Università degli Studi di Trento

Facoltà di Scienze Matematiche Fisiche e Naturali
Dipartimento di Matematica

Dottorato in Matematica
Ciclo XXI

On algebraic and statistical properties of AES-like ciphers

Anna Rimoldi

Supervisor: Prof. M. Sala

Head of PhD School: Prof. A. Valli

Università degli Studi di Trento

Facoltà di Scienze Matematiche Fisiche e Naturali

Dipartimento di Matematica

Dottorato in Matematica

Ciclo XXI

On algebraic and statistical properties of AES-like ciphers

Ph.D.Thesis of: Anna Rimoldi

Supervisors: Prof. M. Sala

Head of PhD School: Prof. A. Valli

Contents

Acknowledgment

First of all, I would like to express sincere gratitude to my supervisor Prof. Massimiliano Sala for his encouragement, his advice and research support throughout my Master's and PhD studies.

I am grateful to my thesis defense Committee, Prof. Carlo Traverso, Prof. Teo Mora and Dr. Ludovic Perret, for their helpful suggestions.

I would like to thank the whole Department of Mathematics of University of Trento for its support, especially Prof. Andrea Caranti, Prof. Marco Sabatini and Prof. Alberto Valli.

Furthermore, thanks to Dr. Giacomo Aletti, Dr. Guido Bertoni, Prof. Francesca Dalla Volta, Dr. Lilli Fragneto, Dr. Ilia Toli for their helpful comments.

Sincere thanks to Dr. Fabrizio Caruso for helping me in the more computational part of this work and for many interesting discussions.

I want to thank all my fellow PhD students and in particular Federica, Elisa, Stefano e Philipp; I want also thank Lara Maines for her scientific contributions and all the fantastic guys in our group.

My last but important acknowledgment is for my family and for my friends Claudia, Marco, Paolo and Roberto and Yudis.

Some Notation

$\mathbb{N} = \{0, 1, 2, \ldots\}$	
$\mathbb{F}_q$	finite field with q elements
$V = (\mathbb{F}_q)^r$	vector space over $\mathbb{F}_q$ of dimension r
$\mathrm{Sym}(V)$	symmetric group on V
$\mathrm{Alt}(V)$	alternating group on V
$\mathrm{GL}(V)$	group of all linear permutations of V
$C^\perp$	the orthogonal space of any vector space $C < V$ w.r.t. the standard scalar product in V
$\mathrm{Im}(f)$	image of any function $f : S \to T$, with S, T any sets.
$\mathrm{w}(v)$	Hamming weight of the vector $v \in (\mathbb{F}_q)^r$
m	Rijndael polynomial, $\mathsf{m} = x^8 + x^4 + x^3 + x + 1 \in \mathbb{F}_2[x]$
$\lceil \ell \rceil$	is $\min\{n \in \mathbb{Z} \mid n \geq \ell\}$ (ceiling)
$\mathsf{o}(\sigma)$	the order of permutation σ
i.e.	id est
e.g.	exempli gratia
w.l.o.g.	without loss of generality
w.r.t.	with respect to
s.t.	such that
NIST	National Institute for Standards and Technology (US)

Introduction

The Advanced Encryption Standard (AES) is nowadays the most widespread block cipher in commercial applications. It represents the state-of-art in block cipher design and provides an unparalleled level of assurance against all known cryptanalytic techniques, except for its reduced versions. Moreover, there is no known efficient way to distinguish it from a set of random permutations.

The AES (and other modern block ciphers) presents a highly algebraic structure, which led researchers to exploit it for novel algebraic attacks. These tries have been unsuccessful, except for academic reduced versions.

Starting from an intuition by I. Toli, we have developed a mixed algebraic-statistical attack. Using the internal algebraic structure of any AES-like cipher, we build an algebraic setting where a related-key (statistical) distinguishing attack can be mounted. Our data reveals a significant deviation of the full AES-128 from a set of random permutations. Although there are recent successful related-key attacks on the full AES-192 and the full AES-256 (with non-practical complexity), our attack would be the first-ever practical distinguishing attack on the full AES-128 (to the best of our knowledge).

In Part I we provide some preliminaries and sketch a survey of known attacks on the AES versions, in particular on AES-128.

In Chapter 1 we give some basic algebraic background and we summarize the notion of a *block cipher*, with its link to hash functions. In particular, we introduce the class of *translation based* cryptosystems, which are ciphers enjoying some interesting algebraic properties.

In Chapter 2 we describe the three main translation-based cryptosystems: AES, SERPENT and PRESENT.

In Chapter 3 we briefly report on known attacks on AES, including structural, statistical and algebraic attacks. This chapter can be skipped in a first reading of this thesis.

In Part II we present our attack and its algebraic setting.

In Chapter 4 we give some algebraic embeddings of a translation based cipher into a much larger cipher. These embeddings are designed to lower the non-linearity of the encryption functions. Two of them are practical and can be applied in principle to AES, PRESENT and SERPENT. In particular, the *orbit* representation works well with AES-128 and PRESENT. However, with group theory proofs we also show that no representation/embedding can completely linearize the full AES-128.

In Chapter 5 we use the orbit representation for AES-128 and we are able to find sets of related matrices with related keys such that the encryption action can be shown to differ significantly from the behavior of a set of random permutations. Our attack may be seen as an extremely refined version of a Marsaglia Diehard test.

Part I

Preliminaries

Preliminaries and notation

In this chapter we recall well-known results in group theory and finite field theory [LN97] in order to fix the notation we will use in the sequel.
We also outline some basic ideas about *block ciphers*, their security level and their cryptanalysis. Definitions and results are mainly from [Sti95], [CW09] and [DR02].
In the last section we will give an overview of the statistical tests adopted by NIST [NIS00] to evaluate the random behavior of the AES candidates.

1.1 Algebraic background

Let $n \geq 2$ be an integer. Let $V = (\mathbb{F}_2)^n$ be the vector space over the finite field $\mathbb{F}_2$ of dimension n. We denote by $\mathrm{Sym}(V)$ and $\mathrm{Alt}(V)$, respectively, the symmetric and alternating group on V. We denote by $\mathrm{GL}(V)$ the group of all linear permutations of V. We recall the well-known formulas:

$$|\mathrm{Sym}(V)| = 2^n!, \quad |\mathrm{Alt}(V)| = \frac{2^n!}{2} \quad |\mathrm{GL}(V)| = \prod_{h=0}^{n-1}(2^n - 2^h) < 2^{n^2} .$$

Given a finite group G, we say that G can be linearized if there is an injective morphism $\rho : G \to \mathrm{GL}(V)$ (this is called a "faithful representation" in representation theory).

Definition 1.1.1. *A (linear)* representation *of a group G over a vector space V is a group homomorphism $\rho : G \to \mathrm{GL}(V)$. If ρ is injective, it is called* faithful.

If G can be linearized, then, for any element $g \in G$, we can compute a matrix M_g corresponding to the action of g over V (via ρ). The matrix computation is easy, since it is enough to evaluate g on a basis of V.
If $\rho : G \to \mathrm{GL}(V)$ is a representation of G on V , then we often write gv instead of $\rho(g)v$, if no confusion arises. Also, G is said to *act linearly* on V, and V is called a *G-module*. The *degree* of the representation is by definition the dimension of V. By taking $V = 2^{|G|}$ we can always linearize G over V via the so-called *regular* representation, but of course this is huge and usually impractical.

Definition 1.1.2. *Let $G = \{g_1, \ldots, g_n\}$ be a finite group. Let V be a vector space with basis $\{e_{g_1}, \ldots, e_{g_n}\}$. The* regular *representation $\rho : G \to \mathrm{GL}(V)$ is defined by $\rho(g_i)(e_{g_j}) = e_{g_i g_j}$ (where $g_i g_j$ is the group product).*

Definition 1.1.3 (Equivalence of Representations). *Two representations ρ_1 and ρ_2 over a vector space V are said to be* equivalent *if they are related by conjugation, i.e. there is $h \in \mathrm{GL}(V)$ such that $\rho_2(g) = h(\rho_1(g))h^{-1}$, $\forall g \in G$.*

1.1.1 Finite Fields

For any prime p and any positive $m \in \mathbb{N}$, $\mathbb{F}_{p^m}$ is the field with p^m elements (unique up to field isomorphism). It contains an isomorphic copy of $\mathbb{F}_p$ and can thus be thought as an extension of $\mathbb{F}_p$. On the other hand, we can construct any $\mathbb{F}_{q^s}$ from $\mathbb{F}_q$ with $q = p^m$ elements, as follows.
Let $f \in \mathbb{F}_q[x]$ be an irreducible polynomial of degree m. We can consider the quotient $R = \mathbb{F}_q[x]/(f)$, where (f) is the ideal generated by f in $\mathbb{F}_q[x]$. By considering the natural projection $\pi : \mathbb{F}_q[x] \to R$, we call $\alpha = \pi(x)$ and clearly any element of R can be uniquely expressed as a polynomial in α of degree less than m:

$$R = \left\{ \sum_{i=0}^{m-1} a_i \alpha^i \mid a_i \in \mathbb{F}_q \right\}$$

with the condition $f(\alpha) = 0$.

Theorem 1.1.4. *$R = \mathbb{F}_q[x]/(f)$ is a field and $R \cong \mathbb{F}_{q^m}$.*

We denote by $\mathbb{F}_q^*$ the multiplicative group of non-zero elements of $\mathbb{F}_q$.

Theorem 1.1.5. *For any finite field $\mathbb{F}_q$, the multiplicative group $\mathbb{F}_q^*$ is cyclic.*

A generator of the cyclic group $\mathbb{F}_q^*$ is called a *primitive element* of $\mathbb{F}_q$.

Definition 1.1.6. *An irreducible polynomial $f \in \mathbb{F}_q[x]$ is* primitive *if its roots are primitive elements.*

We conclude this subsection by observing that for any q and m there are indeed irreducible polynomials of degree m over $\mathbb{F}_q$ and some of them are primitive.

1.1.2 Permutation polynomials

Definition 1.1.7. *A polynomial $f \in \mathbb{F}_q[x]$ is a* permutation polynomial *of $\mathbb{F}_q$ if the associated polynomial function $f : c \mapsto f(c)$ from $\mathbb{F}_q$ into $\mathbb{F}_q$ is a permutation of $\mathbb{F}_q$.*

If f is an affine map $f : x \mapsto ax + b$ ($a \neq 0$), we say that f is a linear polynomial.

We note the following easy results:

1. Every linear polynomial over $\mathbb{F}_q$ is a permutation polynomial of $\mathbb{F}_q$.
2. The monomial x^n is a permutation polynomial of $\mathbb{F}_q$ if and only if
$$\gcd(n, q-1) = 1.$$

Permutation polynomials of $\mathbb{F}_q$ of degree less then q can be combined by the operation of composition and subsequent reduction modulo $x^q - x$. The set of permutation polynomials of $\mathbb{F}_q$ of degree less then q forms a group, which is isomorphic to $\mathrm{Sym}(\mathbb{F}_q)$. Then, the symmetric group $\mathrm{Sym}(\mathbb{F}_q)$ and its subgroups can be represented as groups of permutation polynomials.

Theorem 1.1.8. *For $q > 2$, the symmetric group $\mathrm{Sym}(\mathbb{F}_q)$ is generated by x^{q-2} and all linear polynomials over $\mathbb{F}_q$.*

1.2 Block ciphers

Block ciphers form an important class of cryptosystems in symmetric key cryptography. Stream ciphers [Rue92] form another class. We are interested only in cryptosystems of type block ciphers. These are algorithms that encrypt and decrypt blocks of data (with fixed length[1]) according to a shared secret key. They are commonly used to provide confidentiality during information transmission and storage. We can formally describe such a cryptosystem using the following definition:

Definition 1.2.1. *A cryptosystem is a pair $(\mathcal{M}, \mathcal{K})$, where:*

- *$\mathcal{M}$ is a finite set of possible messages (plaintexts, ciphertexts);*
- *$\mathcal{K}$, the key-space, is a finite set of possible keys;*
- *we have encryption and decryption functions for any key $k \in \mathcal{K}$:*
$$\phi_k : \mathcal{M} \to \mathcal{M}, \quad \psi_k : \mathcal{M} \to \mathcal{M}, \quad \phi_k, \psi_k \in \mathrm{Sym}(\mathcal{M})$$
such that
$$\psi_k = (\phi_k)^{-1}.$$

Following the most used structure in modern ciphers, in the previous definition we set that the plaintext space coincides with the ciphertext space. W.l.o.g, we can consider $\mathcal{M} = (\mathbb{F}_q)^r$ and $\mathcal{K} = (\mathbb{F}_q)^\ell$, with r and ℓ positive integers, and we change slightly our previous definition.

[1]Actually, there is a recent approach that allows a slight change of the block length [CYK09]

Definition 1.2.2. *Let r and ℓ be natural numbers. Let ϕ be any function*

$$\phi : (\mathbb{F}_q)^r \times (\mathbb{F}_q)^\ell \to (\mathbb{F}_q)^r.$$

For any $k \in (\mathbb{F}_q)^\ell$, we denote by ϕ_k the function

$$\phi_k : (\mathbb{F}_q)^r \to (\mathbb{F}_q)^r, \quad \phi_k(x) = \phi(x, k).$$

We say that ϕ is a **algebraic** block cipher *if ϕ_k is a permutation of $(\mathbb{F}_q)^r$ for any key $k \in (\mathbb{F}_q)^\ell$.*

Under this conditions, we can also consider a block cipher as an indexed set of permutations $(\mathbb{F}_q)^\ell \to \mathrm{Sym}((\mathbb{F}_q)^r)$. Any key $k \in \mathcal{K}$ induces a permutation ϕ_k on $\mathcal{M}$. Since $\mathcal{M}$ is usually $V = (\mathbb{F}_2)^r$ for some $r \in \mathbb{N}$, we can consider $\phi_k \in \mathrm{Sym}(V)$.

We recall a typical communication scheme between two parties (traditionally known as Alice and Bob):

- Alice and Bob agree on the key $k \in \mathcal{K}$;
- Alice chooses a plaintext $x \in \mathcal{M}$, uses the encryption function ϕ_k to encrypt x and sends Bob the ciphertext $y = \phi_k(x)$ (where $y \in \mathcal{M}$);
- Bob knows the key k and hence the decryption function ψ_k, so that he recovers the original message x:
$$\psi_k(y) = \psi_k(\phi_k(x)) = x\,.$$

When Eve (the eavesdropper) intercepts the ciphertext, she should not be able to find easily the plaintext, because she does not know the key. Otherwise the system would be "weak" or "insecure". (For more details, see Subsection 1.2.1).

To achieve the desired security, most modern block ciphers are iterated ciphers that typically incorporate sequences of permutation and substitution operations. In fact, according to the ideas that Shannon proposed in his seminal paper [Sha49], the encryption process takes as input a plaintext and a random key and so proceeds through N similar rounds. In each round (except possibly for a couple, which may be slightly different) the iterated ciphers perform a non-linear substitution operation (or S-box) on disjoint parts of the input that provides "confusion", followed by a permutation (usually a linear/affine transformation) on the whole data that provides "diffusion". A cryptosystem reaches "confusion" if the relationship between plaintext, ciphertext and key is very complicated. The "diffusion" idea consists of spreading the influence of all parts of the input (plaintext and key) to all parts of the ciphertext. The operations performed in a round form the round function. The round function at the ρ-th round ($1 \leq \rho \leq N$) takes as inputs both the output of the $(\rho-1)$-th round and the subkey $k^{(\rho)}$ (also called round-key). Any round key $k^{(\rho)}$ is constructed starting from a master key[2] k of some specified length, e.g. $k \in \mathcal{K} = (\mathbb{F}_2)^\ell$ (nowadays we have $2^{64} \leq |\mathcal{K}| \leq 2^{256}$). The key schedule is a public algorithm (strictly dependent on the cipher) which constructs $N+1$ subkeys $(k^{(0)}, \ldots, k^{(N)})$.

Several independent formal definitions have been proposed for iterated block ciphers (or subclasses of them). Here we present three of them.
Stinson in [Sti95] gives the following definition of *substitution permutation network* (SPN for short):

Definition 1.2.3 (SPN). *Let m, b, N be positive integers. Let $r = mb$. Let $\pi_S : (\mathbb{F}_2)^m \to (\mathbb{F}_2)^m$ and $\pi_P : \{1, \ldots, r\} \to \{1, \ldots, r\}$ be permutations. Let $\mathcal{M} = (\mathbb{F}_2)^r$ and let $\mathcal{K}' \subset (\{0,1\}^r)^{N+1}$ consist of all possible key schedules that can be derived from a master key k using the key scheduling. For any key schedule $(k^0, \ldots, k^N)$ we encrypt the plaintext x using the following algorithm* [3]*:*

```
Require: $x, \pi_S, \pi_P, (k^0, \ldots, k^N)$
  $w^0 \leftarrow x$
  for $r \leftarrow 0$ to $N-2$ do
    $u^r \leftarrow w^{r-1} \oplus k^r$
    for $i \leftarrow 1$ to $b$ do
      $v_i^r \leftarrow \pi_S(u_i^r)$
    end for
    $w^r \leftarrow \left(v^r_{\pi_P(1)}, \ldots, v^r_{\pi_P(mb)}\right)$
  end for
  $u^{N-1} \leftarrow w^{N-2} \oplus k^{N-1}$
  for $i \leftarrow 1$ to $b$ do
    $v_i^{N-1} \leftarrow \pi_S(u_i^{N-1})$
  end for
  $y \leftarrow v^{N-1} \oplus k^N$
```

Algorithm 1: SPN Encryption

[2]also called session key.

[3]Only in the 8-th row of the following algorithm we have considered the *bit* components of v^r.

The author notes that this definition is too restricted and suggests some variations, as for example:

- to use more than one S-box (as for DES [Nat77], in which 8 different S-boxes are employed in each round);
- to include an invertible linear transformation in each round, either as a replacement for, or in addition to, a permutation operation (as for instance for AES (see Section 2.1)).

In [DR02] we can find another class of iterated block cipher, called the *key-alternating* block ciphers. This kind of ciphers are characterized by the following properties:

- `Alternation`: the cipher is defined as the alternated application of key independent round transformations and key additions; the first round key is added before the first round and the last round key is added after the last round.
- `Simple key addition`: the round keys are added to the state (the intermediate value) by means of a simple XOR. We have

  $$B[k] = \sigma[k^{(r)}] \circ \rho^{(r)} \circ \sigma[k^{(r-1)}] \circ \cdots \sigma[k^{(1)}] \circ \rho^{(1)} \circ \sigma[k^{(0)}]$$

  where $\sigma[k^{(i)}]$ is the key addition using the i-th round key $k^{(i)}$, $\rho^{(i)}$ is the i-th round transformation.

A special class of key-alternating block ciphers are the *key-iterated* block ciphers, in which all rounds (except possibly for a couple of those) use the same round transformation. An advantage of this class of ciphers is the fact that allows efficient implementations. We note that this kind of characterization is very general. It is therefore quite difficult to obtain general theoretical results.

Finally, we consider a more recent definition [CDVSar] that defines a class (see Definition 1.2.5), large enough to include some common ciphers, yet restricted enough to have simple criteria guaranteeing an interesting property of the cipher (for details see Section 4.5).

Let $V = (\mathbb{F}_2)^r$ with $r = mb$, $b \geq 2$. The vector space V is a direct sum

$$V = V_1 \oplus \cdots \oplus V_b,$$

where each V_i has the same dimension m (over $\mathbb{F}_2$). For any $v \in V$, we will write $v = v_1 \oplus \cdots \oplus v_b$, where $v_i \in V_i$. Also, we consider the projections $\pi_i : V \to V_i$ mapping $v \mapsto v_i$.

Any $\gamma \in \mathrm{Sym}(V)$ that acts as $v\gamma = v_1\gamma_1 \oplus \cdots \oplus v_b\gamma_b$, for some $\gamma_i \in \mathrm{Sym}(V_i)$, is a bricklayer transformation (a "parallel map") and any γ_i is a brick. The maps γ_i's are traditionally called S-boxes and map γ is called a "parallel S-box". A linear (or affine) map $\lambda : V \to V$ is traditionally called a "Mixing Layer" when used in composition with parallel maps. We denote by σ_v a translation over V.

Definition 1.2.4. *A linear map $\lambda \in \mathrm{GL}(V)$ is a* proper mixing layer *if no sum of some of the V_i (except $\{0\}$ and V) is invariant under λ.*

We can characterize the "translation based" class by the following

Definition 1.2.5. *We say that $\mathcal{C}$ is* translation based (tb) *if:*

- *it is the composition of a finite number of rounds, such that any round τ_k can be written[4] as $\gamma\lambda\sigma_{\bar{k}}$, where*
  - *γ is a round-dependent bricklayer transformation (but it does not depend on k),*
  - *λ is a round-dependent linear map (but it does not depend on k),*
  - *$\bar{k}$ is in V and depends on both k and the round ($\bar{k}$ is called a "round key");*
- *for at least one round we have (at the same time) that λ is proper and that the map $\mathcal{K} \to V$, $k \mapsto \bar{k}$, is surjective (a "proper" round).*

In [CDVSar] the authors gave several non-trivial remarks that can be useful. Let us recall the principal ones.

Remark 1.2.6. A generalization is obtained by allowing a key-independent permutation at the beginning and/or another at the end. This is the case for example for the SERPENT cipher. Since these permutations have no influence on the cryptanalysis of a cipher, they can be ignored.

Remark 1.2.7. A round consisting of only a translation is still acceptable, by assuming $\gamma = \lambda = 1_V$ (the identity map on V), although obviously it is not proper. Indeed, we can always assume that the first round is of this kind, otherwise we can remove its γ and λ (Remark 1.2.6). Then, we can also assume that $0\gamma = 0$, since we can add 0γ to the round key of the previous round (if the previous round is proper, it remains proper since $\sigma_{0\gamma}$ is a permutation over V).

Remark 1.2.8. To allow affine mixing layers, rather than linear mixing layers, seems a generalization. However, this case is indeed already present in Definition 1.2.5, since it is enough to change σ_v to incorporate the "translation part" of the mixing layer.

[4] *we drop round indexes.*

Remark 1.2.9. A generalization can be obtained by only requiring *at least one* of the rounds to be of the prescribed form (with a proper mixing layer). Although the authors' results still hold in this more general case, we do not know any interesting cipher of this kind.

