathbb{F}^{128} \times \mathbb{F}^{128}$. Then, we will represent the plaintext with the pair $(\mathbf{x}, \mathbf{y})$, the vectors $\mathbf{w}_i$ with $(\mathbf{u}_i, \mathbf{v}_i) \quad (i = 0 \ldots 10)$ and the ciphertext $(\mathbf{u}_{10}, \mathbf{v}_{10})$ with $(\mathbf{u}_c, \mathbf{v}_c)$.

Now, we would like to describe $\mathbf{u}_i$ and $\mathbf{v}_i$ in some expressions depending on $(\mathbf{x}, \mathbf{y})$ and $((\mathbf{k}_i)_1, (\mathbf{k}_i)_2)$. To this end, we first notice that the matrix multiplication $M_{Big} \cdot \mathbf{b}$, $\mathbf{b} \in \mathfrak{B}$, can be simply computed as follows,

$$M_{Big} \cdot \mathbf{b} = M_{Big} \cdot (\mathbf{b}_1, \mathbf{b}_2) = (M \cdot \mathbf{b}_1, M \cdot \mathbf{b}_2),$$

where $\mathbf{b}_1,\ \mathbf{b}_2 \in \mathbb{F}^{128}$. Namely, since every elements of M_{Big} is the image under the trivial embedding of the elements of M_B, we see that such a matrix multiplication operates identically on the two $\mathbb{F}^{128}$-components of $\mathbf{b}$. An other important operation in Big-BES is the componentwise inversion of the vectors $\mathbf{w}_i$, $(0 \leq i \leq 10)$. In the description of Big-BES, we will suppose here that $\mathbf{u}_i$ has no elements of $\mathbb{F}$ equal to zero. Hence, we have

$$\mathbf{w}_i^{(-1)} = (\mathbf{u}_i, \mathbf{v}_i)^{(-1)} = (\mathbf{u}_i^{-1}, \mathbf{v}_i \cdot \mathbf{u}_i^{-2}),$$

where $\cdot$ denotes the componentwise multiplication.
We are now in a position to describe the vectors $\mathbf{u}_i$, $\mathbf{v}_i$. Firstly, for $i = 0$ we have obviously

$$(\mathbf{u}_0, \mathbf{v}_0) = (\mathbf{x} + (\mathbf{k}_0)_1, \mathbf{y} + (\mathbf{k}_0)_2)$$

and after the first round we obtain

$$(\mathbf{u}_1, \mathbf{v}_1) = \left(M \cdot (\mathbf{x} + (\mathbf{k}_0)_1)^{-1} + (\mathbf{k}_1)_1, M \cdot \left((\mathbf{y} + (\mathbf{k}_0)_2) \cdot (\mathbf{x} + (\mathbf{k}_0)_1)^{-2}\right) + (\mathbf{k}_1)_2\right).$$

We remark that the vectors $\mathbf{v}_i$ are obtained by iterating some affine functions depending on the subkeys and $\mathbf{x}$. So, we can write $\mathbf{v}_c$ in the following form

$$\mathbf{v}_c = A_0 \cdot \mathbf{y} + A_0 \cdot (\mathbf{k}_0)_2 + A_1 \cdot (\mathbf{k}_1)_2 + \cdots + A_9 \cdot (\mathbf{k}_9)_2 + (\mathbf{k}_{10})_2, \tag{8}$$

where A_i are some matrices depending on the subkeys $(\mathbf{k}_j)_1$, $(0 \leq j \leq 9)$ and $\mathbf{x}$. They are defined such that

$$A_i \cdot p = M^* \cdot ((\cdots M \cdot ((M \cdot (p \cdot \mathbf{u}_i^{-2})) \cdot \mathbf{u}_{i+1}^{-2}) \cdots) \cdot \mathbf{u}_9^{-2}),$$

for all $p \in \mathbb{F}^{128}$. Since the componentwise multiplication with the vectors $\mathbf{u}_j$ can be represented by the diagonal matrix $\mathrm{diag}(\mathbf{u}_j)$, we have

$$A_i = M^* \cdot \mathrm{diag}(\mathbf{u}_9^{-2}) \cdot M \cdot \mathrm{diag}(\mathbf{u}_8^{-2}) \cdots M \cdot \mathrm{diag}(\mathbf{u}_i^{-2}). \tag{9}$$