Note that some famous ciphers, such as the DES, KASUMI and IDEA ciphers, cannot be seen easily as tb ciphers. Some of them (e.g. DES and KASUMI) are of *Feistel* type. They modify only one half of the cipher state in each round. It has been suggested that the Feistel ciphers suffer from a slow speed of diffusion compared to SPN (or *key-iterated*) ciphers.

1.2.1 Perfect secrecy

The concept of *perfect secrecy* (or *unconditional security*) has been formalized several decades ago by Shannon in [Sha49]. The *perfect ciphers* (for instance, the One Time Pad) are ciphers with a very strong model because one assumes that Eve's computational power is infinite. They are impractical for a real use, as they require at least as many key bits as the message length.
We are going to give a mathematical definition of *perfect ciphers*.
Let $\mathcal{P}$ be the plaintext space, $\mathcal{C}$ be the ciphertext space[5] and we assume that a particular key $\bar{k} \in \mathcal{K}$ is used for only one encryption $\phi_{\bar{k}}$. Suppose that there exists a probability distribution on $\mathcal{P}$. Let X be the random variable defined by the plaintexts and we denote by $Pr[X = x]$ the probability that the plaintext x occurs. Let Y be the random variable defined by the ciphertexts and we denote by $Pr[Y = y]$ the probability that the ciphertext y occurs. We assume that Alice and Bob have chosen the key $\bar{k}$ using some fixed probability distribution and we denote by $Pr[K = \bar{k}]$ the probability that the key $\bar{k}$ is chosen. We observe that the key $k \in \mathcal{K}$ is often chosen at random. This guarantees that all the keys are equiprobable, which is what we really need, but the random choice *per se* is irrelevant in the model we are describing. Since Alice and Bob agree on the keys before Alice knows her plaintext, we can assume that key and plaintext are independent random variables. Moreover, the two probability distributions on $\mathcal{P}$ and $\mathcal{K}$ induce a probability distribution on $\mathcal{C}$, and we consider $Pr[Y = y]$ where $y = \phi_{\bar{k}}(x)$.

Definition 1.2.10. *A cryptosystem is said to have the property of* perfect secrecy *if, for all $x \in \mathcal{P}$ and $y \in \mathcal{C}$, the two probability distributions satisfy*

$$Pr[X = x | Y = y] = Pr[X = x].$$

[5]We note that, only in this subsection, the plaintext space and the ciphertest space are not necessarily the same space.

Perfect secrecy means that the *a posteriori* distribution of the plaintext x after viewing the ciphertext y is identical to the *a priori* distribution of the plaintext. In other words, it means that Eve learns nothing more about the plaintext after having viewed the ciphertext than she knew before.

Let us consider a perfect cipher. Let $\bar{x}$ be a fixed plaintext in $\mathcal{P}$. For each $y \in \mathcal{C}$, the probability $Pr[\bar{x}|y] = Pr[y]$ is positive and so there must be at least one key k such that $\phi_k(\bar{x}) = y$. Hence $|\mathcal{K}| \geq |\mathcal{C}|$. Since the encoding function is injective, we have $|\mathcal{C}| \geq |\mathcal{P}|$ and so $|\mathcal{K}| \geq |\mathcal{P}|$. In other words, in a perfect cipher the key must be at least as large as the plaintext. Shannon gave a characterization of perfect secrecy, in case $|\mathcal{K}| = |\mathcal{C}| = |\mathcal{P}|$, as follows

Theorem 1.2.11. *Suppose that $|\mathcal{P}| = |\mathcal{C}| = |\mathcal{K}|$. A cryptosystem provides perfect secrecy if and only if every key is used with equal probability $1/|\mathcal{K}|$ and, for every $x \in \mathcal{P}$ and $y \in \mathcal{C}$, there is a unique key $\bar{k}$ such that $\phi_{\bar{k}} = y$.*

Remark 1.2.12. We have restricted our attention to the particular case in which a key k is used for only one encryption. In order to tell something about "perfect secrecy" when more and more plaintexts are encrypted using the same key k, Shannon used the concept of *entropy.* The reader can see this kind of description in [Sha49].

Remark 1.2.13. Theorem 1.2.11 can be rephrased in group theory notations as
Theorem: *Suppose that $|\mathcal{P}| = |\mathcal{C}| = |\mathcal{K}|$. A cryptosystem provides perfect secrecy if and only if every key is used with equal probability $1/|\mathcal{K}|$ and the action of $\{\phi_{\bar{k}}\}_{\bar{k}\in\mathcal{K}}$ on $\mathcal{P} = \mathcal{C}$ is a regular action.*

1.2.2 What do we mean by a "good" Block Cipher?

Up to now, there is no received definition of "good block cipher", but there are several criteria that contribute to the evaluation of a cipher. We list some of them.
Security
The most important criterion in the evaluation of a block cipher consists of estimating its *security level.* Obviously, the security of a block cipher is highly dependent on the properties of the different components:

- substitution layer consisting of a number of highly non-linear S-boxes (which are Boolean functions, see [Carar]),

- affine or linear invertible transformations.

However, there is no mathematical method to prove the security of a given block cipher, although it is sometimes possible to prove the insecurity of such a cipher. What usually happens is that a relative measure of the security of a block cipher (for instance the *K-security* in [DR02]) is given. Some necessary requests on the ciphers are made and it is a very hard problem to determine the sufficient conditions that guarantee the security. To evaluate the security, an additional concept is often considered: *practical security*. According to this concept, a block cipher is considered secure if the best-known attack requires too many resources by a suitable and acceptable margin. One can test the block cipher with different known attacks and assign a certain security level to it. Obviously, it is impossible to predict the security of the underlying block cipher with respect to yet unknown attacks.

Remark 1.2.14. Some authors believe that also the concept of *historical security* should be taken into consideration when assessing a cipher security. This is derived according to the amount of cryptanalytic work on the ciphers performed over the years. An old block cipher which has resisted to all cryptanalytical attacks for a long time will inevitably inspire a larger security feeling than a new block cipher which has not been extensively cryptanalyzed.

Efficiency

It refers to the amount of resources required to perform ϕ or ψ. In fact, in software implementations the speed of ϕ/ψ and the required amount of working memory/memory storage are relevant.
When quoting the speed of a cipher, one often makes the silent assumption that a large amount of data is encrypted with the same key. In that case, the key schedule can be neglected. However, if a cipher key is used to secure only a few messages, the amount of cycles taken by the computation of the key-schedule becomes important. The ability to efficiently change keys is called *key agility*.
Block ciphers are often used to encrypt large amounts of data; this makes data throughput an important evaluation criterion as well. One often differentiates hardware and software cases, the speed of the algorithm setup, the key setup, a key change and the encryption and decryption operations.

Flexibility

An expected important property of a block cipher is that it offers a large flexibility. For instance, a flexible algorithm may offer several possible block and key sizes, allowing to tailor an instance of the block cipher to precise external requirements. Another flexibility form concerns implementation issues. Finally, a block cipher can be used as a building block in various cryptographic constructions (like a hash function, an authentication code, or a stream cipher); if it offers an acceptable security level in all of these situations, then one can consider that it is a flexible block cipher.

Some authors (for instance see [DR02]) claim that other design criteria should be considered, such as the *simplicity.* A powerful tool for introducing simplicity is the symmetry.

Security and efficiency are applied by all ciphers designers. There are cases in which efficiency is neglected to obtain a higher security margin. The challenge is to come up with a cipher design that offers a reasonable security margin while optimizing efficiency. Flexibility is not felt as necessary as the others, since in some cases the cipher is meant for a particular application and will be implemented on a specific platform.

1.2.3 Cryptanalytic scenarios

Traditionally, the goal of Eve consists of recovering the plaintext or even the key. According to the possibilities and the capabilities of Eve, we can classify the different modes of attack (from the most practical to the most hypothetical, or equivalently, from the least powerful to the most powerful) as follows:

- *Ciphertext-only:* Eve tries to deduce some information about the key (or about the plaintext) starting from the sole knowledge of several ciphertexts and, usually, assuming some properties about the distribution of the plaintexts. This is a very unlikely scenario for modern block ciphers.

- *Known-plaintext:* in this kind of attack, we assume that Eve knows a certain amount of (plaintext,ciphertext) pairs in order to recover the key. This is a realistic scenario, where Eve can observe encrypted version of well-known data and, for instance, exploit the fact that messages often have a lot of redundancy. Linear cryptanalysis [Mat93] is a typical example of such an attack.

- *Chosen-plaintext or ciphertext:* when performing this kind of attack, Eve is able to choose plaintexts and obtain the corresponding ciphertexts. Subsequently, Eve uses any information deduced in order to recover either the key, or plaintexts corresponding to previously unseen ciphertexts. A typical example is differential cryptanalysis [AC08].

- *Adaptive chosen-plaintext or ciphertext:* such an attack consists of a chosen-plaintext (or chosen-ciphertext) attack wherein the choice of the plaintext (or ciphertext) depends on the information learned during the attack.

- *Combined chosen-plaintext and chosen-ciphertext:* this is a powerful type of adaptive attack which assumes that Eve can encrypt and decrypt arbitrary messages as she desires. A typical example of such an attack is Wagner's boomerang attack (see [Wag99], or Section 3.2.5).

- *Related-key:* in this model, Eve knows (or can choose) additionally some mathematical relations between the keys used for encryption, but not their values. This is usually employed in conjunction with some of the scenarios above. Even if in itself this attack may not be considered to be a practical threat against a block cipher (because it lives in a too strong threat model), it may be practical when a block cipher is used as a primitive for a hash function.

By considering one of the attacks described above and according to the type of information recovered during it, the possible outcomes of an attack could be classified as follows. We describe only the main outcomes from the least favorable for Eve to the most favorable. (For more details, see e.g. Knudsen [Knu99]).

- *Distinguishing attack:* Eve is able to tell whether the attacked block cipher is a permutation (chosen uniformly at random from the set of all permutations) or one of the permutations $\{\phi_k\}_{k\in\mathcal{K}}$. Infact, most modern block ciphers are designed to model a random permutation. Even if distinguishing attacks are considered as the least serious threat in practice, they often indicate some structural weaknesses of the cipher and they might be transformed into a *Key recovery* (or a *Global deduction*).

- *Local deduction:* Eve finds the plaintext (or ciphertext) of an intercepted ciphertext (or plaintext) which she did not obtain from the legitimate sender. If the number of likely plaintexts (or ciphertexts) is small, such an attack may be fatal for the cryptosystem.

- *Partial Key Recovery:* Eve is able to get some information on the key k (e.g. some relations, some bits, ...). An efficient partial key recovery is very undesirable because it could be used to determine the remaining bits of the key.

- *Global deduction:* Eve finds an algorithm functionally equivalent to ϕ or ψ, without knowing the actual value of the key k. For instance, a possibility of global deduction is that an attack is able to recover the round subkeys but not the key.

- *Key recovery (Total break):* Eve is able to recover (or reconstruct) the secret key $k \in \mathcal{K}$, thus reaching the highest goal of the attacker.

A modern cipher is considered *totally secure* if it can withstand all chosen plaintext attacks, including distinguishing attack. It is only considered *secure* if it can withstand all chosen plaintext attacks, except possibly distinguishing attacks. However, distinguishing attacks might be transformed into a key-recovery attack.

The security of a cipher against the types of attack described above is in practice measured by several additional parameters that are necessary:

- *time complexity:* it measures the computational processing required to perform an attack, i.e. it is closely related to the input. Usually, the choice of the computational unit is done to compare the attack with an exhaustive key search.
- *data complexity:* it is the number of data (like ciphertexts, (known/chosen)-plaintext, ...) required to perform an attack, according to a specific model.
- *success probability:* it measures the frequency at which the attack is successful when repeated a certain number of times in a statistically independent way.
- *memory complexity:* it measures the amount of memory units necessary to store pre-computed/obtained data necessary to perform the attack.

Usually, an attack is considered to be successful (and the attacked block cipher is considered to be broken) if the *time complexity* is significantly smaller than 2^ℓ evaluations of the block ciphers, with $\mathcal{K} = (\mathbb{F}_2)^\ell$. A block cipher is considered to be partially broken if some of the plaintext bits can be discovered in time faster than an exhaustive search. Moreover, a block cipher can be completely characterized if, using a fixed key k, the encryption via ϕ_k of all 2^r plaintexts is available. This puts an upper bound on the *data complexity*. Quoting NESSIE's final security report [CGC03],
"A block cipher is considered secure if no attack requires both time and data complexity significantly less than 2^ℓ and 2^r, respectively."

Exhaustive key search

One of the simplest way to attack a block cipher consists in trying one key after the other until the right one is found. Typically, for a block cipher with key-size ℓ and a block-size r, and provided that a very small number of known plaintext-ciphertext pairs (slightly more than $\lceil \frac{\ell}{r} \rceil$) are encrypted using the same key k, it is possible to recover the key k by exhaustive search. In the worst case, this operation has time complexity equal to 2^ℓ evaluations and an average time complexity of $2^{\ell-1}$. Moreover, if the plaintext space is known to contain some redundancy, then one can even consider a ciphertext-only exhaustive search. The success probability of an exhaustive key search is equal to the fraction of the key space searched; for instance, if one searches one tenth of the key space, then one has roughly a 10% probability to succeed. Therefore, a fixed key of size ℓ defines an upper bound on the security of a block cipher. Thus, for any secure block cipher, ℓ should be large enough to prevent exhaustive key search attacks.

As it is often difficult (or it may even be impossible) to exhibit an attack against the full version of an iterative block cipher, another common mean to assess its security consists in taking into account the maximal number of rounds for which an attack is known. This can give some feeling about the security margin of such a block cipher. For instance, we summarize in Section 3.2 the currently best known attacks on various reduced-round versions of AES.

Remark 1.2.15. There exist attacks against block ciphers that can be applied without attacking the internal structure of the cipher: the *Black-box attacks*. These are attacks which treat the block cipher as a black box taking plaintexts and keys in input and outputting ciphertexts; their complexity depends only on parameters like the key length ℓ and the block length r of the block ciphers under consideration. Note that these attacks include, for instance, the exhaustive key search.

1.3 Cryptographic hash functions

Cryptographic hash functions are a useful building block for several cryptographic applications. The most important are the protection of information authentication and digital signatures. Such functions are also used to construct pseudo-random number generators. Hash functions appeared in cryptographic literature when it was realized that the encryption of information is not sufficient to protect its authenticity. They are functions that map (compress) an input of arbitrary length to a result string with fixed length, the *hashcode*. If these mappings satisfy some additional cryptographic conditions, they are a very powerful tool in the design of techniques to protect the integrity of information.
The most commonly used hash functions are MD5 [Riv92], designed by Ronald Rivest, and SHA-1 , designed by the National Security Agency (NSA). In practice, a hash function is a fixed function that maps arbitrary strings into binary strings of fixed length. In theory, we usually consider (keyed) hash functions, as in the following definition:

Definition 1.3.1. *A* hash family *is the tuple* $(\mathcal{M}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$*, where the following conditions hold:*

1. *$\mathcal{M}$ is the set of possible messages;*

2. *$\mathcal{Y}$ is the set of possible hash values or authentication tags;*

3. *$\mathcal{K}$ is the key space (the finite set of all possible keys);*

4. *for any $k \in \mathcal{K}$, there is a* hash function *$h_k \in \mathcal{H}$, were $h_k : \mathcal{M} \to \mathcal{Y}$.*

In the previous definition, the set $\mathcal{M}$ could be finite or infinite but $\mathcal{Y}$ is always a finite set. If $\mathcal{M}$ is a finite set, a hash function is sometimes called a *compression function* and we will always assume that $|\mathcal{M}| \geq |\mathcal{Y}|$. A pair $(x, y) \in \mathcal{M} \times \mathcal{Y}$ is said to be a *valid pair*, under the key k, if $h_k(x) = y$.
In some cryptographic application, it is desirable that the hash function is a one-way function:

Definition 1.3.2. *A Hash function h is* one-way *if, for random key k and an n-bit string y, it is hard for the attacker presented with k,y to find x so that $h_k(x) = y$.*

Remark 1.3.3. The rigor of this definition is questioned and no one way functions has been found.

If a hash function is to be considered secure, it should be the case that the following three problems are difficult to solve.
Preimage: given $y \in \mathcal{Y}$, to find an $x \in \mathcal{M}$ such that $h(x) = y$.
Second Preimage: given $x \in \mathcal{M}$, to find an $x' \in \mathcal{M}$ such that $x' \neq x$ and $h(x') = h(x)$.
Collision: to find $x, x' \in \mathcal{M}$ such that $x' \neq x$ and $h(x') = h(x)$.

It is easy to see that Collision resistance implies Second Preimage resistance. The Second Preimage resistance and one-wayness are incomparable (the properties do not follow from one another), although construction which are one-way but not Second Preimage resistant are quite contrived. In practice, Collision resistance is the strongest property of all three, hardest to satisfy and easiest to breach, and breaking it is the goal of most attacks on hash functions.

Hash function based on a block cipher

Two arguments can be indicated for designers of cryptographically secure hash functions to base their schemes on existing encryption algorithms. The first argument is the minimization of the design and implementation effort: hash functions and block ciphers that are both efficient and secure are hard to design. Moreover, existing software and hardware implementations can be reused, which will decrease the cost. The main advantage is that the trust in existing encryption algorithms can be transferred to a hash function. Moreover, a limited number of design principles for encryption algorithms are also valid for hash functions. The main disadvantage of this approach is that dedicated hash functions are likely to be more efficient. Finally we note that block ciphers may exhibit some weaknesses that can be exploited only if used in a hashing mode.

1.4 Statistical tests

When a statistical test on data from a cryptographic algorithm is performed, we wish to test whether the data "seem" random or not. It seems impossible to design a test that gives decisive answer. However, there are many different properties of randomness and non-randomness, and it is possible to design tests for these specific properties, as we are going to explain in this section.

There are two basic types of generators used to produce random sequences: random number generators and pseudo-random number generators. For cryptographic applications, both types produce a stream of zeros and ones that may be divided into sub-streams or blocks of random numbers. If a pseudo-random sequence is properly constructed, each value in the sequence is produced from the previous value via transformations which appear to introduce additional randomness. A series of such transformations can eliminate statistical autocorrelations between input and output.

Typically the random properties of binary sequences to be tested are the following:

- Uniformity: at any point in the generation of a sequence of bits, the occurrence of a zero or one is equally likely, i.e., the probability of each is exactly 1/2. The expected number of zeros (or ones) is $n/2$, where n is the sequence length.

- Scalability: Any test applicable to a sequence can also be applied to subsequences extracted at random. If a sequence is random, then any such extracted subsequence should also be random. Hence, any extracted subsequence should pass any test for randomness.

- Consistency: The behavior of a generator must be consistent across starting values (seeds). It is inadequate to test a pseudo-random number generator based on the output from a single seed, or a random number generators on the basis of an output produced from a single physical output.

Although there are many tests for disproving the randomness of a sequence, no specific finite set of tests is deemed "complete." We focus on the statistical testing that the NIST has conducted on the AES candidate algorithms to evaluate their suitability as random number generators. The NIST Test Suite [NIS00] is a statistical package consisting of 16 tests that were developed to test the randomness of (arbitrarily long) binary sequences produced by either hardware or software based cryptographic random or pseudorandom number generators. These tests focus on a variety of different types of non-randomness that could exist in a sequence. We give a sketch (see also [Sot98]) of the objective of sixteen such tests.

1. *The Frequency (Monobit) Test*: it determines whether the number of ones and zeros in a sequence are "approximately" the same as it would be expected for a truly random sequence. All subsequent tests are conditioned on having passed this first basic test.

2. *Frequency Test within a Block*: it determines whether the frequency of m-bit blocks in a sequence appears as often as would be expected for a truly random sequence; the frequency of ones in an m-bit block should be approximately $m/2$.

3. *The Runs Test*: a run of length k consists of exactly k identical bits and is bounded before and after with a bit of the opposite value. The purpose of the runs test is to determine whether the number of runs of ones and zeros of various lengths is as expected for a random sequence. In particular, this test determines whether the oscillation between such zeros and ones is too fast or too slow.

4. *Test for the Longest-Run-of-Ones in a Block*: it determines whether the distribution of long runs of ones agrees with the theoretical probabilities. Note that an irregularity in the expected length of the longest run of ones implies that there is also an irregularity in the expected length of the longest run of zeros. Therefore, only a test for ones is necessary.

5. *The Binary Matrix Rank Test*[6]: it determines whether the distribution of the rank of (32×32) bit matrices, constructed with bits coming from the sequence, agrees with the theoretical probabilities.

6. *The Discrete Fourier Transform (Spectral) Test*: it determines whether the spectral frequency of the binary sequence agrees with what would be expected for a truly random sequence.

7. *The Non-overlapping Template Matching Test*: it determines whether the number of occurrences for a specified non-periodic template agrees with the number expected for a truly random sequence.

8. *The Overlapping Template Matching Test*: it determines whether the number of occurrences for a template of all ones agrees with what is expected for a truly random sequence.

9. *Maurer's "Universal Statistical" Test*: it determines whether a binary sequence does not compress beyond what is expected of a truly random sequence.

[6]We will use a variation of this test in Chapter 5.

10. *The Lempel-Ziv Compression Test*: the focus of this test is the number of cumulatively distinct patterns (words) in the sequence. It determines how far the tested sequence can be compressed with the Lempel-Ziv algorithm. The sequence is considered to be non-random if it can be significantly compressed. A random sequence will have a characteristic number of distinct patterns.

11. *The Linear Complexity Test*: it determines whether or not the sequence is complex enough to be considered random.

12. *The Serial Test*: it determines whether the number of occurrences of the 2^m m-bit overlapping patterns is approximately the same as would be expected for a random sequence. Random sequences have uniformity; that is, every m-bit pattern has the same chance of appearing as every other m-bit pattern.

13. *The Approximate Entropy Test*: it compares the frequency of overlapping blocks of two consecutive/adjacent lengths (m and $m + 1$) against the expected result for a normally distributed sequence. It determines whether a sequence appears more regular than is expected from a truly random sequence.

14. *The Cumulative Sums Test*: it determines whether the maximum of the cumulative sums in a sequence is too large or too small; indicative of too many ones or zeros in the early (late) stages.

15. *The Random Excursions Test*: it examines the number of cycles within a sequence and determine whether the number of visits to a given state, $[-4, -1]$ and $[1, 4]$, exceeds the expected for a truly random sequence.

16. *The Random Excursions Variant Test*: it determines if the total number of visits to states between $[-9, -1]$ and $[1, 9]$ exceeds the expected for a truly random sequence.

These tests may be useful as a first step in determining whether or not a generator is suitable for a particular cryptographic application. However, no set of statistical tests can absolutely certify a generator as appropriate for usage in a particular application, i.e., statistical testing cannot serve as a substitute for cryptanalysis. In terms of testing encryption algorithms, these two errors can be described as follows:

- Type I Error: The statistical test classifies a “good” encryption algorithm as “bad”.

- Type II Error: The statistical test classifies a “bad” encryption algorithm as “good”.

Using the previous tests, the NIST analyzed nine different Categories of Data:

1. 128-Bit Key Avalanche;
2. Plaintext Avalanche;
3. Plaintext/Ciphertext Correlation;
4. Cipher Block Chaining Mode;
5. Random Plaintext/Random 128-Bit Keys;
6. Low Density Plaintext;
7. Low Density 128-Bit Keys;
8. High Density Plaintext;
9. High Density 128-Bit Keys;

Based on the version of the NIST statistical tests we are considering, each of the AES candidates algorithms was evaluated [Sot98]. Those algorithms that did not demonstrate deviation from randomness include CAST-256, DFC, E2, LOKI-97, MAGENTA, MARS, Rijndael, SAFER+, and SERPENT. The remaining algorithms (CRYPTON, DEAL, FROG, HPC, RC6 and TWOFISH) appeared to have displayed deviation from randomness.

The results suggest that data flagged as non-random for the TWOFISH and RC6 algorithms should be treated as statistical anomalies. Similarly, due to the natural filtering process for the random excursion test, small sample sizes may incorrectly lead one to commit a type I error.

A description of AES, SERPENT and PRESENT

In this chapter we describe three well-known iterated (algebraic) block ciphers: Rijndael, SERPENT and PRESENT. They belong to all the three classes of iterated ciphers described in the previous chapter. In fact they satisfy both the SPN structure (with a slight change in the last round), the "key-iterated block cipher" structure and the "translation based" approach. SERPENT and PRESENT are so similar to AES that people talk loosely of "AES-like ciphers". In the following section we propose, respectively, the encryption of Rijndael, SERPENT and PRESENT according to the definition of "translation based" (Section 1.2.5).

Rijndael and SERPENT were designed as candidates for the Advanced Encryption Standard (AES) competition. They were two of the five finalists (joint with MARS [BCD$^+$98], RC6 [RRY00], and Twofish [Sch98]) and were all felt to be secure. All the AES candidates were evaluated for their suitability according to criteria as security, cost, properties of the algorithm and the corresponding implementation. *Security* of the proposed algorithms was claimed essential; in fact any algorithm found insecure would not be considered any further. *Cost* refers to the computational efficiency (in particular, speed and memory requirements) of various types of implementations, including software, hardware and smart cards. Among other factors, *algorithm* and *implementation characteristics* include flexibility and algorithm simplicity. We refer the reader to Section 1.2.2 for a description of all these criteria.

Rijndael [DR98], designed by Daemen and Rijmen, was chosen to be the Advanced Encryption Standard (AES) because its combination of security, performance, efficiency, implementability and flexibility was judged to be superior to the other finalists. The AES (Rijndael) is secure against all previously-known attacks. Various aspects of its design incorporate specific features that help provide security against specific attacks. For example, the use of the finite field inversion operation in the construction of the S-box yields linear approximation and difference distribution tables in which the entries are close to uniform. This provides security against differential and linear attacks. The linear transformation, makes it impossible to find *differential* and *linear* attacks that involve few active S-boxes.

There are three variants of the AES: AES-128, AES-192, AES-256. They are similar, but different in some details:

- **AES-128** has a 128-bit key and uses 10 rounds,
- **AES-192** has a 192-bit key and uses 12 rounds,
- **AES-256** has a 256-bit key and uses 14 rounds.

There are (non-practical) attacks on the full AES-192 and the full AES-256, but there are apparently no known attacks on (the full) AES-128 (faster than exhaustive search). We will mainly consider only the AES-128 and so, from now, we will write AES instead of Rijndael cryptosystem with a 128-bit key. The best attacks on the AES are applied to small scale variants of the cipher in which the number of rounds (or the cipher size) is reduced (see Section 2.1.4 and Section 3.2).

SERPENT [BAK98] was designed by Ross Anderson, Eli Biham and Lars Knudsen. It was widely viewed as taking a more conservative approach to security than the other AES finalists, opting for a larger security margin. For example, the designers deemed 16 rounds to be sufficient against known attacks, but they specified 32 rounds as insurance against possible future discoveries in cryptanalysis. Initially Anderson, Biham and Knudsen decided to use S-boxes from DES in a new structure optimized for efficient implementation on modern processor, designing an algorithm (known as Serpent 0) that was as fast as DES and apparently more secure than three key DES (triple DES). Then they selected new (presumably stronger S-boxes) and changed the key schedule slightly, obtaining what now is called SERPENT.

PRESENT [ABKL$^+$07] was proposed by Bogdanov et al. at CHES 2007 conference as an ultra-lightweight block cipher, suitable for RFIDs and similar devices. The authors claim that, besides security and efficient implementation, the main goal when designed PRESENT was simplicity (see Section 1.2.2). Moreover, another goal that they had in mind was to design an ultra-lightweight block cipher that offers a level of security commensurate with a 64-bit block size and an 80-bit key.

2.1 The AES cryptosystem

Let $\mathcal{M} = \mathcal{K} = V = (\mathbb{F}_2)^r$ with $r = 128$ and let $x \in \mathcal{M}$ be our plaintext, $k \in \mathcal{K}$ our random key and $y = \phi_k(x)$ the corresponding ciphertext. Before describing the individual components γ, λ and σ_k of the *round function*, we recall (see Section 1.1) that it is possible to identify $(\mathbb{F}_2)^8$ with the field $\mathbb{F}_{2^8}$, via the quotient map $\mathbb{F}_{2^8} \leftrightarrow \mathbb{F}_2[x]/\langle \mathsf{m} \rangle$, where $\mathsf{m} \in \mathbb{F}_2[x]$ is an irreducible polynomial such that $\deg(\mathsf{m}) = 8$. The irreducible (but not primitive) AES polynomial is $\mathsf{m} = x^8 + x^4 + x^3 + x + 1$.

It is also useful to recall a particular concept that is inherent in the structure of the AES: the State. Internally, the AES algorithm's operations are performed on a two-dimensional array of bytes, called the State. It consists of 4 rows and 4 columns and each element of this matrix is one byte (i.e. an element of $\mathbb{F}_{2^8} = \mathbb{F}_{256}$).
At the start of the encryption process, the input x (the plaintext) is a vector in V and it is first changed into a 16-byte vector:

$$\nu : (\mathbb{F}_2)^{128} \longrightarrow (\mathbb{F}_{256})^{16}, \quad \mathsf{x} \mapsto \mathsf{y}.$$

Then its 16 bytes are “rolled down” to the State.

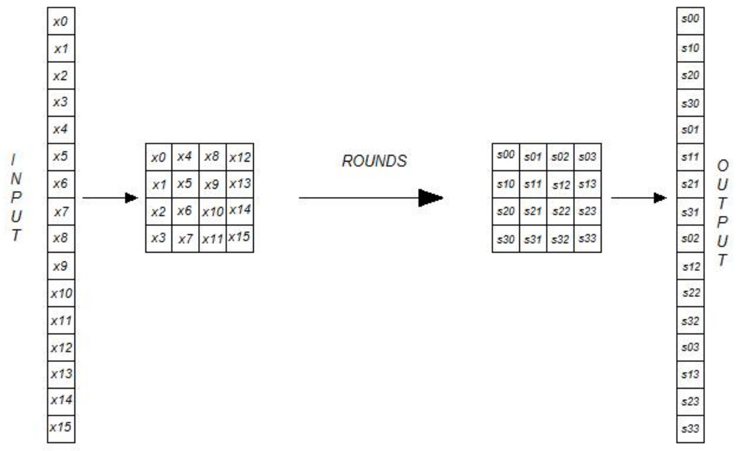


Figure 2.1: Wrapping and unwrapping the State

Each round performs its operations on the State and after the last round the State is “unwrapped” and “fills up” the output vector.

A preliminary translation $\sigma_{k^{(0)}}$, where $k^{(0)} \in (\mathbb{F}_2)^r$ is the first round key, is applied to the plaintext to form the input to the (`Round 1`). It means that we can consider a preliminary round (`Round 0`) such that $\gamma = 1_V$ and $\lambda = 1_V$ (see Remark 1.2.7).
In order to obtain the ciphertext, other $N = 10$ rounds follow.

Let $1 \leq \rho \leq N-1$. A typical round (`Round` ρ) can be written as the composition[1] $\gamma\lambda\sigma_{k^{(\rho)}}$, where

- the parallel map γ is called SubBytes and it works in parallel to each of the 16 bytes of the data;
- the affine map λ is the composition of two operations known as ShiftRows and MixColumns;
- $\sigma_{k^{(\rho)}}$ is the translation with the session key $k^{(\rho)}$ (this operation is called AddRoundKey).

The last round (`Round` N) is atypical and is characterized by $\gamma\bar{\lambda}\sigma_{k^{(N)}}$ where the affine map $\bar{\lambda}$ is only made by the ShiftRows operation. So we obtain our ciphertext $\mathsf{y} = \phi_k(\mathsf{x})$.
In the following, we analyze the structure of each component of the round function.

2.1.1 SubBytes

The vector space V is the direct sum $V = V_1 \oplus \cdots \oplus V_{16}$ where each $V_i = (\mathbb{F}_2)^8$ $(1 \leq i \leq 16)$. Any parallel map $\gamma \in \mathrm{Sym}(V)$ acts on an element $v \in V$ as $v\gamma = v_1\gamma_1 \oplus \ldots \oplus v_{16}\gamma_{16}$, where $v_i \in V_i$ and $\gamma_i \in \mathrm{Sym}(V_i)$. The SubBytes operation γ is composed by two transformations: the inversion in $\mathbb{F}_{2^8}$ and an affine transformation. The *inversion operation* is the *patched inversion*[2] in $\mathbb{F}_{2^8}$ (i.e. $\varphi(x) = x^{254}$).
The *affine transformation* over $\mathbb{F}_2$ consists of a linear mapping $\xi : (\mathbb{F}_2)^8 \to (\mathbb{F}_2)^8$, specified by an 8×8 circulant matrix over $\mathbb{F}_2$, plus a translation. The result of inversion is regarded as a vector in $(\mathbb{F}_2)^8$ and the output is given by $y = \xi(x)$, where

$$\begin{pmatrix} y_7 \\ y_6 \\ y_5 \\ y_4 \\ y_3 \\ y_2 \\ y_1 \\ y_0 \end{pmatrix} = \begin{pmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{pmatrix} \begin{pmatrix} x_7 \\ x_6 \\ x_5 \\ x_4 \\ x_3 \\ x_2 \\ x_1 \\ x_0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

[1]Note that the order of the operation is exactly: γ, λ, and then σ_k.

[2]Since the AES consists of 10 rounds and each round requires 16 S-box computations, the probability of there being no 0-inversions during an encryption is $(255/256)^{160} \approx 0.53$.

Remark 2.1.1. The inversion resists standard cryptanalysis, while the other components in the S-box are meant to disguise its algebraic simplicity and to provide a complicated algebraic expression if combined with the inverse mapping. This should provide resistance to interpolation and similar attacks. Furthermore, the S-box constants was chosen is such a way that the S-box has no fixed points and no opposite fixed points. (See Section 3.5)

2.1.2 Mixing Layer

The map $\lambda : V \to V$ is a composition of two linear operations: ShiftRows and MixColumns. The ShiftRows operation is performed as follows. Any byte (an element of $\mathbb{F}_{2^8}$) in row i of the State, where $0 \le i \le 3$, is cyclically shifted (towards left) by i positions, as follows:

$$\begin{array}{|c|c|c|c|} \hline s_0 & s_4 & s_8 & s_{12} \\ \hline s_1 & s_5 & s_9 & s_{13} \\ \hline s_2 & s_6 & s_{10} & s_{14} \\ \hline s_3 & s_7 & s_{11} & s_{15} \\ \hline \end{array} \quad \to \quad \text{ShiftRows} \quad \to \quad \begin{array}{|c|c|c|c|} \hline s_0 & s_4 & s_8 & s_{12} \\ \hline s_5 & s_9 & s_{13} & s_1 \\ \hline s_{10} & s_{14} & s_2 & s_6 \\ \hline s_{15} & s_3 & s_7 & s_{11} \\ \hline \end{array}$$

In other words, we can describe the ShiftRows operation by the map

$$\mathsf{sh} : (\mathbb{F}_{2^8})^{16} \to (\mathbb{F}_{2^8})^{16}$$

$$(s_0, s_1, \cdots, s_{15}) \mapsto (s_0, s_5, s_{10}, s_{15}, s_4, s_9, s_{14}, s_3, s_8, s_{13}, s_2, s_7, s_{12}, s_1, s_6, s_{11}).$$

We can also represent the ShiftRows operation with the following 16×16 block diagonal matrix

$$S = \begin{pmatrix} I & 0 & 0 & 0 \\ 0 & R & 0 & 0 \\ 0 & 0 & R^2 & 0 \\ 0 & 0 & 0 & R^3 \end{pmatrix} \qquad R = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

where the matrix R is a permutation matrix over $\mathbb{F}_{2^8}$ that represents the shift of one row by one position.

In order to describe the MixColumns operation, each column of the State can be treated as a four-term polynomial in $\mathbb{F}_{256}[z]$. Let $c(z)$ be one such polynomial. Then each column is replaced by the result of the multiplication in $\mathbb{F}_{256}[z]/(z^4+1)$ by $a(z)$, $c \mapsto c \cdot a \mod (z^4+1)$,

$$(c_1, c_2, c_3, c_4) \longrightarrow (c_1 \cdot a, c_2 \cdot a, c_3 \cdot a, c_3 \cdot a)\,.$$

Note that $a(z)$ is invertible in $\mathbb{F}_{256}[z]/(z^4+1)$. On the other hand, we can see the MixColumns operation as 4-block diagonal matrix, each blocks the same MDS matrix (i.e. all minors are non-zero):

$$\begin{pmatrix} z & z+1 & 1 & 1 \\ 1 & z & z+1 & 1 \\ 1 & 1 & z & z+1 \\ z+1 & 1 & 1 & z \end{pmatrix}$$

Remark 2.1.2. This MDS property is used to ensure that the number of active S-boxes involved in a differential or linear attack increases rapidly, and the security of the AES against these particular attacks can be established.

Obviously, we can also see the whole Mixing Layer (λ linear operation) as a matrix $\mathbf{M}$. We observe that the order of this matrix is quite small, i.e. $\mathbf{M}^8 = 1$. (Also, both the order of ShiftRows and MixColumns are equal to 4.)

2.1.3 Key schedule

We need $N+1 = 11$ round keys, each of which consists of 16 bytes. The key schedule algorithm is word-oriented (a *word* consists of 4 bytes). Therefore each round key is comprised of four words. In the following figure we summarize the technique to create the round key $k^{(\rho+1)}$, starting from the round key $k^{(\rho)}$.

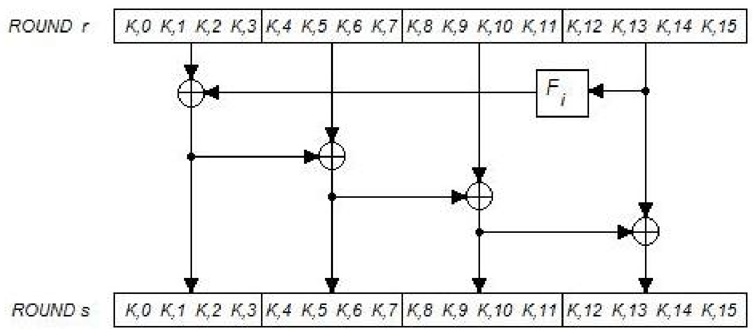


Figure 2.2: The AES Key Schedule

The non-linear function F_i consists of applying the S-box to all components of the input, a rotation of bytes and the addition of a round-specific constant. The reader can find details in [DR98].

2.1.4 Small scale variants of the AES

For most methods of cryptanalysis it is quite straightforward to perform experiments on reduced versions of the cipher to understand how the attack might perform. For new algebraic methods (see Chapter 3.3) it is difficult to design small scale versions that can replicate the main cryptographic and algebraic properties of the cipher. Still, experiments on small versions can give an idea about the behavior of algebraic cryptanalysis on block ciphers.

A family of small scale variants of the AES was proposed by Cid, Murphy and Robshaw in [CMR05]. They define two sets of small scale variants of the AES; they differ in the form of the final round. These two sets of variants will be denoted by $SR(N, r, c, e)$ and $SR'(N, r, c, e)$. Both $SR(N, r, c, e)$ and $SR'(N, r, c, e)$ have the following parameters:

- N is the number of (encryption) rounds, $1 \leq N \leq 10$;
- r is the number of rows in the rectangular arrangement of the input, $r = 1, 2, 4$;
- c is the number of columns in the rectangular arrangement of the input, $c = 1, 2, 4$;
- e is the size (in bits) of a word, $e = 4, 8$.

Both $SR(N, r, c, e)$ and $SR'(N, r, c, e)$ have N rounds and a block size of rce bits, where a data block is viewed as an array of $(r \times c)$ words of e bits. The full AES is equivalent to $SR'(10, 4, 4, 8)$.

A round of the small scale variants of the AES consists of small scale variants of these operations. For the last round of the AES, the operation MixColumns is omitted. Similarly, for $SR'(N, r, c, e)$ the final round does not use MixColumns, whereas MixColumns is retained for the final round of $SR(N, r, c, e)$. The AES is thus identical to $SR'(10, 4, 4, 8)$.
Note that the two ciphertexts produced by $SR(N, r, c, e)$ and $SR'(N, r, c, e)$ when encrypting the same plaintext under the same key are related by an affine mapping. A solution of the system of equations for one cipher would immediately give a solution for the other and so, without loss of generality, we can only consider $SR(N, r, c, e)$.

2.2 The SERPENT cryptosystem

SERPENT is a translation-based cryptosystem, like AES.
Let $\mathcal{M} = V = (\mathbb{F}_2)^r$, with $r = 128$. We consider $\mathcal{K} = (\mathbb{F}_2)^\ell$, with the fixed length $\ell = 128$, although the key is designed with variable length.
The encryption ϕ proceeds by $N = 32$ similar rounds and it works as follows:

- a preliminary permutation is applied $\pi : V \to V$ (this is not used for security, rather to ease the implementation);
- there is a preliminary translation with the first round key;
- $N-1$ rounds with the same structure are applied, but using a different permutation, each composed of a key translation σ_k, a parallel S-box γ and a linear mixing-layer λ (we denote the round ρ by `Round` ρ, with $\rho = 1, ..., 31$);
- the last round (`Round` 32) follows and it consists of the composition $\gamma\lambda\sigma_k$ where $\lambda = 1_V$;
- a final permutation $\pi^{-1} : V \to V$ is performed.

The decryption process is easily obtained by inverting every step of the encryption, using the inverse of the S-boxes, the inverse of the mixing-layer and the reverse order of the round keys.

2.2.1 A SERPENT round

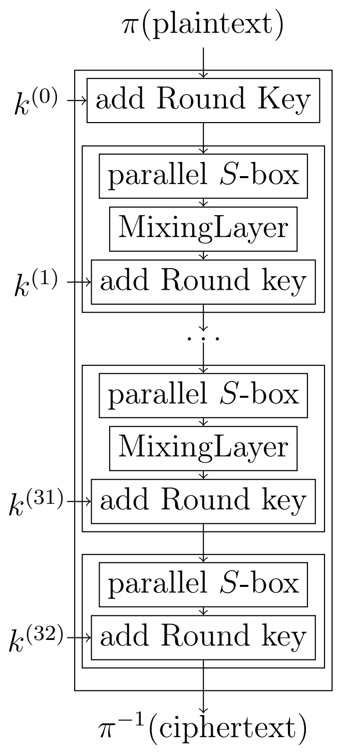
Let ρ be a natural number such that $1 \leq \rho \leq 31$. In order to describe a typical round (`Round` ρ) we have to specify how the components γ, λ and σ_k are applied. We note that, after the permutation $\pi : V \to V$, we perform a preliminary translation $\sigma_{k^{(0)}}$, where $k^{(0)} \in (\mathbb{F}_2)^r$ is the first round key.
Let $V = V_1 \oplus \cdots \oplus V_{32}$, where , for any $1 \leq j \leq 32$, each $V_j = (\mathbb{F}_2)^4$. Any $\gamma \in \mathrm{Sym}(V)$ acts as $v\gamma = v_1\gamma_1 \oplus \ldots \oplus v_{32}\gamma_{32}$, where $v_j \in V_j$ and $\gamma_j \in \mathrm{Sym}(V_j)$. We have to characterize each γ_j (i.e. we have to construct each S-box).

The S-boxes of SERPENT were built "ad hoc" starting from the 8 fixed S-boxes of DES (see Appendix) as follows. They were generated using a matrix with 32 arrays each with 16 entries. The matrix was initialized with the 32 rows of the DES S-boxes and transformed by swapping the entries in the r-th array depending both on the value of the entries in the $(r+1)$-st array and on an initial string representing a key. If the resulting array had some desired (differential and linear) properties, the array was saved as a SERPENT S-box. The procedure was repeated until the eight S-boxes $S_1, \ldots, S_8$ have been generated.

To each v_j we apply the same $S_{i \bmod 8}$ S-box, so that $S_{i \bmod 8}(v_j)$ lies in $(\mathbb{F}_2)^4$. That is, $\gamma_1 = \gamma_2 = \cdots = \gamma_{32} = S_{i \bmod 8}$.

Then the linear transformation λ (described in the Subsection 2.2.2) and a final translation $\sigma_{k^{(\rho)}}$ are applied.

The last round (`Round` 32) is only slightly different. The only difference with a typical round is the replacing of a linear transformation λ by 1_V.