In order to finish our description of Big-BES, it remains only to express the $\mathbf{u}_i$'s as some functions of $\mathbf{x}$ and the subkeys $(\mathbf{k}_i)_1$. This is given by noticing that the $\mathbf{u}_i$'s are the intermediate state vectors of a BES encryption taking $\mathbf{x}$ as plaintext and $(\mathbf{k}_i)_1$ as subkeys. Thus, we have

$$\mathbf{u}_c = \mathrm{BES}_{(\mathbf{k}_i)_1}(\mathbf{x}),$$

and finally, we have expressed the ciphertext $(\mathbf{u}_c, \mathbf{v}_c)$ as a function of the plaintext $(\mathbf{x}, \mathbf{y})$ and the subkeys $((\mathbf{k}_i)_1, (\mathbf{k}_i)_2)$.

Remark. If we fix $\mathbf{x}$ the first component of the plaintext and if we consider the subkeys as some fixed parameters, we notice from (8) that $\mathbf{v}_c$, the second component of the ciphertext is simply given by the affine transformation

$$\mathbf{v}_c = A \cdot \mathbf{y} + \mathbf{r}, \tag{10}$$

where $A \in \mathbb{F}^{128 \times 128}$ and $\mathbf{r} \in \mathbb{F}^{128}$ are some constants. However, this representation works under the assumption that the $\mathbf{u}_i$'s $(0 \leq i \leq 9)$ have no component equal to zero. Actually, this assumption is quite strong, because it concerns $10 \cdot 128 = 1280$ elements of $\mathbb{F}$. Namely, the probability that all these elements are not equal to zero is

$$\left(\frac{255}{256}\right)^{1280} = 0,006672 \approx \frac{1}{150}. \tag{11}$$

4.2 The Attack

The attack we will describe below is a chosen plaintext attack and it exploits the fact that Big-BES possesses certain linearity properties we already mentioned above. In fact, if we fix $\mathbf{x}$ we know that the second component of the ciphertext is affine in $\mathbf{y}$ if there is no 0-inversion in the vectors $\mathbf{u}_i \quad (0 \leq i \leq 9)$. It is not

difficult to see that this affine property depends only on $\mathbf{x}$, once the subkeys are fixed. The idea of our attack will be to collect sufficiently many $\mathbf{x}$'s such that the affine property is verified. From this, we will be able to find the subkeys $(\mathbf{k}_i)_1$ by a sieving method.

The affine property. Here, for a given $\mathbf{x}$, we show how we can check the affine property, i.e, that no 0-inversion occurred in the $\mathbf{u}_i$. We consider here a Big-BES encryption with some given fixed subkeys, two plaintexts of the form $(\mathbf{x}, \mathbf{y})$, $(\mathbf{x}, \mathbf{y}')$ and their corresponding ciphertexts $(\mathbf{u}, \mathbf{v})$, $(\mathbf{u}, \mathbf{v}')$. By applying the equality (10) for the two above plaintexts, we have

$$A \cdot \mathbf{y} = \mathbf{v} + \mathbf{r} \tag{12}$$

$$A \cdot \mathbf{y}' = \mathbf{v}' + \mathbf{r}. \tag{13}$$

Since the underlying field is of characteristic two, $\mathbf{r} + \mathbf{r} = 0$ and we obtain