2.2.2 *The Linear transformation*

The linear transformation occurs in each typical round ($1 \leq i \leq 31$) and works on $v \in V$, where V is a direct sum $V = V_1 \oplus \cdots \oplus V_4$ with $V_j = (\mathbb{F}_2)^{32}$ ($1 \leq j \leq 4$), in such a way that the input vector at the i-th round is $v = v_1 \oplus v_2 \oplus v_3 \oplus v_4$.
Starting by the initial `State` (v_1, v_2, v_3, v_4), λ the linear transformation is characterized by the following operations[3]:

- we sum to v_2 the $\mathsf{rotat}_{13}(v_1)$ and $\mathsf{rotat}_3(v_3)$, obtaining `State 1`:

$$(v_1^1, v_2^1, v_3^1, v_4^1) = (\mathsf{rotat}_{13}(v_1), v_2 + \mathsf{rotat}_{13}(v_1) + \mathsf{rotat}_3(v_3), \mathsf{rotat}_3(v_3), v_4);$$

- we sum to v_4^1 the $\mathsf{shift}_3(v_1^1)$ and v_3^1, obtaining `State 2`:

$$(v_1^2, v_2^2, v_3^2, v_4^2) = (v_1^1, v_2^1, v_3^1, \mathsf{shift}_3(v_1^1) + v_3^1 + v_4^1);$$

[3] rotat denotes a rotation of the bits and shift denotes a bit shift toward the right

- we sum to v_1^2 the $\mathsf{rotat}_1(v_2^2)$ and $\mathsf{rotat}_7(v_4^2)$, obtaining `State 3`:

$$(v_1^3, v_2^3, v_3^3, v_4^3) = (v_1^2 + \mathsf{rotat}_1(v_2^2) + \mathsf{rotat}_7(v_4^2), \mathsf{rotat}_1(v_2^2), v_3^2, \mathsf{rotat}_7(v_4^2));$$

- we sum to v_3^3 the $\mathsf{shift}_7(v_2^3)$ and v_4^3, obtaining `State 4`:

$$(v_1^4, v_2^4, v_3^4, v_4^4) = (v_1^3, v_2^2, \mathsf{shift}_7(v_2^3) + v_3^3 + v_4^3, v_4^3);$$

- we consider the $\mathsf{rotat}_5(v_1)$ and the $\mathsf{rotat}_{22}(v_3)$, obtaining `State 5`:

$$(v_1^5, v_2^5, v_3^5, v_4^5) = (\mathsf{rotat}_5(v_1^4), v_2^4, \mathsf{rotat}_{22}(v_3^4), v_4^4).$$

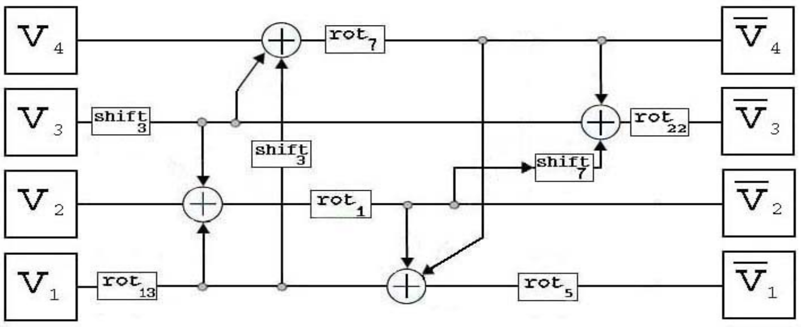


Figure 2.3: Linear transformation of SERPENT

2.2.3 The SERPENT's key schedule

The round keys of the SERPENT cipher are constructed starting from suitable "prekeys" $(w_1, \dots, w_{131})$ (for details about the "prekeys" construction, see [BAK98]). Then the authors use the S-boxes to transform the prekeys w_i into words k_i of the round key by dividing the vector of prekeys into 4 section and transforming the i-th words of each of the 4 sections using $S_{(r+3-i) \bmod r}$. In case $r = 32$, we have

$$\begin{aligned}
\{k_0, k_{33}, k_{66}, k_{99}\} &= S_3(w_0, w_{33}, w_{66}, w_{99}) \\
\{k_1, k_{34}, k_{67}, k_{100}\} &= S_2(w_1, w_{34}, w_{67}, w_{100}) \\
&\vdots \\
\{k_{31}, k_{64}, k_{97}, k_{130}\} &= S_4(w_{31}, w_{64}, w_{97}, w_{130}) \\
\{k_{32}, k_{65}, k_{98}, k_{131}\} &= S_3(w_{32}, w_{65}, w_{98}, w_{131}).
\end{aligned}$$

Then, the 32-bit values k_j are renumbered as 128-bit subkeys K_i, $(0 \le i \le r)$, as follows $K_i = \{k_{4i}, k_{4i+1}, k_{4i+2}, k_{4i+3}\}$.

2.3 PRESENT: an ultra-lightweight block cipher

PRESENT is an iterated block cipher that consists of $N = 31$ rounds.
Let $\mathcal{M} = V = (\mathbb{F}_2)^r$ with $r = 64$. Let $\mathcal{K} = (\mathbb{F}_2)^\ell$, where ℓ may be equal to 80 or 128. We consider only the PRESENT's version such that $\mathcal{K} = (\mathbb{F}_2)^{80}$, since its authors recommend it in order to have a good performance.
We are going to describe how the *round function* $\gamma\lambda\sigma_{k^{(\rho)}}$ (in the ρ-th typical round) is performed.
As in the AES and SERPENT cryptosystems, the encryption process starts with a preliminary round (`Round 0`) that consists of a parallel map $\gamma = 1_V$, a linear transformation $\lambda = 1_V$ and the translation $\sigma_{k^{(0)}}$, where $k^{(0)} \in (\mathbb{F}_2)^r$ is the first round key. A typical round consists of the non-linear operation, called sBoxLayer, the linear transformation, known as pLayer and the sum with the round key.

2.3.1 sBoxLayer

The parallel map $\gamma \in \mathrm{Sym}(V)$ used in PRESENT acts as $v\gamma = v_1\gamma_1 \oplus \ldots \oplus v_{16}\gamma_{16}$, where each $v_i \in (\mathbb{F}_2)^4$ and $\gamma_i \in \mathrm{Sym}((\mathbb{F}_2)^4)$ $(1 \leq i \leq 16)$. The action of any brick $\gamma_i : (\mathbb{F}_2)^4 \to (\mathbb{F}_2)^4$ is given by the following table, using an hexadecimal notation:

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$\gamma[x]$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

2.3.2 pLayer

The affine map $\lambda : V \to V$ is a bit permutation as given by the following table, where the bit i of the intermediate state is moved to the bit position $P(i)$.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(i)$	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

The key schedule

PRESENT supports keys of either 80 or 128 bits. However, we focus on the version with 80-bit keys. A further useful key is stored in key register K and it is represented as $K_{79}K_{78}\dots K_0$. The ρ-th round key consists of the 64 leftmost bits of the current content of the key register K, i.e. $k^{(\rho)} = K_{63}K_{62}\dots K_0 = K_{79}K_{78}\dots K_{16}$. After extracting the round key $k^{(\rho)}$, the key register $K = K_{79}K_{78}\dots k_0$ has to be update. The updating procedure occurs in this way:

1. $[K_{79}K_{78}\dots K_1K_0] = [K_{18}K_{17}\dots K_{20}K_{19}]$
2. $[K_{79}K_{78}K_{77}K_{76}] = \gamma_i[K_{79}K_{78}K_{77}K_{76}]$
3. $[K_{19}K_{18}K_{17}K_{16}K_{15}] = [K_{19}K_{18}K_{17}K_{16}K_{15}] \oplus c_r$

where c_r is a round-counter. Thus, the key register is rotated by 61 bit positions to the left, the left-most four bits are passed through the present S-box, and the round-counter value ρ is exclusive xored with bits $K_{19}K_{18}K_{17}K_{16}K_{15}$ of K with the least significant bit of c_r on the right.

On the AES cryptanalysis

In this chapter we give an overview of the known attacks, especially when applied to the AES cryptosystem. In Section 3.1 we recall that AES is optimized to resist all known *statistical* attack. In Section 3.2 we describe some of the *structural* attacks that are relevant in the cryptanalysis of reduced variants of AES-128. In Section 3.3, Section 3.4, and Section 3.5, we see how the *algebraic attacks* could be useful for a cryptanalyst and we point out some of the alternative representations proposed for the AES. In particular, we explain the BES representation and the *Dual Cipher*. Finally, in Section 3.6 we give a sketch of an approach based on changing the S-boxes.

3.1 Statistical attacks

Differential and Linear cryptanalysis are two conventional methods of attack against block ciphers. They attempt to construct statistical patterns via many encryptions, in order to distinguish the cipher from a random permutation and get the key. In the Differential cryptanalysis, the statistical pattern depends on bitwise differences, instead in Linear cryptanalysis it depends on the correlation among bits.

Linear cryptanalysis, described by Matsui in [Mat93], is a known-plaintext attack. It requires to find a set of linear approximations of the S-boxes that can be used to derive a linear approximation of the whole cipher. The S-boxes used in the approximations are called *active S-boxes*. Suppose that it is possible to find a probabilistic linear relationship between a subset of plaintext-bits and a subset of state-bits immediately preceding the substitutions performed in the last round. We assume that Eve has a *large number* of (plaintext, ciphertext) pairs encrypted using the same unknown key k. For each of the (plaintext, ciphertext) pairs, we will begin to decrypt the ciphertext, using *all possible candidate keys* for the last round of the cipher. For each candidate pair, we compute the values, of the relevant state-bits involved in the linear relationship and determine if the above mentioned linear relationship holds. Whenever it does, we increment a counter corresponding to a particular "candidate key". At the end of this process, we hope that the candidate key with a frequency count furthest from half times the number of pairs contains the correct values for these key bits.

Differential cryptanalysis is a chosen-plaintext attack and was described by Biham and Shamir in [BS93]. The main difference from the linear cryptanalysis is that it involves comparing the XOR of two inputs to the XOR of the corresponding two outputs. Eve encrypts pairs of plaintexts and studies the propagation of differences between inputs of rounds of the cipher. We assume that Eve has a *large number* of tuples (P, P', Q, Q'), where the value $P \oplus P'$ is fixed. The plaintext elements P and P' are encrypted using the same unknown key yielding the cyphertexts Q and Q' respectively. For each of these tuples, we will begin to decrypt the corresponding ciphertexts using *all possible candidate keys* for the last round of the cipher. For each candidate keys, we compute the values of certain state bits and determine if their XOR has a certain value. Whenever it does, we increment a counter corresponding to the particular candidate key. At the end of this process, we hope that the candidate key having the highest frequency count contains the correct values for these bits.
The Differential cryptanalysis is thwarted by

- careful S-box construction: the probability p of a given bitwise non-zero difference propagation across an S-box is $< 2^{-6}$, for DES;
- carefully designed diffusion layer.

The total differential probability behaves as p^n; attack requirements are proportional to $1/p^n$.

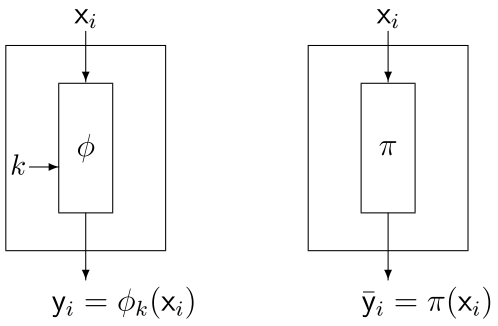
The AES cryptosystem is very resistant to these statistical attacks. For differential and linear cryptanalysis, attacks over 4 rounds of the AES require at least 25 active S-boxes. More careful analysis takes account of additional complicated phenomena. Exploiting differential and linear techniques requires a massive number of (plaintext, ciphertext) pairs and the complexity usually grows exponentially with the number of rounds, ensuring that such attacks rapidly become impractical, and so a different cryptanalytic approach is required.

3.1.1 Distinguishing Attacks

Statistical tools have usually been used with cryptanalytic attacks against block ciphers. Statistical hypothesis testing is a formal way for distinguishing between two probability distributions. Starting from some samples coming from these two distributions, it is possible to compute the probability that the two distributions differ. In particular, a distinguishing attack on a cipher $\mathcal{C}$ relates to the formal model of security, where an adversary can distinguish between the output of $\mathcal{C}$ (with a fixed key) and the output of a random process, with significant certainty.

Distinguishing attack against block ciphers aim at determining whether a permutation corresponds to a permutation chosen uniformly at random from the set of all permutations or one of the permutations specified by a secret key. Of course, there is always a distinguishing attack against any algorithmic cipher, since it must have a finite key, and so brute-force key enumeration will yield a distinguishing attack of complexity $2^{\ell-1}$, where ℓ is the key length. Being able to identify some distinguishing characteristic of the output might lead to an attack that reveals information about the key of the cipher. Any such attack against an iterated block cipher is a serious threat, since it can usually be transformed into a key-recovery attack, for example by combining it with an exhaustive search for the last round key.

Let $\mathsf{x}_1, \dots, \mathsf{x}_n$ be some plaintexts, let k be a chosen key. We denote by π any random permutation and by ϕ_k the encryption function for the key k; we have to consider the following situation:


x_i
k
ϕ
$\mathsf{y}_i = \phi_k(\mathsf{x}_i)$
x_i
π
$\bar{\mathsf{y}}_i = \pi(\mathsf{x}_i)$

We have to provide an algorithm to see that the ciphertexts $\mathsf{y}_1, \dots, \mathsf{y}_n$ do not come from π.

As of now, the full version[1] of the AES-128 is considered very secure, since there is no successful attack against it, not even a statistical distinguishing attack.

3.2 Structural attacks

The AES is optimized against known statistical attacks. However, its structure can be used to carry out some innovative analysis. Such attacks tend to have a similar form:

- they identify a property that holds for a few rounds with a good probability;
- they use special techniques to extend the attack to more round.

Well-known examples of structural attacks are Square Attack, Impossible Differentials, Boomerang Attacks, Related Key and Collision Attacks (and their variants).

[1]Reduced versions of the AES-128 have been broken, but these attacks are far from being applicable to the full version, since they would require more time than a brute force key check.

The following table summarizes the more successful attacks on reduced versions of the AES cryptosystem:

Key	Rounds	Texts	Time	Type	Reference
128	5	2^{11}	2^{40}	Square attack	[DR98]
128	5	$2^{29.5}$	2^{31}	Impossible diff.	[BK00]
128	5	2^{39}	2^{39}	Boomerang attack	[Wag99]
128	6	2^{32}	2^{72}	Square attack	[DR98]
128	6	$2^{34.6}$	2^{44}	Partial Sum	[FKL+00]
128	6	$2^{91.5}$	2^{122}	Impossible diff.	[CKK+01]
128	6	2^{71}	2^{71}	Boomerang attack	[Wag99]
128	7	$2^{128} - 2^{119}$	2^{120}	Partial Sum	[FKL+00]
128	7	2^{32}	2^{128}	Collision	[GM00]
192	7	2^{92}	2^{186}	Impossible diff.	[Pha04]
192	8	2^{127}	2^{188}	Partial Sum	[FKL+00]
192	10	2^{124}	2^{183}	(Related-key) Rectangle	[BDK05]
192	12	2^{123}	2^{176}	(Related-key) Ampl. Boomerang	[BK09]
256	7	$2^{92.5}$	$2^{250.5}$	Impossible diff.	[Pha04]
256	8	2^{32}	2^{194}	Partial Sum	[FKL+00]
256	9	2^{85}	2^{126}	Partial Sum	[FKL+00]
256	10	2^{114}	2^{173}	(Related-key) Rectangle	[BDK05]
256	14	2^{119}	2^{119}	(Related-key) Boomerang	[BK09]

3.2.1 Square attack

This is a chosen-plaintext attack that works on any cipher with a round structure similar to that of AES. It was first described in the paper presenting a predecessor of AES, the block cipher Square [DKR97]. For this reason, it is usually called "Square" attack. Other names for this attack are *Saturation attack* and *Integral attack* or *Structural attack*. The original Square attack is able to break reduced variants of AES up to 6 or 7 rounds faster than exhaustive key search. In 2000 Ferguson et al. [FKL+00] proposed some optimizations that reduce the work factor of the attack. In this way, this attack can break up to 9-round of the AES-256 keys with 2^{77} plaintexts under 256 related keys and 2^{224} encryptions.

In order to describe the 6-round Square Attack, we first explain the basic attack on 4 rounds and then we will show how Deamen and Rijmen extended it to the 5-th and the 6-th rounds in their original proposal [DR98]. This attack is independent of the specific choices of SubBytes, the multiplication polynomial of MixColumns and the Key Schedule and works against all block sizes and key sizes.

Consider a set in which only one byte is active, that is, we change the value of one bytes only. We can see the following situation

- MixColumns of the 1st round converts the active byte to a complete column of active bytes.

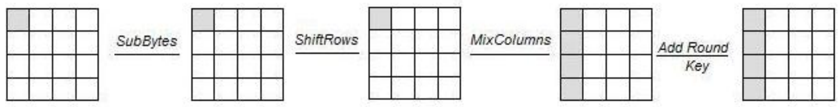


Figure 3.1: Square attack, first round

- The four active bytes of this column are spread over four distinct columns by ShiftRows of the 2nd round.
- MixColumns of the 2nd round subsequently converts this to 4 columns of only active bytes.

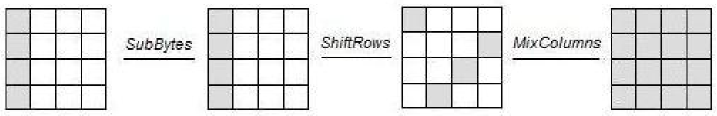


Figure 3.2: Square Attack, second round

- Although at the end of the second round all bytes potentially change, we are still able to recover the first byte by solving a linear system. This is not possible any more after the input of MixColumns of the 3rd round.

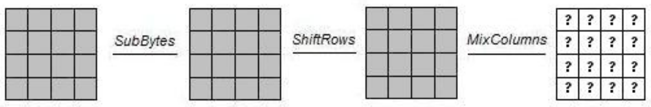


Figure 3.3: Square Attack, third round

We use $m^{(\rho)}$, $b^{(\rho)}$, and $t^{(\rho)}$ to refer to intermediate text values used in round ρ after the MixColumns, key addition, and ShiftRows operations, respectively. We write $k^{(\rho)}$ for the subkey in round ρ , and $k^{(\rho)'}$ for an equivalent subkey value that may be xored into the state before (instead of after) the MixColumns operation in round ρ .
The idea is to choose a set of plaintexts that results in a set at the output of the 1st round with a single active S-box. This requires the assumption of values of four bytes of the Round Key that is applied before the first round.
If the intermediate state after MixColumns of the first round has only a single active byte, this is also the case for the input of the second round. This imposes the following conditions on a column of 4 input bytes of MixColumns of the second round: one particular linear combination of these bytes must range over all 256 possible values (active) while 3 other particular linear combinations must be constant for all 256 states. This imposes identical conditions on 4 bytes, in different positions at the input of ShiftRows of the first round. If the corresponding bytes of the first round key are known, these conditions can be converted to conditions on 4 plaintext bytes.
Now we consider a set of 2^{32} plaintexts, such that one column of bytes at the input of MixColumns of the first round ranges over all possible values and all other bytes are constant. Now, an assumption is made for the value of the 4 bytes of the relevant bytes of the first Round Key. From the set of 2^{32} available plaintexts, a set of 256 plaintexts can be selected that result in a set at the input of round 2. Now the 4-round attack can be performed.
For the given key assumption, the attack can be repeated for several plaintext sets. If the byte values of the last Round Key are not consistent, the initial assumption must have been wrong. A correct assumption for the 32 bytes of the first Round Key will result in the swift and consistent recuperation of the last Round Key.

3.2.2 Partial Sum

The attack of previous section on 4 rounds of AES can be improved. Instead of guessing four bytes of $k^{(0)}$ we simply use all 2^{32} plaintexts.
For any value of the first round key, these encryptions consist of 2^{24} groups of 2^8 encryptions that vary only in a single byte of the output of the MixColumns operation at first round. All we have to do is to guess the five key bytes at the end of the cipher, do a partial decrypt to a single byte of $b^{(4)}$, sum this value over all the 2^{32} encryptions, and check for a zero result. Compared to the original version, we guess only 40 bits of key instead of 72. On the other hand, we have to do 2^{24} times as much work for each guess. All in all, this improvement reduces the workload by a factor of 2^8, although it needs about $6 \cdot 2^{32}$ plaintexts to provide enough sets of 2^{32} plaintexts to uniquely identify the proper value for the five key bytes.

We will now look at this attack in more detail.
We have 2^{32} ciphertexts. We guess five key bytes, do a partial decryption from each of the ciphertexts to a single byte in $b^{(4)}$, and sum this byte over all ciphertexts.
Consider this partial decryption. From any ciphertext, we use four ciphertext bytes. Each of these is xored with a key byte. We then apply the inverse S-box to each byte, and multiply each with an appropriate factor from the inverse MDS matrix. The four bytes are then xored together, a fifth key byte is xored into the result, the inverse S-box is applied, and the resulting value is summed over all ciphertexts.
Let $c_{i,j}$ be the j-th byte of the $i-$th ciphertext. (We leave out the i subscript if we are not talking about any particular ciphertext.) For simplicity we will number the four bytes of each ciphertext that we use from 0 to 3.
Let $k_0, \ldots, k_4$ denote the five key bytes that we are guessing. We want to compute the following

$$\sum_i S^{-1}[S^0[c_{i,0} \oplus k_0] \oplus S^1[c_{i,1} \oplus k_1] \oplus S^2[c_{i,2} \oplus k_2] \oplus S^3[c_{i,3} \oplus k_3] \oplus k_4]$$

where $S^0, \ldots, S^3$ are bijective S-boxes, each of which consists of an inverse AES S-box followed by a multiplication by a field element from the inverse MDS matrix.
Given 2^{32} ciphertexts and 2^{40} possible key guesses, we have to sum 2^{72} different values, which corresponds roughly in amount of work to doing about 2^{64} trial encryptions.
We can organize this more efficiently in the following manner. For each k, we associate a "partial sum" x_k to each ciphertext c, defined as follows:

$$x_k := \sum_{j=0}^{k} S_j[c_j \oplus k_j]$$

This gives us a map $(c_0, c_1, c_2, c_3) \mapsto (x_k, c_{k+1}, \ldots, c_3)$ that we can apply to each ciphertext if we know $k_0, \ldots, k_k$.
We start out with a list of 2^{32} ciphertexts. We guess k_0 and k_1 and compute how often each triple (x_1, c_2, c_3) occurs in the list. That is, for each i, we compute the three-byte value $(S_0[c_{i,0} \oplus k_0] \oplus S_1[c_{i,1} \oplus k_1], c_{i,2}, c_{i_3})$ as a function of the i-th ciphertext and the guessed key material, and we count how many times each three-byte value appears during this computation. As there are only 2^{24} possible values for three bytes, we do not have to list all (x_1, c_2, c_3) values; rather, we count how often each triple occurs. We then guess k_2 and compute how often each tuple (x_2, c_3) occurs; and guess k_3 and compute how often each value of x_3 occurs. Finally, we guess k_4 and compute the desired sum. Because all sums are taken using the XOR operation (and because $z \oplus z = 0$ for all z), it suffices to only count modulo two. Thus, a single bit suffices for each count, and so the space requirement for the 2^{24} counters is just 2^{24} bits.

How much work has this been? In the first phase we guessed 16 bits and processed 2^{32} ciphertexts, so this phase costs 2^{48} overall. In the next phase, we guessed a total of 24 bits but we only had to process 2^{24} triples, so this costs 2^{48} as well. This holds similarly for each of the phases. In total, the entire computation requires the equivalent of about 2^{48} evaluations of equation 1, or about 2^{50} S-box applications.
This is the amount of work required for a single structure of 2^{32} ciphertexts. The first structure already weeds out the overwhelming majority of the wrong key guesses, but we still have to do the first steps of our partial sum computation for each of the six structures that we use. The total number of S-box lookups is thus about 2^{52}.
Using our earlier rough equivalence of 2^8 S-box applications to a trial encryption with a new key, the 2^{52} S-box applications are comparable to 2^{44} trial encryptions. This is a significant improvement over the earlier 2^{72} work factor.

Remark 3.2.1. We can extend the previous approach to the 7-round version of this attack. To express a single byte of $b^{(4)}$ in the key and the ciphertext, we get a formula similar to equation 1 but with three levels, 16 ciphertext bytes, and 21 key bytes. The partial sum technique is only helpful during the last part of the computation as it only saves work if there are more ciphertexts than possible values for the intermediate result. With 2^{32} plaintext/ciphertext pairs in a structure, these techniques will not help until the very last part of the computation.

3.2.3 Impossible Differentials

There exists an impossible differential attack on 5 rounds [BK00], requiring $2^{29.5}$ chosen plaintext, 2^{31} encryptions, 2^{42} bytes of memory and 2^{26} time for pre-computation. This result was improved and lead to an attack on a 6 round version [CKK$^+$01].

3.2.4 Collision Attacks

This attack has been introduced by Gilbert and Minier in [GM00] and is still the best attack in the sense that it can break 7 rounds of AES-128, AES-192 and AES-256. For AES-128 the authors claim that the complexity of the attack is marginally lower than the complexity of an exhaustive key search.

3.2.5 Boomerang attack

The Boomerang attack is an "adaptive chosen-plaintext" and "adaptive chosen-ciphertext" attack (see Section 1.2.3) and was introduced by Wagner in [Wag99]. This attack is based on *two differentials*. The main problem consists of finding "good" differentials. Suppose we have a block cipher $\phi : \{0,1\}^r \times \{0,1\}^\ell \to \{0,1\}^r$ for which we know that an input difference α induces an output difference β with probability p. We use this property to retrieve the round key both in the cipher itself, or if we are dealing with part of the cipher, in the rounds before and after the part for which we have a differential. This kind of work is less effective against ciphers where the probability of the next differential drops very fast with the number of rounds: AES and SERPENT are such two examples. In the AES cryptosystem there are 1-round differentials with probability 2^{-7} and 2-round differentials with probability 2^{-35}. The probability of the best differential drops faster than exponentially with the number of rounds. In particular, it can be shown that the best 4-round differential for AES has probability no more than 2^{-96}.

Sketch of the attack

Let us assume that the encryption function ϕ is a composition of two parts: ϕ_0 and ϕ_1. Suppose to have a good differential in $\phi_0 : \alpha \to \beta$ with probability p and another good differential in $\phi_1 : \gamma \to \delta$ with probability q.
As mentioned before, it is easy to find such short differentials (for parts of the cipher), but we need to find long differentials with good probability. The question is the following: how to combine these two differentials to mount an attack against the full cipher ϕ?
Let us examine the encryption of the plaintexts P_1 and $P_2 = P_1 + \alpha$ through ϕ_0. Starting from the differential we have that $\phi_0(P_1) + \phi_0(P_2) = \beta$ with probability p. The corresponding ciphertexts are $C_1 = \phi_1(\phi_0(P_1))$ and $C_2 = \phi_1(\phi_0(P_2))$.
Now we compute $C_1 + \delta = C_3$ and $C_2 + \delta = C_4$. When we partially decrypt the values of C_1 and C_3 though ϕ_1, we know that

$$\phi_1^{-1}(C_1) + \phi_1^{-1}(C_3) = \gamma$$

with probability q. The same holds for C_2 and C_4:

$$\phi_1^{-1}(C_2) + \phi_1^{-1}(C_4) = \gamma.$$

Since $\phi_0(P_1) = \phi_1^{-1}(C_1)$ and $\phi_0(P_2) = \phi_1^{-1}(C_2)$, using the previous equalities, we have

$$\begin{aligned} \phi_0(P_1) + \phi_1^{-1}(C_3) &= \phi_0(P_2) + \phi_1^{-1}(C_4) \\ \phi_0(P_1) + \phi_0(P_2) &= \phi_1^{-1}(C_3) + \phi_1^{-1}(C_4) \\ \beta &= \phi_1^{-1}(C_3) + \phi_1^{-1}(C_4) \end{aligned}$$

Hence, if we continue the decryption process, we get $P_3 + P_4 = \alpha$ with probability q^2. Under the above assumptions, we say that both (P_1, P_2) and (P_3, P_4) are *right pairs* for the characteristic $\alpha \to \beta$ and that (C_1, C_2) and (C_3, C_4) are *right pairs* for the characteristic $\gamma \to \delta$. Then, we can use these for retrieving the subkeys in ϕ_0 and ϕ_1 similarly to differential cryptanalysis. The probability for such a quartet to be *right* is 2^{-n}.

The Amplified Boomerang attack

Kohno, Kelsey and Scheneir [KKS00] introduced the Amplified Boomerang Attack. This method converts the Adaptive Chosen Plaintext and Ciphertext into a Chosen Plaintext attack. This is done by encrypting many plaintexts pairs with input difference α. If we start with N plaintext pairs $(P, P + \alpha)$, then about Np of those pairs will have a difference β after ϕ_0.
With probability 2^{-n} the difference between the intermediate encryption value of P_1 and P_3 is γ. If this is the case, then the difference between the intermediate encryption values of P_2 and P_4 is also γ. Then, with probability q for each of these pairs, we get that the corresponding ciphertexts pairs have difference δ.
Hence, out of the N pairs we started with, we get $2^{-n}p^2q^2N^2/2$ quadruplets.

The Rectangle attack

The amplified boomerang attack has a major drawback: one needs to check all possible quadruplets to find the *right* quadruplets. Unlike the boomerang attack, where we know exactly what plaintexts we need to compare, in the amplified boomerang attack, we have no idea about the quadruplets that we need to check.
Another drawback is the very low probability of a quadruplet to be a right quadruplet. For example, even if the probabilities of the differentials are 1 (i.e. $p = q = 1$), in order to obtain one right quadruplet, we still need $2^{n/2}$ pairs.
Biham, Dunkelman and Keller [BK00] introduced the Rectangle Attack. The attack has much higher probability (i.e. the probability of a quadruplet to be a right quadruplet is higher) and also better attack algorithm (based on more efficient data structures and attack algorithm).
By examining both the pair (C_1, C_3) and the pair (C_2, C_4) for a δ difference, the number of right quadruplets is multiplied by 2, (as we check N^2 different quadruplets).
Moreover it is possible to apply the improvements of the boomerang attack which use many differentials simultaneously.

Attacks on small scale variants of AES

We describe a generic method of breaking 5 round (using boomerang techniques) of one AES-like cipher. This structural attack does not use specific properties of the S-boxes or the Mixing Layer, but it uses only the fact that diffusion is incomplete.

We also note that the exact constants in the MixColumns matrix will be irrelevant to the attack.

- Let $\{P_i\}$, $i = 0, \ldots, 2^{32} - 1$ be a pool of plaintexts which have all possible values in four bytes and arbitrary constants in the other bytes. Encrypt each P_i obtanining a pool of 2^{32} ciphertexts $\{C_i\}$.
- Contruct a pool of modified ciphertexts: $D_i = C_i \oplus \nabla$, where ∇ is a fixed non-zero difference with only one active S-box. Decrypt the pool $\{D_i\}$ to obtain a pool $\{Q_i\}$ of 2^{32} new plaintexts.
- Sort the pool $\{Q_i\}$ by the bytes corresponding to eight inactive S-boxes. Pick only those pairs Q_i, Q_j which have zero difference in these 8 bytes. If none is found we have to come back to the first step.
- For each of the right (according to the previous step) quartets P_i, P_j, Q_i, Q_j, guess the 32-bit key value that enters the 4 S-boxes corresponding to non-constants bytes. Using the guessed key value, partially encrypt one round and check the resulting difference in a single active S-box, which is a 22-bit filtering condition for each pair (P_i, P_j) and (Q_i, Q_j). This gives a 44-bit condition in total for both sides of the boomerang in the case of common 4-tuples to active S-boxes. However, with half probability we will have no common 4-tuples, i.e. all the 12 active S-boxes (4 from the (P_i, P_j) pairs and 8 from the (Q_i, Q_j) pairs) do not overlap. We will pick key-candidates that are suggested at least twice.

Remark 3.2.2. The attack described above can be extended by one round to the bottom at the cost of guessing 32-bit of the key of the 6th round.

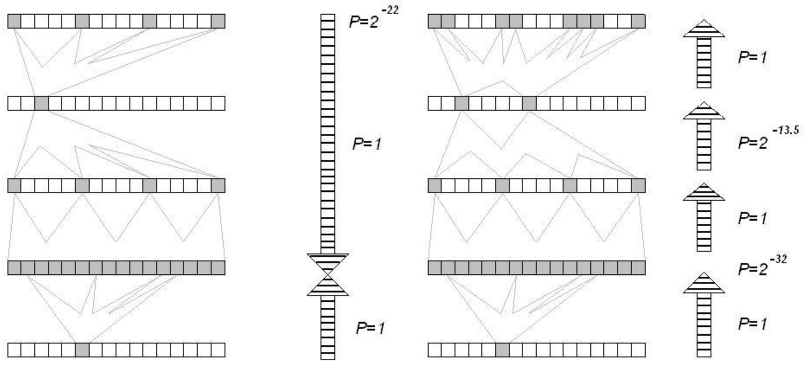


Figure 3.4: Schematic description for AES reduced to five rounds

3.3 First algebraic attacks

In contrast to conventional block cipher cryptanalysis, algebraic cryptanalysis exploits the intrinsic algebraic structure of a cipher. In its most common form, a cryptanalyst describes the encryption transformation as a set of multivariate polynomial equations, which once solved can be used to recover information about the secret key. The algebraic attacks can be briefly sketched as follows:

- *Collecting Step*: Eve expresses the cipher as a set of "suitable" equations in one or more variables. These variables may include bits (or bytes) from the plaintext, ciphertext and the key. Typically, the variables include also intermediate computation values and round keys.

- *Solving Step*: Eve uses some data input, such as the pairs (plaintext, ciphertext); she substitutes these input values in the corresponding variables (in the set of equations collected in the previous step) and tries to solve the resulting set of equations, thereby recovering the key.

Several attempts have been made to construct algebraic attacks for the AES. They have resulted in small scale variants attacks as yet, and many of the related papers conclude that more research is required. The most important attempts in this sense are the following:

- Ferguson, Shroeppel and Whiting in [FSW01] derive a closed formula for AES that can be seen as a generalization of continued fractions.

- Courtois and Pieprzyck [CP02] observe that the S-box used in the AES can be described by a number of implicit quadratic Boolean equations.

- Murphy and Robshaw [MR02] define the block cipher BES which operates on data blocks of 128 bytes instead of bits. According to Murphy and Robshaw, the algebraic structure of BES is even more elegant and simple than that of the AES.

- In 2002 Barkan and Biham [BB02b] introduced the concept of *Dual ciphers*. It is basically a generalization of the embedding technique.

3.3.1 Continued fractions

Let us consider a typical round of the AES. We write an algebraic expression of all the steps:

$$\begin{aligned}
\textsf{SubBytes:}\quad & s_{i,j}^{(\rho)} = S[a_{i,j}^{(\rho)}] = \textstyle\sum_{d_\rho=0}^{7} w_{d_\rho} (a_{i,j}^{(\rho)})^{-2^{d_\rho}} \\
\textsf{ShiftRows:}\quad & t_{i,j}^{(\rho)} = s_{i,i+j}^{(\rho)} \\
\textsf{MixColumns:}\quad & m_{i,j}^{(\rho)} = \textstyle\sum_{e_\rho=0}^{3} v_{i,e_\rho} t_{e_\rho,j}^{(\rho)} \\
\textsf{AddRoundKey:}\quad & a_{i,j}^{(\rho+1)} = m_{i,j}^{(\rho)} + k_{i,j}^{(\rho)}
\end{aligned}$$

where w_{d_ρ} are suitable constants, $a_{i,j}^{(\rho)}$ is the byte at position (i,j) at the input of round ρ, $v_{i,j}$ are the coefficients of the MDS matrix and $k_{i,j}^{(\rho)}$ is the ρ-th round key at position (i,j). Clearly, we can rewrite the last formula in the following way:

$$a_{i,j}^{(\rho+1)} = k_{i,j}^{(\rho)} + \sum_{e_\rho \in \mathcal{E},\, d_\rho \in \mathcal{D}} \frac{w_{i,e_\rho,d_\rho}}{(a_{e_\rho,e_\rho+j})^{2^{d_\rho}}},$$

where $\mathcal{E} := \{0, \dots 3\}$ and $\mathcal{D} := \{0, \dots 7\}$.
What happens after the second round? We get the following expression

$$a_{i,j}^{(3)} = k_{i,j}^{(3)} + \sum_{e_2 \in \mathcal{E},\, d_2 \in \mathcal{D}} \frac{w_{i,e_2,d_2}}{\left(k_{e_2,e_2+j}^{(1)} + \sum_{e_1 \in \mathcal{E},\, d_1 \in \mathcal{D}} \frac{w_{e_2,e_1,d_1}}{(a_{e_1,e_1+e_2+j})^{2^{d_1}}}\right)^{2^{d_2}}}$$

or equivalently, we have

$$a_{i,j}^{(3)} = k_{i,j}^{(3)} + \sum_{e_2 \in \mathcal{E},\, d_2 \in \mathcal{D}} \frac{w_{i,e_2,d_2}}{(k_{e_2,e_2+j}^{(1)})^{2^{d_2}} + \sum_{e_1 \in \mathcal{E},\, d_1 \in \mathcal{D}} \frac{w_{e_2,e_1,d_1}^{2^{d_2}}}{(a_{e_1,e_1+e_2+j})^{2^{d_1+d_2}}}}.$$

Note that all the subscripts are known and they are independent of the key or plaintext. The same holds for all the exponents, that is they are known and independent of the plaintext and key.
A fully expanded version of any byte of the intermediate result after 5 rounds has 2^{25} terms. Even the full 10 round formula would require only 2^{50} terms or so, which is certainly computable within the workload allowed for an attack on a 128-bit cipher. Therefore, the security of AES depends on the following assumption: it is computationally infeasible to solve equations of this type.
In order to break the AES cryptosystem, Eve could use two equations of this type for each intermediate byte. The first one would express the intermediate variables after 5 rounds as function of the plaintext bytes. The second equation would cover rounds, from the 6th to the 10th, by expressing the same intermediate variables as a function of the ciphertext bytes. Combining both equations would result in an equation

with 2^{26} unknowns. By repeating this equation for $2^{26}/16$ known pairs (plaintext, ciphertext), enough information could be gathered to solve for the unknowns, in an information-theoretic sense. It is currently unknown what practical algorithm might solve this type of equations.

3.3.2 Polynomial system approach

In this subsection we consider the encryption as a set of multivariate polynomials equations. Once this system is solved, we can use the corresponding solution to recover information about the secret key.

Let $\phi : (\mathbb{F}_2)^r \times (\mathbb{F}_2)^\ell \rightarrow (\mathbb{F}_2)^r$ be the encryption function. The components $\phi_1, \cdots, \phi_r$ of ϕ are polynomials in $\mathbb{F}_2[x_1, \ldots, x_r, k_1, \ldots, k_\ell]$.
Since we are looking for solutions in $\mathbb{F}_2$, we have the following relations

$$\begin{cases} (x_1)^2 = x_1 \\ \quad\vdots \\ (x_r)^2 = x_r \\ (k_1)^2 = k_1 \\ \quad\vdots \\ (k_\ell)^2 = k_\ell \end{cases}$$

We want to apply the Chosen Plaintext attack. Let $\mathsf{x} \in \mathcal{M}$ be a plaintext and let $\mathsf{y} \in \mathcal{M}$ be the corresponding ciphertext. For any pair $\{(\mathsf{x}, \mathsf{y})\}$ we have this system

$$\begin{cases} \phi_1(\mathsf{x}, k) = \mathsf{y}_1 \\ \phi_2(\mathsf{x}, k) = \mathsf{y}_2 \\ \quad\vdots \\ \phi_r(\mathsf{x}, k) = \mathsf{y}_r \end{cases}$$

Remark 3.3.1. Note that y_i and x_i $(i = 1, \ldots, r)$ are constants and that the components of the key $k_1, \ldots, k_\ell$ are variables.

We denote with $\varepsilon_i(k) = \phi_i(k, \mathsf{x})$ for $i = 1, \ldots, r$. Note that ε_i is a polynomials in $\mathbb{F}_2[k_1, \ldots, k_\ell]$. We have to solve the following system

$$\begin{cases} \varepsilon_1(k) = \mathsf{y}_1 \\ \varepsilon_2(k) = \mathsf{y}_2 \\ \quad\vdots \\ \varepsilon_r(k) = \mathsf{y}_r \\ (k_1)^2 = k_1 \\ (k_2)^2 = k_2 \\ \quad\vdots \\ (k_\ell)^2 = k_\ell \end{cases}$$

We overcome the following kinds of Problems:

1. We can describe the encryption function ϕ but it could be very difficult to find the components $\varepsilon_1, \cdots, \varepsilon_r$ describing the function as polynomials.

2. Suppose that we know $\varepsilon_1, \cdots, \varepsilon_r$. The corresponding system can be too dense to be stored on a computer.

3. Even if we can store the system, to solve it could be very difficult (*unless* it is very sparse and it has low degree).

How to solve polynomial systems? Well-known techniques to solve a system are the following:

- explicit substitutions when possible;

- compute Gröbner basis.

A classical general algorithm for computing a Gröbner basis of polynomial ideal is Buchberger's algorithm ([Buc65], [Buc06]). The $F4$ and $F5$ algorithms have been proposed as alternative approaches for computing Gröbner bases. Since the main computational cost of Buchberger's algorithm lies in polynomial reductions, which take place sequentially, the $F4$ essentially replaces many sequential polynomial reductions with a matrix reduction, $F5$ algorithm permits in addition to detect useless polynomial reductions. These algorithms are based on the idea of combining Gröbner basis computation with Gaussian elimination and work by performing the multivariate division algorithm as a matrix reduction.
For AES-128 [BPW06a], a zero-dimensional Gröbner basis for the key-recovery ideal can be constructed with minimal computational effort, without performing a single polynomial reduction. This is achieved by constructing a polynomial system in which all leading terms are pairwise prime, allowing the first Buchberger criterion to be used to show that the resulting set of polynomials forms a Gröbner basis. The Gröbner basis of the key-recovery ideal for AES-128 $I_{AES} \in R_{AES}$ [BPW06b] consists of 200 polynomials of degree 254 and 152 linear polynomials in a ring of 352 variables. The vector space dimension $\dim(R_{AES}/I_{AES})$ unfortunately is 254^{200}, which makes the Gröbner basis unsuitable for cryptanalysis.

In the real situations the polynomial approach does not work and one can change the system representation. In practice, a very efficient solver of sparse polynomial equations over $\mathbb{F}_2$ is PolyBori ([pol], [BD07]).

3.4 Alternative representations

Several alternative representations have been proposed for the AES. Let ϕ be our original block cipher and ϕ' be a new block cipher. We can define a representation map σ (1-1) and

$$\begin{tikzcd}
(\mathbb{F}_q)^n \arrow[r, "\sigma"] \arrow[d, "\phi_k"] & (\mathbb{F}_Q)^m \arrow[d, "\phi'_k"] \\
(\mathbb{F}_q)^n \arrow[r, "\sigma"] & (\mathbb{F}_Q)^m
\end{tikzcd}$$

We say that the block cipher ϕ' is an alternative representation of the block cipher ϕ if the previous diagram is a commutative diagram.

Murphy and Robshaw defined a new block cipher, the Big Encryption Standard (BES), which operates on data blocks of 128 bytes instead of bits. The algebraic structure of BES is even simpler than that of AES. Furthermore, the AES can be embedded into BES; indeed there exists a map σ such that

$$AES_K(\mathsf{x}) = \sigma^{-1}(BES_{\sigma(K)}(\sigma(\mathsf{x}))).$$

Murphy and Robshaw believed that when the XSL method [CKPS00] is applied to BES, the complexity of the *Solving step* could be significantly smaller than in the case where XSL is directly applied to AES. In [LK07] the authors descussed of the XSL method against BES.
Toli and Zanoni proved in [TZ05] that Murphy and Robshaw's system is not complete, because they omitted some indispensable equations and so this reduced-complexity do not work to attack AES. Note that the BES could still be an interesting representation.