$$A \cdot (\mathbf{y} + \mathbf{y}') = \mathbf{v} + \mathbf{v}' \tag{14}$$

by adding (12) with (13). Hence, we notice that the sum $\mathbf{v} + \mathbf{v}'$ is constant when $\mathbf{y} + \mathbf{y}'$ is constant. To determine if the affine property holds for a given $\mathbf{x}$, we will encrypt 4 plaintexts of the form $(\mathbf{x}, \mathbf{y}), (\mathbf{x}, \mathbf{y}'), (\mathbf{x}, \mathbf{y}'')$ and $(\mathbf{x}, \mathbf{y} + \mathbf{y}' + \mathbf{y}'')$ and check that the corresponding ciphertexts satisfy

$$\mathbf{v} + \mathbf{v}' + \mathbf{v}'' + \mathbf{v}''' = 0.$$

Hence, we should be able to conclude that no 0-inversion occurred in the Big-BES encryption with this $\mathbf{x}$. Notice that the inversion of the form $(0, 0)^{(-1)} = (0, 0)$ preserves the linearity and that some bad $\mathbf{x}$ could pass the test. However, since this should occur for 4 plaintexts, we will assume that the probability of such an event is negligible.

The attack. In this attack, we assume that we have a Big-BES encryption oracle that allows to encrypt any plaintexts. The goal is to find the subkeys used in this Big-BES encryption. We describe below the different steps of this attack.

1. We pick some vectors $\mathbf{x} \in \mathbb{F}^{128}$ at random and check if the affine property holds for each of them by using our oracle to encrypt the plaintexts $(\mathbf{x}, \mathbf{y})$ needed to this end. Then, we collect the $\mathbf{x}$'s that satisfied the affine property. The set of these collected vectors is denoted as $P = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$.
2. For every $1 \leq i \leq 128$, we search for the subkey element $(\mathbf{k}_0)_1^i$ that satisfies

   $$(\mathbf{k}_0)_1^i \neq \mathbf{x}_j^i$$

   for all $1 \leq j \leq n$. If this subkey element is not uniquely determined by the set P, we can look for some new $\mathbf{x} \in P$ that allows us to exclude some values of $(\mathbf{k}_0)_1^i$. All these new $\mathbf{x}'s$ are collected in P too. At the end of this step, we can deduce the value of the subkey $(\mathbf{k}_0)_1$.

3. For every $\mathbf{x}_i \in P$, we compute $\mathbf{u}_{0i} = \mathbf{x}_i + (\mathbf{k}_0)_1$ and we collect all the vectors $M \cdot \mathbf{u}_{0i}^{(-1)}$ in a set denoted as P_1. Again, we know that the subkey element $(\mathbf{k}_1)_1^i$ can be deduced as above by the statement

$$(\mathbf{k}_1)_1^i \neq \left(M \cdot \mathbf{u}_{0j}^{(-1)}\right)^i$$

for all $1 \leq j \leq n$. If it is needed, we can complete the set P_1 with new appropriate vectors in order to find the right subkey $(\mathbf{k}_1)_1$.
4. We continue the same process by computing some successive sets P_j, $(2 \leq j \leq 9)$ and by using the same sieving method as above to find the subkeys $(\mathbf{k}_1)_j$, $(2 \leq j \leq 9)$.
5. We pick an element $\mathbf{x} \in \mathbb{F}^{128}$ and compute its corresponding ciphertext $\mathbf{u}_{10}$ by the encryption oracle. The already known subkeys allow us to calculate $\mathbf{u}_9$ and the subkey $(\mathbf{k}_1)_{10}$ is found by the equality

$$(\mathbf{k}_1)_{10} = M^* \cdot \mathbf{u}_9^{(-1)} + \mathbf{u}_c.$$

6. Since the subkeys $(\mathbf{k}_1)_i$'s are known, we can now compute the matrices A_i's corresponding to any plaintext $(\mathbf{x}, \mathbf{y})$ by using the formula (9). By choosing many plaintexts $(\mathbf{x}, \mathbf{y})$ and computing the corresponding matrices A_i's and the corresponding ciphertext $\mathbf{v}_{10}$, we can find a sufficient number of linear equations taking the subkeys $(\mathbf{k}_2)_i$'s as variables. Namely, applying the linear equation (8) to many different plaintexts provide a linear system allowing to determine the $(\mathbf{k}_2)_i$'s.