3.4.1 BES

The BES cipher, in which AES is embedded using a natural map, involves only computations in $\mathbb{F}_{2^8}$. This fact permits to describe AES using some polynomial equation systems. Solving them means to find the key or another variable and therefore to break the cryptosystem.
We denote by $\mathbf{A} = (\mathbb{F}_{2^8})^{16}$ and $\mathbf{B} = (\mathbb{F}_{2^8})^{128}$ the state spaces of AES and BES respectively. The basic tool for embedding is the conjugation operation σ, taking eight successive square powers, for each value in $\mathbb{F}_{2^8}$:

$$\begin{tikzcd}
(\mathbb{F}_{2^8})^{16} \arrow[r, "\sigma"] \arrow[d, "AES"] & (\mathbb{F}_{2^8})^{128} \arrow[d, "BES"] \\
(\mathbb{F}_{2^8})^{16} \arrow[r, "\sigma"] & (\mathbb{F}_{2^8})^{128}
\end{tikzcd} .$$

The map σ is defined in the following way

$$\begin{aligned} a \in \mathbb{F}_{2^8} &\mapsto \sigma(a) = (a^{2^0}, a^{2^1}, \dots, a^{2^7}) \in (\mathbb{F}_{2^8})^8 \\ \mathbf{a} \in (\mathbb{F}_{2^8})^n &\mapsto \sigma(\mathbf{a}) = (\sigma(a_0), \dots, \sigma(a_7)) \in (\mathbb{F}_{2^8})^{8n} \end{aligned}$$

It is easily verified that $\sigma(\mathbf{a}+\mathbf{a}') = \sigma(\mathbf{a}) + \sigma(\mathbf{a}')$ and $\sigma(\mathbf{a}^{-1}) = \sigma(\mathbf{a})^{-1}$ assuming that $0^{-1} = 0$. Moreover, we define $\mathbf{B_A} = \sigma(\mathbf{A}) \subset \mathbf{B}$ as the subset of $\mathbf{B}$ corresponding to $\mathbf{A}$.

Let $\mathbf{p}, \mathbf{c} \in \mathbf{B}$ be the plaintext and the ciphertext, respectively; $\mathbf{w}_i, \mathbf{x}_i \in \mathbf{B}$, where $0 \le i \le 9$, the state vectors before and after the inversion phases and $\mathbf{h}_i \in \mathbf{B}$ the used key. All the phases of AES algorithm may be translated in $\mathbf{B}$ using just linear algebra in $\mathbb{F}_{2^8}$, excepting from inversion, which is done component-wise, as follows. The matrix $L_A : (\mathbb{F}_2)^8 \to (\mathbb{F}_2)^8$ for the affine transformation for one byte in the SubBytes operation can be represented by the polynomial function $f : \mathbb{F}_{2^8} \to \mathbb{F}_{2^8}$:

$$f(a) = \sum_{k=0}^{7} \lambda_k a^{2^k}$$

with

$$\begin{aligned} \lambda_0 &= t^2 + 1 & \lambda_4 &= t^7 + t^6 + t^5 + t^4 + t^2 \\ \lambda_1 &= t^3 + 1 & \lambda_5 &= 1 \\ \lambda_2 &= t^7 + t^6 + t^5 + t^4 + t^3 + 1 & \lambda_6 &= t^7 + t^5 + t^4 + t^2 + 1 \\ \lambda_3 &= t^5 + t^2 + 1 & \lambda_7 &= t^7 + t^3 + t^2 + t + 1 \end{aligned}$$

It is possible to translate this formulation in the $\mathbf{B}$ state space as follows

$$L_B(a) = \sigma(L_A(a)) = (f(a)^{2^0}, \dots, f(a)^{2^7}).$$

The successive squares of f are needed, and the answer is given by a simple induction with basic step

$$(f(a))^2 = \Big(\sum_{k=0}^{7} \lambda_k a^{2^k}\Big)^2 = \sum_{k=0}^{7} \lambda_k^2 a^{2^k \cdot 2} = \sum_{k=0}^{7} \lambda_k^2 a^{2^{k+1}}.$$

The resulting matrix, which we still indicate with L_B, is

$$L_B = [\ell_{ij}]_{i,j=0\dots7} \quad \textit{with} \quad \ell_{ij} = \lambda^{2^i}_{(8-i+j) \mod 8}$$

If we consider the entire transformation $Lin_B : (\mathbb{F}_{2^8})^{128} \to (\mathbb{F}_{2^8})^{128}$, we obtain a block diagonal matrix with 16 blocks equal to L_B.

The AES SubBytes constant $c_A = \mathsf{63} = t^6 + t^5 + t + 1 \in \mathbb{F}_{2^8}$ maps into

$$\begin{aligned}\sigma(c_A) \;=\; & (\mathsf{63}, \mathsf{C2}, \mathsf{35}, \mathsf{66}, \mathsf{D3}, \mathsf{2F}, \mathsf{39}, \mathsf{36}) = (t^6 + t^5 + t + 1, t^7 + t^6 + t, \\ & t^5 + t^4 + t^2 + 1, t^6 + t^5 + t^2 + 1, t^7 + t^6 + t^4 + t + 1, \\ & t^5 + t^3 + t^2 + t + 1, t^5 + t^4 + t^3 + 1, t^5 + t^4 + t^2 + t).\end{aligned}$$

The corresponding BES vector $\mathbf{c}_B$ is obtained by the juxtaposition of 16 consecutive copies of $\sigma(c_A)$, $\mathbf{c}_B = \sigma(c_A, \ldots, c_A) = (\sigma(c_A), \ldots, \sigma(c_A))$, such that

$$[\mathbf{c}_B]_i = [\sigma(c_A)]_{i \mod 8}.$$

The AES ShiftRows operation permutes the bytes in the array. Clearly, this process can be considered as a transformation of the components of a column vector $a \in \mathbf{A}$. It is straightforward to represent this transformation as multiplication of the state vector $a \in \mathbf{A}$ by a 16×16 $(\mathbb{F}_{2^8})$-matrix $\mathbf{R_A} : (\mathbb{F}_{2^8})^{16} \longrightarrow (\mathbb{F}_{2^8})^{16}$:

$$\mathbf{R_A} = \left(\begin{array}{cccc|cccc|cccc|cccc}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
\hline
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
\hline
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
\hline
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0
\end{array}\right)$$

The corresponding BES matrix is obtained by expanding each 1 in R_A with an identity matrix of the order 8 and each 0 with a zero (8×8)-matrix. The result will be $\mathbf{R_B} : (\mathbb{F}_{2^8})^{128} \longrightarrow (\mathbb{F}_{2^8})^{128}$. Moving from $\mathbf{R_A}$ to $\mathbf{R_B}$ we only need to ensure that the vector conjugates are moved as a single entity.

The AES MixColumns may be represented using the following $\mathbf{C_A} : (\mathbb{F}_{2^8})^4 \longrightarrow (\mathbb{F}_{2^8})^4$

$$\mathbf{C_A} = \begin{pmatrix} t & t+1 & 1 & 1 \\ 1 & t & t+1 & 1 \\ 1 & 1 & t & t+1 \\ t+1 & 1 & 1 & t \end{pmatrix}$$

The AES transformation is given by the $Mix_A : (\mathbb{F}_{2^8})^{16} \longrightarrow (\mathbb{F}_{2^8})^{16}$ block diagonal matrix having as blocks four copies of $\mathbf{C_A}$. In order to obtain the corresponding matrix, we first need to compute the following matrices $\mathbf{C_B}^{(k)}$, for $k = 0, \dots, 7$:

$$\mathbf{C_B^{(k)}} = \begin{pmatrix} t^{2^k} & (t+1)^{2^k} & 1 & 1 \\ 1 & t^{2^k} & (t+1)^{2^k} & 1 \\ 1 & 1 & t^{2^k} & (t+1)^{2^k} \\ (t+1)^{2^k} & 1 & 1 & t^{2^k} \end{pmatrix}$$

with

$$\begin{aligned} t^{2^0} &= t & t^{2^4} &= t^6 + t^4 + t^3 + t^2 + t \\ t^{2^1} &= t^2 & t^{2^5} &= t^7 + t^6 + t^5 + t^2 \\ t^{2^2} &= t^4 & t^{2^6} &= t^6 + t^3 + t^2 + 1 \\ t^{2^3} &= t^4 + t^3 + t + 1 & t^{2^7} &= t^7 + t^6 + t^5 + t^4 + t^3 + t \end{aligned}$$

from which $(t+1)^{2^k} = t^{2^k} + 1$ may be very easily computed.
Using an appropriate basis, the resulting matrix $M_B : (\mathbb{F}_{2^8})^{128} \longrightarrow (\mathbb{F}_{2^8})^{128}$ may be written as a block diagonal one, with four consecutive copies of $\mathbf{C_B}^{(k)}$ for all possible k. The different position of value powers in σ's image, with respect to our needs, makes the change of basis necessary. Indeed, if $\mathbf{a} \in (\mathbb{F}_{2^8})^{16}$, then

$$\sigma(\mathbf{a}) = (a_0, \dots, a_0^{2^7}, a_1, \dots, a_1^{2^7}, \dots, a_{15}, \dots, a_{15}^{2^7}),$$

while in order to use the block diagonal representation, we would need the following vector:

$$\mathbf{a}' = (a_0, \dots, a_{15}, a_0^2, \dots, a_{15}^2, \dots, a_0^{2^7}, \dots, a_{15}^{2^7}).$$

This transformation is given by a permutation matrix $Perm_B : (\mathbb{F}_{2^8})^{128} \longrightarrow (\mathbb{F}_{2^8})^{128}$. To represent it easily, suppose to divide it into (16×8) sub-matrices P_{hk}, $h = 0, \dots, 7$ and $k = 0, \dots, 15$. Each sub-matrix element (with $i = 0, \dots, 15$ and $j = 0, \dots, 7$) is :

$$[P_{hk}]_{ij} = \begin{cases} 1 & \text{if } i = k \text{ and } j = h \\ 0 & \text{else} \end{cases}$$

Key schedule: we can use the same techniques from previous sections to describe the key schedule for the BES. In fact the key schedule in the AES uses the same operations as the $\mathbb{F}_2$-linear map, component-wise inversion, byte rotation and addition.

Round function of BES: we can write the final formula for the round function for the BES, supposing that the state at the beginning of the round of the BES is $\mathbf{b} \in \mathbf{B}$ and the BES round key is $\mathbf{k}_{\mathbf{B}i} \in \mathbf{B}$. We can obtain the following

$$\begin{aligned} Round_B(\mathbf{b}, (\mathbf{k}_B)_i) &= Mix_B(R_B(Lin_B(\mathbf{b}^{-1}) + \mathbf{c}_B)) + (\mathbf{h}_B)_i \\ &= M_B \cdot (\mathbf{b}^{-1}) + (\mathbf{C_B}(\mathbf{c}_B)(\mathbf{h}_B)_i) \\ &= M_B \cdot (\mathbf{b}^{-1}) + (\mathbf{k}_B)_i \end{aligned}$$

where

$$M_B = Mix_B \cdot R_B \cdot Lin_B, \quad \mathbf{C_B} = Mix_B \cdot \mathbf{R_B}, \quad (\mathbf{k}_B)_i = \mathbf{C_B}(\mathbf{c}_B) + (\mathbf{h}_B)_i.$$

Note that for the last round, we will have $(\mathbf{k}_B)_i = \mathbf{R_B}(\mathbf{c}_B) + (\mathbf{h}_B)_i$.

3.4.2 Polynomial system

Murphy and Robshaw claimed that recovering an AES key is equivalent to solving particular systems of extremely sparse multivariate quadratic equations by expressing a BES (and hence an AES) encryption as such a system. The problem of solving such systems of equations lies at the heart of several public key cryptosystems and there has been some progress in providing solutions to such problems.

Courtois and Pieprzyk have suggested the use of a system of multivariate quadratic equations over $\mathbb{F}_2$ to analyze the AES. However, such a $\mathbb{F}_2$-system derived directly from the AES is far more complicated than the $\mathbb{F}_{2^8}$-system derived from the BES.
A BES encryption is described by the following system of equation, remembering that the last round differs slightly from the other ones, with $M_B^* = \mathbf{R_B} \cdot Lin_B = Mix_B^{-1} \cdot M_B$,

$$\begin{cases} \mathbf{w} = \mathbf{p} + \mathbf{k}_0 & \\ \mathbf{x}_i = \mathbf{w}_i^{-1} & i = 0, \ldots, 9 \\ \mathbf{w}_i = M_B \mathbf{x}_{i-1} + \mathbf{k}_i & i = 1, \ldots, 9 \\ \mathbf{c} = M_B^* \mathbf{x}_9 + k_1 0 & \end{cases}$$

If we consider the previous equations component wise, let the $(8j+m)^{th}$ component of all the vectors be indicated using the indexes expression (j, m) with $j = 0, \ldots 15$ and $m = 0, \ldots 7$. Under the hypothesis that no 0-inversion occurs (true for the 53% of encryptions and 85% of 128-bit keys), it is possible to expand the above system as follows, for all possible values of j and m,

$$\begin{cases} 0 = w_{0,(j,m)} + p_{(j,m)} + k_{0,(j,m)} & \\ 0 = x_{i,(j,m)} w_{0,(j,m)} + 1 & i = 0, \ldots, 9 \\ 0 = w_{i,(j,m)} + (M_B \mathbf{x}_{i-1})_{(j,m)} + k_{i,(j,m)} & i = 1, \ldots, 9 \\ 0 = c_{(j,m)} + (M_B^* \mathbf{x}_9)_{(j,m)} + k_{10,(j,m)} & \end{cases}$$

The previous system is a collection of simultaneous multivariate quadratic equations which apparently fully describe a BES encryption. A BES encryption can therefore be described as a multivariate quadratic system using 2688 equations over $\mathbb{F}_{2^8}$, of which 1280 are (extremely sparse) quadratic equations and 1408 are linear (diffusion) equations. These equations comprise 5248 terms, made from 2560 state variables and 1408 key variables.

When we consider an AES encryption embedded in the BES framework, we obtain more multivariate quadratic equations because the embedded state variables of an AES encryption are in $\mathbf{B_A}$ and possess the conjugacy property. Now, let $\alpha, \beta \in \mathbb{F}_{2^8}$ be the generic coefficients of M_B and M_B^*, respectively (polynomials modulo $m(t)$). Adding the fact that the above equations should be valid for the $\mathbf{B_A}$ subset, this is, the state vectors must have the conjugation property, we finally have (with $m+1$ considered modulo 8):

$$S = \begin{cases} 0 = w_{0,(j,m)} + p_{(j,m)} + k_{0,(j,m)} & \\ 0 = w_{i,(j,m)} + k_{i,(j,m)} + \sum_{(j',m')} \alpha_{(j,m),(j',m')} x_{i-1,(j',m')} & i = 1, \ldots, 9 \\ 0 = c_{(j,m)} + k_{10,(j,m)} + \sum_{(j',m')} \beta_{(j,m),(j',m')} x_{9,(j',m')} & \\ 0 = x_{i,(j,m)} w_{0,(j,m)} + 1 & i = 0, \ldots, 9 \\ 0 = x_{i,(j,m)}^2 + x_{i,(j,m+1)} & i = 0, \ldots, 9 \\ 0 = w_{i,(j,m)}^2 + w_{i,(j,m+1)} & i = 0, \ldots, 9 \end{cases}$$

Let S_ℓ, $\ell = 1, \ldots, 6$ indicate the equations in the ℓ^{th} line of the above system for all the possible values of i, j and m, and I_ℓ the ideal they generate. As we see, the system is very sparse, with $S' = \{S_1, S_2, S_3\}$ linear and the remaining equations in $S'' = \{S_4, S_5, S_6\}$ quadratic. We can compare the number of equations with the number of variables that occur in the previous system:

Line	Number of equations
S_1	$16 \cdot 8 = 128$
S_2	$9 \cdot 16 \cdot 8 = 1152$
S_3	$16 \cdot 8 = 128$
S_4	$10 \cdot 16 \cdot 8 = 1280$
S_5	$10 \cdot 16 \cdot 8 = 1280$
S_6	$10 \cdot 16 \cdot 8 = 1280$
S	**Total** $= 5248$

Block	Number of variables
$\mathbf{k}$	$11 \cdot 16 \cdot 8 = 1408$
$\mathbf{x}$	$10 \cdot 16 \cdot 8 = 1280$
$\mathbf{w}$	$10 \cdot 16 \cdot 8 = 1280$
	Total $= 3968$

Remark 3.4.1. Given the BES algebraic formulation, assuming the system S to be correct, it is clear that an efficient method for the solution of this type of multivariate quadratic system would give a cryptanalysis of the AES with potentially very few plaintext-ciphertext pair. While the problem of solving generic large systems of multivariate equations of degree greater than one over a finite field is known to be NP-complete, it is not entirely unlikely that a technique can be developed which exploits the particular algebraic structure of the AES and BES systems. In the previous chapter we presented a few approaches for solving such systems.

Furthermore, the AES key schedule can be expressed as a similar multivariate quadratic system. In its most sparse form, the key schedule system uses 2560 equations over $\mathbb{F}_{2^8}$, of which 960 are (extremely sparse) quadratic equations and 1600 are linear equations. These key-schedule equations comprise 2368 terms made from the 2048 variables, of which 1408 are basic key variables and 640 are auxiliary variables.

3.4.3 Toli-Zanoni's remark

Toli and Zanoni, in [TZ05], rewrote the system S modifying the way the systems are presented, doing some substitutions and performing Gröbner basis computations to obtain the final systems. In order to do this, they removed the imposed restriction about inversion, substituting S_4 with an equation expressing the true definition of the general inversion in $\mathbb{F}_{2^8}$ and using the field equations.

In this way, they obtained the system of this kind:

$$S^* = \begin{cases} 0 = w_{0,j} + p_j + k_{0,j} \\ 0 = w_{i,j} + k_{i,j} + \sum_{(j',m')} \alpha_{(j,0),(j',m')} x_{i-1,j'}^{\omega_{m'}} \quad i = 1, \dots, 9 \\ 0 = c_j + k_{10,j} + \sum_{,m'} \lambda_{m'} w_{9,j'}^{\omega_{m'}} \end{cases}$$

The system has $16 + 9 \cdot 16 + 16 = 176$ equations in $11 \cdot 16 + 10 \cdot 16 = 336$ variables. Therefore, it is actually under-defined and so it cannot be solved, unless by adding enough equations to describe completely the encryption. For example, Toli and Zanoni show that it is enough to add the field equations. But as soon as they are added, any known method to solve the system is forced to generate huge intermediate polynomials (e.g. computing a few S-polynomials and making inter-reductions).

3.5 Dual ciphers

Suppose that we take invertible mappings f, g and h. Then, there exists a *dual cipher* $\overline{AES}$ such that:

$$AES_k(\mathsf{x}) = f^{-1}\overline{AES}_{g(k)}(h(\mathsf{x}))$$

where k is the cipher key and x the plaintext.
This means that the dual cipher is equivalent to the original cipher in the sense that it produces the same ciphertext, for a given plaintext and a given key, by applying fixed permutations on the plaintext, the key and the output of the dual cipher. As a consequence, one can implement and cryptanalyze the dual cipher instead of the original cipher.

Definition 3.5.1. *Two ciphers ϕ and ϕ' are called* dual ciphers *if they are isomorphic, i.e. if there exist invertible transformations h_k, h_x and h_y such that, $\forall \mathsf{x}, k$,*

$$h_\mathsf{y}(\phi_k(\mathsf{x})) = \phi'_{h_k(k)}(h_\mathsf{x}(\mathsf{x})).$$

There are two types of trivial dual ciphers. Every cipher is dual to itself with the identity transformations. The addition of non-cryptographic invertible initial and final transformations creates a trivial dual cipher.
An interesting question is the following: do non-trivial dual ciphers of widely-used (or well-known ciphers) exist? However, the existence of dual ciphers for a specific cipher does not necessarily mean that the security of the cipher is compromised. Dual ciphers can actually be used to strengthen the cipher against side-channel attacks.
An interesting extension of dual ciphers are *semi-dual* ciphers:

Definition 3.5.2. *A cipher ϕ' is called a* semi-dual *cipher of ϕ if there exist transformations h_k, h_x, h_y such that, $\forall \mathsf{x}, k$,*

$$h_\mathsf{y}(\phi_k(\mathsf{x})) = \phi'_{h_k(k)}(h_\mathsf{x}(\mathsf{x}))$$

where h_y, h_x and h_y are not necessarily invertible (and even not necessarily length-preserving).

Semi-dual ciphers potentially reduce the plaintext, the ciphertext and the key spaces. Thus they may allow to develop efficient attacks on their original ciphers.

Biham and Barkan in [BB02a] identified 240 dual ciphers for AES. No weakness of these dual ciphers has been reported. They investigated the importance of the specific choice of constants in a cipher. They asked what happens if they replace all

the constants in AES (including the irreducible polynomial, the coefficients of the MixColumns operation, the affine transformation in the S-box). They showed that such replacements create dual ciphers (which are isomorphic to the original one).
The choice of the specific constants raises some natural questions for the cryptanalyst:

1. Does the choice of constants provides the highest level of security?
2. Is there another choice of constants that provides the same or higher level of security?
3. Does the choice of constants have any relevance to the security of the cipher, i.e., there is a suitable choice of constants that provides a lower level of security?

Biham and Barkan [BB02b] presented the following kind of dual ciphers for the AES:

- Square of AES;
- dual ciphers with the irreducible polynomial replaced by primitive polynomials;
- logarithms of AES;
- Non trivial Self-Dual cipher (they can be attacked in a time faster than exhaustive search).

A similar concept, called Rijndael-GF, is defined by Deamen and Rijmen in [DR98]: it is demonstrated that all the ciphers of the Rijndael-GF family have exactly the same security level against differential and linear cryptanalysis.
Many of the dual ciphers are alternative AES representations where the mapping of state and key spaces are algebra isomorphisms of the AES state space algebra. The resultant ciphers are isomorphic to the AES. The finite field $\mathbb{F}_{2^8}$ can be constructed as an extension field of any of its subfields. Isomorphic representations of $\mathbb{F}_{2^8}$ can thus be constructed from the chain of subfields

$$\mathbb{F}_2 \subset \mathbb{F}_{2^2} \subset \mathbb{F}_{2^4} \subset \mathbb{F}_{2^8}.$$

3.5.1 Square dual ciphers

Given a cipher ϕ, we define the cipher ϕ' by modifying the constants of ϕ. In terms of Definition 3.5.1, we set $h_k = h_{\mathsf{x}} = h_{\mathsf{y}} = x^2$, where x^2 is squaring each byte of x in $\mathbb{F}_{2^8}$. The notation k^2 and x^2 denote the square operation of each byte of k and x (and similarly for any other byte vector). We define ϕ^2 such that $\phi^2_{k^2}(\mathsf{x}^2) = (\phi_k(\mathsf{x}))^2$.

We note that all the operation not involving constants remain unchanged.
In the affine transformation, A is replaced by QAQ^{-1}, where in the case of AES the matrix Q is the following

$$Q = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

It easy to show that $Qx = x^2$ (multiplying by the Boolean matrix Q is actually squaring). From now on, A^2 denote QAQ^{-1}, as for any x,

$$QAQ^{-1}x^2 = QAx = (Ax)^2.$$

The matrices Q and Q^{-1} depend on the irreducible of $\mathbb{F}_{2^8}$. Finally, we replace $S(x)$ with $S^2(x)$, where $S^2(x)$ is defined as

$$S^2(x) = QS(Q^{-1}x).$$

We can now define the dual cipher ϕ^2 of a cipher ϕ: we take the specifications of the cipher ϕ, raise all the constants in the cipher to their second power, replace matrices A by $A^2 = QAQ^{-1}$ and replace $S(x)$ by $S^2(x) = QS(Q^{-1}x)$.
If we take AES as an example of ϕ, the polynomial of the MixColumns operation is replaced by a polynomial which coefficients are the squaring of the official AES polynomial. The affine transformation $Ax+b$ is replaced by the affine transformation $A^2x + b^2 = QAQ^{-1}x + b^2$. Then, it is not difficult to show that ϕ and ϕ^2 are dual ciphers, with $f(x) = x^2$.

Theorem 3.5.3. *For any k and x,*

$$\phi^2_{k^2}(\mathsf{x}^2) = (\phi_k(\mathsf{x}))^2.$$

Proof. See [BB02b]. □

Remark 3.5.4. The cipher $\phi^4 = (\phi^2)^2$ is a dual cipher of ϕ^2 and thus also of ϕ. Moreover, all ciphers ϕ^{2^i} (for all i), i.e. $\phi, \phi^2, \phi^4, \phi^8, \phi^{16}, \phi^{32}, \phi^{64}$ and ϕ^{128} are all dual ciphers of each other (there exist 8 such ciphers as $\phi^{2^8} = \phi$).
It is interesting to note that any variant of the AES possesses these dual ciphers, *independently* of the key size, the block size, the number of rounds and even the arrangement of operations in the cipher.

3.5.2 Dual ciphers modifying the irreducible polynomial

We show that it is *irrelevant* if the irreducible polynomial is primitive or not, due to existence of dual ciphers of AES with any of above irreducible polynomials.
Replacing the irreducible polynomial creates an isomorphic $\mathbb{F}_{2^8}$ field; the isomorphism function is linear. We denote this linear function by R.
Let x be a binary vector representing an element under AES's irreducible polynomial $g(x)$. The representation of x is under another irreducible polynomial $\hat{g}(x)$ is given by $R \cdot x$, where R is an 8×8 binary matrix. Moreover, the matrix R is always of the form $R = (1, a, a^2, a^3, a^4, a^5, a^6, a^7)$, where the columns a^i are computed modulo the irreducible polynomial $\hat{g}(x)$.

We define a new cipher ϕ^R using the new irreducible polynomial $\hat{g}(x)$, such that ϕ^R is a dual cipher of ϕ, with $h_k(x) = h_{\mathsf{x}}(x) = h_{\mathsf{y}}(x) = R \cdot x$.
We define ϕ^R using the matrix R in the same way as when we used the matrix Q to define the square dual cipher. As a result, the operations in ϕ^R are identical to the operations in ϕ, up to a change of constants. To fully specify ϕ^R replace Q with R in the previous section and replace x^2 with $R \cdot x$. The proof of duality follows.

Remark 3.5.5. Note that the matrix Q is actually a special case of the R matrix, where $\hat{g}(x) = g(x)$. For each irreducible polynomial we can define its eight square dual ciphers. Since there are 30 irreducible polynomials, we get that there are 240 dual cipher for each cipher satisfying suitable operations.The full description of these 240 dual ciphers of AES can be found in *The Book of Rijndaels* [BB02a].

Due to the existence of a dual cipher with any irreducible polynomial, we conclude that the choice of the irreducible polynomial of AES is arbitrary. In particular, there is no advantage in selecting a primitive polynomial over the current polynomial of AES.

3.5.3 Logarithmic dual ciphers

In this section we sketch another family of dual ciphers: the *logarithmic dual cipher*. Let g be a generator of the multiplicative group of $\mathbb{F}_{2^8}$. Since the cipher works on elements of $\mathbb{F}_{2^8}$ we can write any element x as an exponent of g, i.e. $x = g^i$, except for $x = 0$, where we set $g^{-\infty}$. In the logarithmic cipher we use the logarithm representation of the element instead of the polynomial representation used in the original description of the cipher.
Let x and y be elements of $\mathbb{F}_{2^8}$ and let $i = \log_g x$ and $j = \log_g y$. Barkan and Biham showed that the cipher $\phi^{\log}$ is a dual cipher of ϕ, where $h_k(x) = h_{\mathsf{x}}(x) = h_{\mathsf{y}}(x) = \log_g x$. The logarithmic dual cipher is defined by taking the specifications of the cipher and replacing the operations in a suitable way, as proposed in [BB02b].

The following theorem suggests that if x is the plaintext, k is the key and y is the result of encrypting x under the key k with cipher ϕ, then the result of encrypting $\log_g(\mathsf{x})$ under the key $\log_g(k)$ with the cipher $\phi^{\log}$ is necessarily $\log_g(\mathsf{y})$.

Theorem 3.5.6. *Let g be a generator in $\mathbb{F}_{2^8}$. For any K and x:*

$$(\phi_{\log_g k})^{\log}(\log_g \mathsf{x}) = \log_g(\phi_k(\mathsf{x})).$$

where $\log_g X$ denotes the logarithm of each byte of X, where X is one of x, y, or k.

Remark 3.5.7. Note that the non-linear part of the SubBytes transformation of AES in the log dual cipher, i.e. finding the multiplicative inverse of an element, becomes very simple (and linear). This operation is replaced by negation in the logarithmic dual cipher:

$$x \to i \Leftrightarrow x^{-1} \to -i.$$

How does the 240 mentioned representation of AES affect the number of logarithmic dual ciphers? The group of 240 representations of AES has a single group of 128 logarithmic dual ciphers. Choosing a generator g in AES's representation generates the same dual cipher as choosing the generator $R \cdot g$ in another dual cipher. Therefore, the number of logarithmic dual cipher is the same as the number of generators, i.e., there are only 128 logarithmic dual ciphers.

3.5.4 Self-dual ciphers

We mention that any cipher is trivially dual to itself. However, it is possible to find ciphers that are self-dual in a non trivial way. One such interesting case of self dual ciphers can be derived from square dual ciphers.
Let ϕ be a square self dual cipher. It follows that $(\phi_k(\mathsf{x}))^2 = \phi_{k^2}(\mathsf{x}^2)$. In other words, by encrypting the square of x by the square of k under the cipher, we get the square of the original ciphertext. For that, we require that each constant is the square of itself.

The self-duality property of a cipher can be used to mount an attack, which reduces the complexity of exhaustive search by a factor about 8 for a square dual cipher in the case above (or by a factor of the number of the self-duals in the more general case). For example, if the key size is 128 bit, exhaustive search requires 2^{128} applications of the cipher ϕ, and the attack we proposed requires about 2^{125} applications of ϕ using 8 chosen plaintexts. If we consider the expected time to complete the attack, exhaustive search takes about 2^{127} applications of ϕ, and our attack takes about 2^{124} applications of ϕ. The attack takes advantage of cycles of keys under the squaring operation. The algorithm of the attack was presented in [BB02b]. It is also interesting to remark that the number of rounds of the cipher does not affect the complexity of this attack (in terms of the number of applications of ϕ). AES is not a self-dual cipher (except in a trivial sense).

A possible application of dual ciphers is for developing differential and linear attacks. In such cases the insight gained from the dual ciphers can be used to attack the dual cipher, an attack which can be easily transformed to the original. A possible example for such insight might be the simplification of the affine transformation in the S-box to a triangular matrix, which reduces the effect of modifying bits in the input on the resultant output of this transformation. Another interesting application of dual ciphers might be an optimization of the speed of the cipher, as in some cases the dual cipher might actually be faster to compute than the original cipher. For example, many ciphers include multiplications by constants. The Hamming weight and the size of the constants has implications on the implementation efficiency. Thus, finding a *more efficient* dual cipher might be a good optimization strategy. Also, in some cases encryption might be fastest using one dual cipher, and decryption be fastest using another dual cipher.

The existence of dual ciphers can also be used to protect implementation against fault analysis and power analysis, by selecting a different dual cipher at random each time an encryption or decryption is desired.

Wu, Lu and Laih in 2004 [WLL04] generalized the dual AES and proposed a complete setup procedure to determine all dual ciphers and a hardware implementation of AES based on the combination of dual cipher and composite field. Moreover, they demonstrated that their AES design not only offer better performance and smaller area requirement than the design proposed by Wolkerstorfer et al. ([WOL02]) in which uses a composite field only. Their result also confirms Barkan et al's conjecture that it is possible to design an AES cipher more efficiently than ever.

3.6 S-boxes equivalence

Biryukov, De Canniere, Braeken and Preneel presented in [BDCBP03] two algorithms for solving the linear and the affine equivalence problem for arbitrary permutations (S-boxes). [2] For a pair of $n \times n$ permutations, the complexity of the linear equivalence algorithm (LE for short) is $O(n^3 2^n)$. The affine equivalence algorithm (AE for short) has complexity $O(n^3 2^{2n})$. The algorithms are efficient and allow to study linear and affine equivalences for bijective S-boxes of almost all size. (LE is efficient up to $n \leq 32$).

[2]The principle of such algorithms has been presented earlier in [PGC98]. We note that the isomorphism of polynomials is actually (almost) the same problem as the Affine Equivalence.

Using these tools, new equivalent representations are found for a variety of ciphers like Rijndael, DES, SERPENT and other ciphers. The algorithms are furthermore extended for the case of non-bijective n to m-bit S-boxes with a small value of $|n-m|$ and for the case of almost equivalent S-boxes.
An efficient algorithm tool allows to study the properties of a whole equivalence class by analyzing a single representative.
The motivations to study this problem are

- deeper understanding of Rijndael,
- recent interest in potential algebraic attacks,
- the discovery of a variety of equivalent representation.

Such representations help to describe ciphers with simpler systems of low-degree equations, allow more efficient implementations and are very useful in the design of countermeasures against side-channel attacks. They provided algorithms that can quickly test if two S-boxes S_1 and S_2 are equivalent; in other words they tested if there exist a (linear or affine) mappings A_1 and A_2 such that

$$A_2 \circ S_1 \circ A_1 = S_2.$$

LE and AE will either return the mappings A_1 and A_2 or detect that the S-boxes are inequivalent, within $O(n^3 2^n)$ for LE and $O(n^3 2^{2n})$ for AE. They solved the affine equivalence problem using a method of interest in itself that consists of finding unique representatives for the linear equivalence. The efficiency of the given algorithms allows to find linear equivalences for n up to 32 and affine equivalences for n up to 17, which covers most of the S-boxes used in modern symmetric primitives and allows to study partial function composed of several S-boxes and portions of the mixing layers. They extended their results for the case of non-bijective S-boxes with n input bits and m output bits when the input/output deficiency $|n-m|$ is small.
Another interesting extension is the search for almost equivalent S-boxes, which is as efficient as the basic algorithms. This allows to check quickly if a certain S-box is close to the set of affine functions or two S-boxes, one with unknown structure and the other with known algebraic structure are almost equivalent. This approach induces an interesting metric in the space of affine equivalence classes of S-boxes.
Using this toolbox of algorithms they found that many S-boxes of popular ciphers are self affine equivalent, which allows to produce equivalent representations of these ciphers, like AES, DES, SERPENT, etc.

3.6.1 Linear equivalence

Let us consider the problem of checking linear equivalence between two permutations (S-boxes) S_1 and S_2. The main problem consist of finding two invertible linear mapping L_1 and L_2 such that $L_1 \circ S_1 \circ L_1 = S_2$.
We give the following naive approach:

- to guess one of the mappings (L_1 for example), such that $L_2 = S_2 \circ L_1^{-1} \circ S_1^{-1}$;
- to check if it is a linear, invertible mapping. There are $O(2^{n^2})$ choices of invertible linear mappings over $n-$bit vector. For each guess one will need about n^3 steps to check for linearity and invertibility using Gaussian elimination.

For $n \leq 32$ (which is of main practical interest) we can use 32 bit processor instructions to bring the complexity to n^2 steps. Complexity: $O(N^3 2^{n^2})$ steps.
Improving the naive approach:

- we need only n equations in order to check L_2 for invertibility and linearity;
- if one guesses only $\log_2 n$ vectors from L_1 one may span a space of n points (by trying all linear combinations of the guessed vectors);
- evaluate the results through L_1, S_1 and S_2 and have n constraints required to check for linearity of L_2;
- if the n new equations are not independent one will need to guess additional vectors of L_1.

Complexity: Such an algorithm would require guessing of $n \log_2 n$ bits of L_1 and the total complexity would be $O(n^3 2^{n \log n})$.
In Biryukov et al's algorithm [BDCBP03] there are two ideas. The first one, a needlework effect, in which guesses of portions from L_1 provided us with free knowledge of the values of L_2. These new values from L_2 permit to extract new free information about L_1, etc.. This process is supported by a second observation, which they call exponential amplification of guesses, which happens due to the linear (affine) structure of the mappings. The idea is that knowing k vectors from the mapping L_1, we know 2^k linear combinations of these vectors for free.

3.6.2 Affine equivalence

It is possible to generalize the equivalence problem to the affine case. In this case they wanted an algorithm that takes two $n \times n$- bit S-boxes S_1 and S_2 as input, and checks whether there exists a pair of invertible affine mappings A_1 and A_2 such that $A_2 \circ S_1 \circ A_1 = S_2$. Each of these affine mappings can be expressed as a linear transform followed by an addition, which allows us to rewrite the affine equivalence relation as $B^{-1}S_1(A \cdot x \oplus a) \oplus b = S_2(x)$, $\forall x \in \{0,1\}^n$ with A and B invertible $n \times n$ bit linear mapping and with $n-$bit constants a and b.

3.6.3 Extension

Self-Equivalent S-boxes: the affine equivalence algorithm was designed to discover equivalence relations between different S-boxes, but nothing prevents us from running the algorithm for a single S-box. In this case, the algorithm will return affine mappings A_1 and A_2 such that $A_2 \circ S \circ A_1 = S$. The number of different solutions for this equation (denoted by $s \geq 1$) can be seen as a measure for the symmetry of the $S-$box. We call $S-$boxes that have at least one non trivial solution ($s > 1$) self equivalent $S-$box.

Equivalence of Non invertible S-boxes: it is possible to extend the equivalence problem to the non-invertible n to m-bit S-boxes with $m < n$. This problem becomes the following: find an $(n \times n)$-bit affine mapping A_1 and an $(m \times m)$-bit affine mapping A_2 such that

$$A_2 \circ S_1 \circ A_1 = S_2$$

for two given $(n \times m)$-bit $S-$boxes S_1 and S_2.
The main problem when trying to apply the algorithms described above, is that the exponential amplification process explicity relies on the fact that the S-boxes are invertible. In cases where the difference $n - m$ is too large, slightly adapted versions of the algorithms still appear to be very useful.
The difference between the extended and the original algorithm resides in the way information about A_1 is gathered. In the original algorithm, each iteration yields a number of additional distinct points which can directly be used to complete the affine mapping A_1. This time, the $S-$boxes are not uniquely invertible and the information obtained after each iteration will consist of two uncolored sets of about 2^{n-m} values which are known to be mapped onto each other. In order to continue, the algorithm first needs to determine which are the corresponding values in both sets. This can be done exhaustively if 2^{n-m} is not too large, say less than eight. Once the order has been guessed, 2^{n-m} points are obtained. Since slightly more than n points should suffice to reject a candidate for the representative, one would expect that the total complexity is about

$$n^3 2^n (2^{n-m}!)^{\frac{n}{2^{n-m}}}.$$

Almost Affine Equivalent $S-$boxes: another interesting problem related to equivalence is the problem of detecting whether two $S-$boxes are almost equivalent. The $S-$boxes S_1 and S_2 are called almost equivalent if there exist two affine mappings A_1 and A_2 such that $A_2 \circ S_1 \circ A_1$ and S_2 are equal, except in a few points.

3.6.4 Equivalences in the AES cryptosystem

When this AE tool is run for the 8-bit S-box used, as many as 2040 different self-equivalence relations are revealed. Considering the fact that the Rijnadael S-box is defined as $S(x) = A(x^{-1})$ with A a fixed affine mapping (not to be confused with A_1 or A_2), it is possible to derive a general expression for all pairs of affine mappings A_1 and A_2 satisfying $A_2 \circ S \circ A_1 = S$:

$$A_1(x) = [a] \cdot Q^i \cdot x$$
$$A_2(x) = A(Q^{-i} \cdot [a] \cdot A^{-1}(x))$$

where $[a]$ denotes the (8×8)-matrix that corresponds to a multiplication by $a \in \mathbb{F}_{2^8}$ $(a \neq 0)$, Q denotes the (8×8)-matrix that performs the squaring operation in $\mathbb{F}_{2^8}$ and $0 \leq i \leq 8$.
Since i takes on 8 different values and there are 255 different choices for a, we obtain exactly 2040 different solutions, which confirms the outputs of the AE algorithm.
The existence of these affine self-equivalences in Rijndael implies that we can insert an additional affine layer before and after the S-boxes without affecting the cipher. Moreover, since the mixing layer of Rijndael only consists of additions and multiplications with constants in $\mathbb{F}_{2^8}$ and since $[a] \cdot Q^i \cdot [c] = [c^{2^i}] \cdot [a] \cdot Q^i$, we can easily push the input mapping A_1 through the mixing layer. This allows us to combine A_1 with the output mapping of a previous layer of S-boxes, with the plaintext, the round constants or with the key. The resulting ciphers are generalizations of the eight squares of Rijndael, obtained in somewhat different way by Barkan and Biham. By modifying the field polynomial used in these 2040 ciphers, one should be able to expand the set of 240 dual ciphers presented in [BB02b] to a set of 61200 ciphers. Note that these ideas also apply to a large extent to other ciphers that use S-boxes based on power functions.

Remark 3.6.1. For a Boolean function f, we say that the non-linearity of f is ν if, for any affine function α, f and α differ in at least ν points. The AES S-box's non-linearity is the maximum non-linearity for a Boolean function $f : (\mathbb{F}_2)^8 \rightarrow (\mathbb{F}_2)^8$, which means that we cannot hope to find a linear/affine approximation.

Part II

Our Results

A new representation

In Chapter 3 we described how two different ways of representing the same cipher AES, like BES [MR02] or Dual Ciphers [BB02b], could be useful for the cryptanalysis. Several representations of this kind that exploit the structure of the AES cipher have been proposed and the reader can find them in literature, see for example [CMR07]. In this chapter we construct two new representations of “AES-like” ciphers.
In Section 4.1 we consider some permutations acting on a given set Ω. We want to enlarge Ω to a set W such that:

1. W is endowed with a vector space structure;
2. the permutations can be extended to act linearly on the whole W.

In each of Section 4.2 and Section 4.3 we provide one specific representation of AES-like ciphers. That in Section 4.3 can be seen as an improvement of the former. Both of them are applied to AES, PRESENT and SERPENT. We highlight advantages and weaknesses of those cryptosystems, according to our representations. In particular, in Section 4.2 we consider $\Omega = V$ as a vector space and we want to find an embedding $V \hookrightarrow W$ such that the S-boxes and the key-additions become linear. However, in this way we lose the linearity of the Mixing Layer λ and so, in Section 4.3, we make a larger embedding where the linearity of λ is recovered, without losing the linearity of the key addition. We do lose the linearity of the S-boxes, but their non-linearity is kept low.

In Section 4.4 we report other thinkable representations, that unfortunately are impractical. The main objective in these constructions is to identify the right compromise between computational feasibility and quantity of information that can be obtained.

Then, in Section 4.5 we prove the fact, using classical and easy arguments, that it is impossible to embed the AES cipher into a linear cipher, unless one uses a huge-dimensional vector space (and so this embedding is useless in practice).

Finally, the last section contains some results on how our representation could achieve a weaker notion of linearity.

4.1 Some preliminary results

Let Ω be a set such that $|\Omega| = n$, let $\mathrm{Sym}(\Omega)$ be the symmetric group on Ω and let W be a vector space over a field $\mathbb{F}$ (not necessarily a finite field).

Definition 4.1.1. *Let $G \leq \mathrm{Sym}(\Omega)$. An injective map $\phi : \Omega \to W$ is a* **space embedding** *with respect to the group G if $\forall \sigma \in G\ \exists A_\sigma \in \mathrm{GL}(W)$ s.t. $\phi \circ \sigma = A_\sigma \circ \phi$.*

Moreover, $\phi(\Omega)$ is the set of all admissible vectors (w.r.t. ϕ), the subspace $\langle \phi(\Omega) \rangle$ is the admissible space. Note that since $\phi(\Omega) \subset \langle \phi(\Omega) \rangle$ then $\langle \phi(\Omega) \rangle$ is the smallest subspace containing all admissible vectors. Generally speaking, $|\langle \phi(\Omega) \rangle| >> |\phi(\Omega)|$.

Note that the regular representation, defined in Section 1.1, can be considered as a space embedding $\phi : \Omega \to W$ with respect to the group $G = \mathrm{Sym}(\Omega)$, where $\dim(W) = |\Omega| = n$ and $\phi : \omega \mapsto b_\omega$ with $\{b_\omega\}_{\omega \in \Omega}$ a basis of W. Also, $W = \langle \phi(\Omega) \rangle$.

A space embedding permits to construct a faithful representation of G, as explained in the next proposition.