Complexity. Here we estimate roughly the number of Big-BES encryptions required to this attack. The biggest amount of computations is required for collecting the elements $\mathbf{x} \in P_9$ satisfying the affine property. To find each of those $\mathbf{x}$, we have to try 150 candidates in average (11) with which we need to compute 4 encryptions to test the affine property. We estimate in the Appendix A that P_9 contains about 2100 elements. Thus, we conclude that $150 \cdot 2100 \cdot 4 = 1'260'000 \approx 2^{20}$ encryptions have to be calculated. This demonstrates that Big-BES is terribly weak.

Implications for BES. As we have seen in (6), breaking BES corresponds to breaking Big-BES on the restricted set $\mathfrak{B}_E$. From this, we remark that our attack against Big-BES can not be applied against BES. Indeed, this is due to the fact that all 0-inversions in $\mathfrak{B}_E$ are of the form $(0,0)^{-1} = (0,0)$.

5 CES

In this section we will construct a similar extension to AES called CES as for " Crooked Encryption System ". This extension is natural in the sense that the Murphy-Robshaw-like extension of CES is indeed Big-BES, so that we have a

kind of commutative extension diagram

$$\begin{array}{ccc} \text{AES} & \xrightarrow{\mathbb{F}\to R} & \text{CES} \\ {\scriptstyle \mathbb{F}^{16}\to\mathbb{F}^{128}}\downarrow & & \downarrow{\scriptstyle R^{16}\to R^{128}} \\ \text{BES} & \xleftarrow{\mathbb{F}\leftarrow R} & \text{Big-BES} \end{array} \tag{15}$$

5.1 Definition

As for Big-BES, CES is defined on the commutative ring R with the operations defined by (2), (3), (4). This extension is obtained by replacing the field $\mathbb{F}$ by the ring R and by mapping the constant elements of the AES defined on $\mathbb{F}$ under the trivial embedding (5). This concerns the elements of the matrices MIX_A and R_A and the coefficients of the polynomial q of equation (1). These two new matrices will be denoted as MIX_C resp. R_C. Hence, CES is a cipher having the state space R^{16} and a round has the form:

Round. Let $\mathbf{b} \in R^{16}$ a state vector. Then the i^{th} round is

$$\mathbf{b} \mapsto M_C \cdot q(\mathbf{b}^{(-1)}) + (\mathbf{k}_C)_i \quad \text{for } 1 \le i \le 9,$$

where $M_C = \text{MIX}_C \cdot R_C$, $(\mathbf{k}_C)_i$ denotes the i^{th} subkey and the polynomial q operates componentwise. Note that the 0-round is simply a subkey addition and the 10^{th} is obtained by replacing M_C by another matrix.

5.2 The Embedding

As in Section 3.3, we notice that e embeds AES in CES quite well since $q(e(x)) = e(q(x))$.

5.3 Attack against CES

Here we show that the attack in the subsection 4.2 can be easily adapted to CES. We will use the same principle that allows to detect a 0-inversion in $\mathbb{F}$. In order to show this fact, we have still to check that CES transforms the second component of a plaintext linearly when the first component is fixed.

First, we note that

$$q(\mathbf{x}, \mathbf{y}) = (q(\mathbf{x}), \mathtt{05} \cdot \mathbf{y}),$$

where $\mathbf{x}, \mathbf{y} \in \mathbb{F}^{16}$ and q operates componentwise on the 16 components of R^{16} resp. $\mathbb{F}^{16}$. Hence, a CES encryption is linear in $\mathbf{y}$ when $\mathbf{x}$ is fixed, therefore we can apply the same attack consisting in checking a linear property for some given $\mathbf{x}$. Notice also that the first component is transformed as in an AES encryption.

Complexity. For the complexity estimation of this attack, we need to compute the ratio of the plaintexts for which no 0-inversion occur in a CES-encryption. This is given by

$$\left(\frac{255}{256}\right)^{160} = 0.5346 = \frac{1}{1.87}$$

Hence, we have to encrypt 1.87 plaintexts in average until we find the required one for the step 1 of the attack. Thus, the number of CES encryptions needed for this cryptanalysis is $1.87 \cdot 1850 \cdot 4 = 13'838 \approx 2^{14}$ (See Appendix A) .

Implications for the AES. As in the BES case, this attack can not be applied against AES. Namely, the embedding function e induces only 0-inversions of the form $(0,0)^{-1}$.

6 Discussion about Extending Block Ciphers

In this paper, we constructed some extensions of AES and BES by replacing the field $\mathbb{F}$ by a natural extension of it. The first natural extension we thought was inspired by the p-adic numbers. So, we considered formal sums $\sum_{i=0}^{\infty} x_i \cdot 2^i$ where $(x_0, x_1, \dots,) \in \mathbb{F}^{\mathbb{N}}$. To simplify such expressions, we chose to define our extension by taking the projection modulo 4 of these formal sums. This equivalently consists of terms of the form $x_0 + 2 \cdot x_1$ where $x_0, x_1 \in \mathbb{F}$. In this structure, we remark that the inversion has the desired property, namely this operation is linear in x_1 when we fix x_0. This led to the fact that a new BES like block-cipher defined on this ring is weak. Indeed, the attack applied in 4.2 could easily be adapted to this case provided that we define the inversion of an element of the form $0 + 2 \cdot x_1$ similarly as in (4). Nevertheless, it turns out that the calculations are a little bit more complicated than in R. As a conclusion, we would like to mention that there are probably some other extensions of $\mathbb{F}$ presenting similar properties, but R seemed to be one of the simplest and most appropriate.

So, we have shown that several natural extensions for AES and BES are weak. Note that this kind of extension can typically be used in order to prevent from some power analysis or other side channel attacks. Our result demonstrate that this should be done with extreme care.

7 Conclusion

We have described some trivially weak extensions of BES respectively of AES although the extensions are quite natural in the sense that we simply replace $\mathbb{F}$ by another algebraic structure. This shows that some similar results can be obtained in the public key cryptography as well as in the symmetric key cryptography using some Hensel-like ideas. In particular, we have shown that a supposedly

secure block cipher can be naturally embedded in a very weak one by modifying its underlying algebraic structure.

Of course, this construction did not allow to find any security flaws against AES and BES. Moreover, embedding AES in a weak block cipher is certainly not the right way in order to find a cryptanalysis against it. The reason of this comes from the embedding function. Since this one has to be preserved under the basic round operations, it will have a very simple form. So, the equivalent ciphers to AES induced by it will consist in some duplications of AES. Therefore, despite of the elegance of the Murphy-Robshaw algebraic representation of AES, attacks on the BES extension may have no consequence at all for the security of AES. However, our work did not allow to conclude that this must be the case.

References

1. K. Aoki and S. Vaudenay, *On the Use of GF-Inversion as a Cryptographic Primitive*, Selected Areas in Cryptography, 2003.
2. E. Barkan and E. Biham, *In How Many Ways Can You Write Rijndael ?*, Advances in Cryptology - Asiacrypt '02, LNCS vol. 2501, pp. 160-175, Springer-Verlag, 2002.
3. E. Biham and A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard,* Springer-Verlag, New York, 1993.
4. D. Catalano, P. Q. Nguyen and J. Stern, *The Hardness of Hensel Lifting: The Case of RSA and Discrete Logarithm,* Advances in Cryptology - Asiacrypt '02, LNCS vol. 2501, pp. 299-310, Springer-Verlag, 2002.
5. N. Courtois and J. Pieprzyk, *Cryptanalysis of Block Ciphers with Overdefined Systems of Equations,* Advances in Cryptology - Asiacrypt '02, LNCS vol. 2501, pp. 267-287, Springer-Verlag, 2002.
6. J. Daemen and V. Rijmen, *The Design of Rijndael: AES-The Advanced Encryption Standard*, Springer-Verlag, 2002.
7. N. Ferguson, R. Shroeppel and D. Whiting, *A Simple Algebraic Representation of Rijndael,* Selected Areas in Cryptography '01, LNCS vol. 2259, pp. 103-111, Springer-Verlag, 2001.
8. R. Lidl and H. Niederreiter, *Finite Fields,* Encyclopedia of Mathematics and its Applications 20, Cambridge University Press, 1997.
9. M. Matsui, *Linear Cryptanalysis method for DES Cipher,* Advances in Cryptology - Eurocrypt '93, LNCS vol. 765, pp. 386-397, Springer-Verlag, 1994.
10. S. Murphy and M.J.B Robshaw, *New Observations on Rijndael*, NIST AES website `csrc.nist.gov/encryption/aes`, August 2000.
11. S. Murphy and M.J.B Robshaw, *Essential Algebraic Structure Within the AES,* Advances in Cryptology - Crypto '02, LNCS vol. 2442, pp. 1-16, Springer-Verlag, 2002.
12. National Institute of Standards and Technology, *Advanced Encryption Standard*, FIPS 197, 26 November 2001.
13. T. Okamoto and S. Uchiyama, *A New Public-Key Cryptosystem as Secure as Factoring,* Advances in Cryptology - Eurocrypt '98, LNCS vol. 1403, pp. 308-318, Springer-Verlag, 1998.
14. T. Satoh and K. Araki, *Fermat Quotients and the Polynomial Time Discrete Log Algorithm for Anomalous Elliptic Curves*, Commentarii Math. Univ. St. Pauli, 47, pp. 81-92, 1998.

15. Nigel P. Smart, *The Discrete Logarithm Problem on Elliptic Curves of Trace One,* Journal of Cryptology, 12, pp. 193-196, 1999.

A Computation of the size of P_9.

Here we estimate the number of all elements $\mathbf{x} \in \mathbb{F}^{128}$ required for the sieving method, i.e. the cardinality of P_9. To this goal, we suppose that each $\mathbb{F}$-component of the subkeys is sieved with elements that are picked randomly in $\mathbb{F}$ and that all these components are independent. Hence, we will estimate the number of such $\mathbf{x}$ needed for the sieving step of one $\mathbb{F}$-component and assume that they will sieve the other ones.

First, we consider the following computation. Let $n \in \mathbb{N}$ and $a_1, \ldots, a_n \in_U \{1, 2, \ldots, z\}$ a random sequence with uniform distribution. We compute the number of values of the set $\{1, \ldots, z\}$ that lie in the sequence $a_1, \ldots, a_n$ in average. This is given by calculating the following expected value:

$$\mathbb{E}\left(\sum_{i=1}^{z} \mathbf{1}_{\{\exists\, j:\, a_j = i\}}\right) = z \cdot \mathrm{Prob}\left(\exists\, j :\ a_j = 1\right) = z \cdot \left(1 - \left(\frac{z-1}{z}\right)^n\right). \quad (16)$$

Cardinality of P_9. We set $z = 256$ and we will choose $n = 1800$ elements for the set P. To obtain the number of elements of P_9, it remains to compute how many elements are missing for the sieving of all $\mathbb{F}$-subkeys elements. From (16), we deduce that $z \cdot \left(\frac{z-1}{z}\right)^n = 256 \cdot \left(\frac{255}{256}\right)^{1800} = 0.22316$ elements are missing in average. Since we have to sieve 1280 $\mathbb{F}$-elements, we can expect that $1280 \cdot 0.22316 \approx 285$ $\mathbf{x}$'s will have to be added to the set P in order to achieve our sieving method. Thus, $\#P_9 = 1800 + 285 \approx 2100$ in the Big-BES case. A similar computation provides that $\#P_9 \approx 1850$ in the CES case.

Remark. We have chosen $n = 1800$, because it can be shown that this value minimizes $\#P_9$.