Proposition 4.1.2. *Let $\alpha : \Omega \to W$ be a space embedding with respect to G. Suppose that $\forall \sigma \in G \quad \exists! A_\sigma \in \mathrm{GL}(W) \quad$ s.t. $\quad \phi \circ \sigma = A_\sigma \circ \phi$. Then*

1. *we can define a map $\tilde{\phi} : G \to \mathrm{GL}(W)$, where $\tilde{\phi}(\sigma) = A_\sigma$, for any $\sigma \in G$;*
2. *$\tilde{\phi}$ is a group homomorphism.*

Proof. 1. Obvious.
2. We have to prove that $\tilde{\phi}(\sigma\sigma') = \tilde{\phi}(\sigma)\tilde{\phi}(\sigma')$ for all $\sigma, \sigma' \in G$, i.e. $A_{\sigma\sigma'} = A_\sigma A_{\sigma'}$. Using Definition 4.1.1, the following equality holds

$$A_{\sigma\sigma'}(\phi(\omega)) = \phi((\sigma\sigma')(\omega)) = \phi(\sigma(\sigma'(\omega))).$$

Since

$$A_\sigma A_{\sigma'}(\phi(\omega)) = A_\sigma(\phi(\sigma'(\omega))) = \phi(\sigma(\sigma'(\omega))),$$

we conclude that $A_{\sigma\sigma'} = A_\sigma A_{\sigma'}$, for all $\omega \in \Omega$. □

Remark 4.1.3. In Definition 4.1.1 we require only that A_σ exists, however in Theorem 4.1.2 we see that it is also unique.

For example, for the regular representation any permutation $\sigma \in \mathrm{Sym}(\Omega)$ defines a permutation $\sigma \in \mathrm{Sym}(\{b_\omega\}_{\omega \in \Omega})$ and so it defines a unique $A_\sigma \in \mathrm{GL}(W)$, which can be represented as a permutation matrix.

Now, we are interested in a special case of space embedding where the set Ω is a vector space $V = (\mathbb{F}_2)^r$ and W is the vector space $(\mathbb{F}_2)^s$, with $s > r$. For any $1 \leq i \leq s$, let $\mathbf{e}_i \in W$:

$$\mathbf{e}_i = (0, \ldots, 0, \underset{\substack{\uparrow \\ i}}{1}, 0, \ldots, 0)\,.$$

Let $\sigma \in \mathrm{Sym}(V)$ be any permutation over $(\mathbb{F}_2)^r$. We want to embed V into W by an injective map α and to extend σ to a permutation $\sigma' \in \mathrm{Sym}(W)$ as shown in the following commutative diagram:

$$\begin{tikzcd} V \arrow[r, "\alpha"] \arrow[d, "\sigma"] \arrow[dr, phantom, "\circlearrowleft"] & W \arrow[d, "\sigma'"] \\ V \arrow[r, "\alpha"] & W \end{tikzcd}$$

In order to do this, we have to define the permutation $\sigma' \in \mathrm{Sym}(W)$. We say that σ' is an extension of σ. We seek a σ' that is linear on W. The following definition will be useful:

Definition 4.1.4. *Let $\sigma \in \mathrm{Sym}(V)$ and α be an injective map $\alpha : V \to W$. We say that σ is* **linearly extendible** *(via α) if $\forall \{v^i\}_{i \in I} \subset V$ we have*

$$\sum_{i \in I} \alpha(v^i) = 0 \iff \sum_{i \in I} \alpha(\sigma(v^i)) = 0.$$

Remark 4.1.5. Since we are considering the finite field $\mathbb{F}_2$, we note that σ is linearly extendible (via α) if $\forall \{v^i\}_{i \in I} \subset V$ such that $\sum_{i \in I} \alpha(v^i) = 0$ we have $\sum_{i \in I} \alpha(\sigma(v^i)) = 0$. In fact, an injective map defined on the set

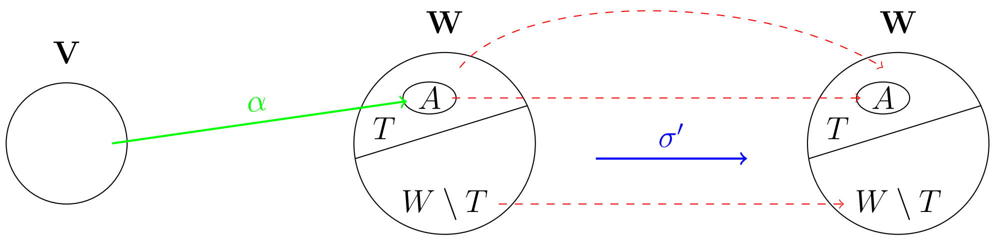
$$\{\{v^i\}_I \subset V \mid \sum_{i \in I} \alpha(v^i) = 0\}\}$$

into the set

$$\{\{\sigma(v^i)\}_I \subset V \mid \sum_{i \in I} \alpha(\sigma(v^i)) = 0\}\}$$

is a bijective map, since the cardinality of the two finite sets is the same.

Let $\alpha : V \to W$ be a space embedding. Let $A = \mathrm{Im}(\alpha) = \alpha(V)$ and let $T = \langle A \rangle$ be the subspace (the admissible space) of W linearly generated by A. Since $\sigma'(\alpha(v)) = \alpha(\sigma(v))$, $\forall v \in V$, we require that $\sigma'(A) = A$.



In order to specify the behavior of σ' on $(T \setminus A)$, which is the space of non-admissible vectors in the admissible space, we have to consider two different cases:

(a) suppose that σ is linearly extendible. Let $t \in T$, we must have $t = \sum_{1 \leq j \leq \iota} a^j$, with $\iota \geq 1$, with $\{a^j\}_{1 \leq j \leq i} \subset A$, $a^j = \alpha(v^j)$ (with $1 \leq j \leq \iota$ and $v^j \in V$). Then we define
$$\sigma'(t) = \sum_{1 \leq j \leq \iota} \sigma'(a^j) = \sum_{1 \leq j \leq \iota} \alpha(\sigma(v^j));$$

(b) in case σ is not linearly extendible, we define $\sigma'_{|_{T \setminus A}} = \mathrm{id}_{T \setminus A}$.

We now define σ' on $W \setminus T$ according to the two previous cases (i.e. depending on the behavior of σ on A).
In case (a), let τ be the dimension of the subspace T. We consider any subset B of $\{\mathbf{e}_1, \dots, \mathbf{e}_s\}$ such that $|B| = s - \tau$ and W is the direct sum $W = T \oplus \langle B \rangle$. It is obvious that B exists. Let $w \in W$, then $w = w_T + w_B$ with $w_T \in T$ and $w_B \in \langle B \rangle$. Finally, we define
$$\sigma'(w) = \sigma'(w_T) + w_B.$$
In case (b) we define $\sigma'_{|_{W \setminus T}} = \mathrm{id}_{W \setminus T}$.

Lemma 4.1.6. *If σ is linearly extendible, then $\sigma' \in \mathrm{GL}(W)$.*

Proof. We first show that σ' is well-defined on T. Let $t = \sum_I a^i$ and $t' = \sum_J a^j$ and suppose that $t = t'$. Since σ is linearly extendible, we have the following
$$0 = t + t' = \sum_I a^i + \sum_J a^j = \sum_I \alpha(v^i) + \sum_J \alpha(v^j) = \sum_{I \cup J} \alpha(v^i)$$
$$\sigma'(t) + \sigma'(t') = \sum_I \sigma'(a^i) + \sum_J \sigma'(a^j) = \sum_I \alpha(\sigma(v^i)) + \sum_J \alpha(\sigma(v^j)) = \sum_{I \cup J} \alpha(\sigma(v^i)) = 0$$
We now show that σ' is linear on T. Let $t_i = \sum_h a_h^{(i)}$. We have to show that $\sigma'(\sum_i t_i) = \sum_i \sigma'(t_i)$. Clearly,
$$\sigma'\big(\sum_i \sum_h a_h^{(i)}\big) = \sigma'\Big(\sum_{i,h} a_h^{(i)}\Big) = \sum_i \sum_h \sigma'(a_h^{(i)}) = \sum_i \Big(\sum_h \sigma'(a_h^{(i)})\Big) = \sum_i \sigma'(t_i)$$
and we have our thesis.
Since σ' is linear on T and T is a finite set, in order to prove that σ' is bijective on T it suffices to show that $\ker \sigma' = 0$. We have (by definition of linearly extendible)
$$0 = \sigma'(t) = \sum \alpha(\sigma(v^j)) \iff 0 = \sum \alpha(v^j) = t$$

Finally, we show the linearity on W. Let $\{w^i\}_{i \in I} \subset W$, we have to show the following equality
$$\sigma'\Big(\sum_{i \in I} w^i\Big) = \sum_{i \in I} \sigma'(w^i). \tag{4.1}$$

Since W is direct sum of T and $\langle B\rangle$, each element w^i in W can be considered as $w^i_T + w^i_B$ and so we can write the following

$$\begin{aligned}\sigma'\Big(\sum_{i\in I} w^i\Big) &= \sigma'\Big(\sum_{i\in I}(w^i_T + w^i_B)\Big) = \sigma'\Big(\sum_{i\in I} w^i_T\Big) + \sum_{i\in I} w^i_B\\ \sum_{i\in I}\sigma'(w^i) &= \sum_{i\in I}\sigma'(w^i_T + w^i_B) = \sum_{i\in I}\sigma'(w^i_T) + \sum_{i\in I} w^i_B.\end{aligned}$$

It easily follows that (4.1) holds if and only if

$$\sigma'\Big(\sum_{i\in I} w^i_T\Big) = \sum_{i\in I}\sigma'(w^i_T).$$

□

Remark 4.1.7. The construction of $\sigma' \in \mathrm{GL}(W)$ from σ linearly extendible (Definition 4.1.4) can be done similarly over any field.

We are now able to prove the main result of this subsection.

Theorem 4.1.8. *Let $W = (\mathbb{F}_2)^r$ and $G \leq \mathrm{Sym}(V)$. An injective map $\alpha : V \to W$ is a space embedding with respect to G if and only if, $\forall\sigma \in G$, σ is linearly extendible.*

Proof. Let α be a space embedding with respect to G. For any fixed $\sigma \in G$, there exists a map $A_\sigma \in \mathrm{GL}(W)$ such that $\alpha\circ\sigma = A_\sigma \circ \alpha$. Now, let $\{w^i\}_{i\in I}$ be a finite set such that $w^i = \alpha(v^i)$ (for any $i \in I$) and $\sum_{i\in I} w^i = 0$. Obviously we have

$$\sum_{i\in I}\alpha(\sigma(v^i)) = \sum_{i\in I} A_\sigma(\alpha(v^i)) = \sum_{i\in I} A_\sigma(w^i) = A_\sigma\Big(\sum_{i\in I} w^i\Big) = 0.$$

The converse immediately follows thanks to the previous lemma. □

Remark 4.1.9. For a fixed α and σ, the map σ' is unique and $\tilde{\alpha} : G \to \mathrm{GL}(W)$ is a representation of G, by Proposition 4.1.2.

Remark 4.1.10. In the following we use A_σ and σ' interchangeably.

4.2 A first representation

We now apply the theory developed in the previous section to a specific space embedding [1] $\varepsilon : V \to W$.

Let us identify $(\mathbb{F}_2)^m$ with the field $\mathbb{F}_{2^m}$, via the quotient map $\mathbb{F}_{2^m} \leftrightarrow \mathbb{F}_2[x]/\langle \mathsf{p}\rangle$, where $\mathsf{p} \in \mathbb{F}_2[x]$ is any primitive polynomial such that $\deg(\mathsf{p}) = m$.

[1]which is called "α" in Section 4.1.

We define a map $\varepsilon' : \mathbb{F}_{2^m} \to (\mathbb{F}_2)^{2^m}$ by means of a primitive element γ of $\mathbb{F}_{2^m}$ (which is a root of p). The map ε' is defined as

$$\varepsilon'(0) = (1, \underbrace{0, \dots, 0}_{2^m-1}) \qquad \varepsilon'(\gamma^i) = (0, \dots, 0, \underset{\substack{\uparrow \\ i+1}}{1}, 0, \dots, 0) \quad \forall 1 \le i \le 2^m - 1\,.$$

Note that $\varepsilon'(1) = \varepsilon'(\gamma^{2^m-1}) = (\underbrace{0, \dots, 0}_{2^m-1}, 1)$.

Let b be a positive integer, let $r = mb$ and $s = 2^m b$. Let $V = (\mathbb{F}_2)^r$ and $W = (\mathbb{F}_2)^s$. We construct our injective map $\varepsilon : V \to W$ in the following way:

$$\varepsilon(v_1, \dots, v_b) = (\varepsilon'(v_1), \dots, \varepsilon'(v_b)) \tag{4.2}$$

for any $v_j \in (\mathbb{F}_2)^m$ ($1 \le j \le b$). Note that ε is a parallel[2] map.
For simplicity of notation, we set $e_1 = \varepsilon'(0) = (1, \underbrace{0, \dots, 0}_{2^m-1})$ and $e_{i+1} = \varepsilon'(\gamma^i)$, for any $1 \le i \le 2^m - 1$. We note that

Lemma 4.2.1. *Suppose that $\sum_{i \in I} e_i = e_h$. Then $h \in I$.*

Proof. It follows from $\mathrm{w}(e_i) = 1$, for all $i \in I$. $\square$

The following lemma is easily proved:

Lemma 4.2.2. *Let I be a finite index multiset such that $\{v^i\}_I \subset V$. For any $1 \le h \le b$ we have $\sum_{i \in I} \varepsilon'(v^i_h) = 0$ if and only if, $\forall i \in I$, $|\{j \in I \mid v^j_h = v^i_h\}|$ is even.*

Proof. Since ε' maps each element of $(\mathbb{F}_2)^m$ into the canonical basis of $(\mathbb{F}_2)^{2^m}$, each $\varepsilon'(v^i_h)$ is a vector such that $\mathrm{w}(\varepsilon'(v^i_h)) = 1$. Considering the following sum in $\mathbb{F}_2$, we have that $\sum_I \varepsilon'(v^i_h) = 0$ if and only if each component is made by an even number of 1, i.e. if and only if each element of the canonic basis that appears in our sum has an even weight. Since ε' is bijective, we have that $|\{j \in I \mid v^j_h = v^i_h\}|$ is even, $\forall i \in I$. $\square$

Proposition 4.2.3. *Let ε as in (4.2). Then $\dim_{\mathbb{F}_2}\left(\langle \mathrm{Im}(\varepsilon) \rangle\right) = 2^m b - (b-1)$.*

Proof. We define the elements $z_{i,j} = (e_1, \dots, e_1, \underset{\substack{\uparrow \\ i}}{e_j}, e_1, \dots, e_1)$, for $1 \le i \le b$ and $1 \le j \le 2^m$. Note that $z_{i,j} \ne z_{h,\ell}$ for $(i,j) \ne (h,\ell)$, except for $z_{11} = z_{21} = \dots = z_{b1}$. We consider the set $\mathcal{B} = \{z_{1,1}\} \cup \{z_{i,j}\}_{j \ge 2,\ 1 \le i \le b}$. For instance, when $m = 2$ and $b = 2$, we have

$$\mathcal{B} = \{(e_1, e_1), (e_1, e_2), (e_1, e_3), (e_1, e_4), (e_2, e_1), (e_3, e_1), (e_4, e_1)\}.$$

Clearly, the cardinality of the set $\mathcal{B}$ is $|\{z_{i,j}\}_{1 \le b,\ 1 \le j \le 2^m}| - |\{z_{i,1}\}_{i \ge 2}| = 2^m b - (b-1)$.

[2]see Section 1.2

We claim that the set $\mathcal{B}$ is a basis for the subspace $\langle \mathrm{Im}(\varepsilon)\rangle$.
First, we prove that $\mathcal{B}$ is a linearly independent set. Suppose $z_{i,j} \in \mathcal{B}$ such that $(i,j) \neq (1,1)$. By definition of $\mathcal{B}$, the element $z_{i,j}$ is the unique element of $\mathcal{B}$ having a vector e_j in position i. Thus, $z_{i,j}$ cannot be the linear combination (i.e. a sum) of any other vectors of $\mathcal{B}$ (see Lemma 4.2.1). Now, we have to consider the element $z_{1,1}$. Let $z_{1,1} = \sum_{(i,j)\in J} z_{i,j}$. W.l.o.g., we can assume by Lemma 4.2.1 that there is $(\bar{i},\bar{j}) \in J$ such that $z_{\bar{i},\bar{j}} = (e_1,\ldots)$. Since $z_{\bar{i},\bar{j}} \neq z_{1,1}$ we can assume w.l.o.g. $z_{\bar{i},\bar{j}} = (e_1, e_{\bar{j}}, e_1, \ldots, e_1)$, i.e. $\bar{i} = 2$. There is no other $z_{i,j}$ having $e_{\bar{j}}$ in the second position. Therefore, the sum $z_{1,1}$ should contain a 1 in component $m+\bar{j}$, which is impossible.

Next, we prove that $\mathcal{B}$ generates $\langle \mathrm{Im}(\varepsilon)\rangle$. To do that, it suffices to prove that every element of $\mathrm{Im}(\varepsilon)$ belongs to the subspace generated by $\mathcal{B}$. If we consider an element $w = (e_{j_1},\ldots,e_{j_b}) \in \mathrm{Im}(\varepsilon)$, we have

$$w = \begin{cases} z_{1,j_1} + \cdots + z_{b,j_b} & \text{if } b \text{ is odd,} \\ z_{1,j_1} + \cdots + z_{b,j_b} + z_{1,1} & \text{if } b \text{ is even,} \end{cases}$$

since

$$\begin{array}{rc}
 & (e_{j_1}, e_1, \ldots, e_1)+ \\
b-1 \left\{ \begin{array}{c} \\ \\ \\ \end{array} \right. & \begin{array}{c} (e_1, e_{j_2}, \ldots, e_1)+ \\ \vdots \\ (e_1, \ldots, e_1, e_{j_b}) = \end{array} \\
\hline
 & (e_{j_1}, e_{j_2}, \ldots, e_{j_b})
\end{array} \quad b \text{ odd} \qquad
\begin{array}{rc}
 & (e_1, e_1, \ldots, e_1)+ \\
 & (e_{j_1}, e_1, \ldots, e_1)+ \\
b-1 \left\{ \begin{array}{c} \\ \\ \\ \end{array} \right. & \begin{array}{c} (e_1, e_{j_2}, \ldots, e_1)+ \\ \vdots \\ (e_1, \ldots, e_1, e_{j_b}) = \end{array} \\
\hline
 & (e_{j_1}, e_{j_2}, \ldots, e_{j_b})
\end{array} \quad b \text{ even}$$

□

Let $\mathcal{A}$ be a subset of the plaintext set $\mathcal{M}$ such that $|\mathcal{A}| = \dim_{\mathbb{F}_2}(\langle \mathrm{Im}(\varepsilon)\rangle) = 2^m b - (b-1)$. Let $a_i \in \mathcal{A}$, $1 \le i \le |\mathcal{A}|$. We construct the $(|\mathcal{A}| \times 2^m b)$-matrix $\mathbf{H}$ such that the i-th row is the image of the parallel map ε applied to the plaintext $a_i \in \mathcal{A}$, for $i \in \{1, \cdots, |\mathcal{A}|\}$:

$$\mathbf{H} = \begin{pmatrix} \varepsilon(a^1) \\ \varepsilon(a^2) \\ \vdots \\ \varepsilon(a^{|\mathcal{A}|}) \end{pmatrix} = \begin{pmatrix} \varepsilon'(a_1^1) & \varepsilon'(a_2^1) & \cdots & \varepsilon'(a_b^1) \\ \varepsilon'(a_1^2) & \varepsilon'(a_2^2) & \cdots & \varepsilon'(a_b^2) \\ \vdots & \vdots & \vdots & \vdots \\ \varepsilon'(a_1^{|\mathcal{A}|}) & \varepsilon'(a_2^{|\mathcal{A}|}) & \cdots & \varepsilon'(a_b^{|\mathcal{A}|}) \end{pmatrix}. \tag{4.3}$$

We would like to determine the expected rank for such a matrix. Generally speaking, for a random $(t \times n)$-matrix with entries in the finite field $\mathbb{F}_q$, we can use the following well known results:

Theorem 4.2.4 ([MMM04]). *Let $t, k, n \in \mathbb{N} \setminus \{0\}$, where $k \leq n$ and $k \leq t$.*

1. *The number of ordered k-tuples of linearly independent vectors in $(\mathbb{F}_q)^n$ is*

$$(q^n - 1)(q^n - q)(q^n - q^2) \cdots (q^n - q^{k-1}).$$

2. *The number of k-dimensional subspaces of $(\mathbb{F}_q)^n$ is given by the q-binomial coefficient*

$$\binom{n}{k}_q = \frac{\prod_{0 \leq i \leq k-1}(q^n - q^i)}{\prod_{0 \leq i \leq k-1}(q^k - q^i)}.$$

3. *The number of $(t \times n)$-matrices of rank k with entries in $\mathbb{F}_q$ is given by the following formula*

$$d_{k,t} = \binom{n}{k}_q \prod_{0 \leq i \leq k-1} (q^t - q^i).$$

We note that

$$\frac{\binom{n}{k-1}_q}{\binom{n}{k}_q} = \frac{q^k - 1}{q^{n-k+1} - 1}, \tag{4.4}$$

since

$$\begin{aligned}
\frac{\binom{n}{k-1}_q}{\binom{n}{k}_q} &= \frac{\prod_{0 \leq i \leq k-2}(q^n - q^i)}{\prod_{0 \leq i \leq k-2}(q^{k-1} - q^i)} \cdot \frac{\prod_{0 \leq i \leq k-1}(q^k - q^i)}{\prod_{0 \leq i \leq k-1}(q^n - q^i)} = \frac{1}{q^n - q^{k-1}} \cdot \frac{\prod_{0 \leq i \leq k-1}(q^k - q^i)}{\prod_{0 \leq i \leq k-2}(q^{k-1} - q^i)} \\
&= \frac{1}{q^n - q^{k-1}} \cdot \frac{q^{k-1}(q^k - 1)\prod_{0 \leq i \leq k-2}(q^{k-1} - q^i)}{\prod_{0 \leq i \leq k-2}(q^{k-1} - q^i)} = \frac{q^k - 1}{q^{n-k+1} - 1}.
\end{aligned}$$

By using the previous theorem, the relation in (4.4) and observing that

$$\frac{d_{t-2,t}}{d_{t,t}} = \frac{d_{t-2,t}}{d_{t-1,t}} \frac{d_{t-1,t}}{d_{t,t}}$$

we immediately get the following corollary:

Corollary 4.2.5. *Let $q = 2$ and suppose $t < n$. We have the following relations:*

$$\begin{aligned}
d_{t,t} &= (2^n - 1)(2^n - 2) \cdots (2^n - 2^{t-1}); \\
\frac{d_{t-1,t}}{d_{t,t}} &= \frac{(2^t - 1)}{(2^n - 2^{t-1})} < \frac{1}{2^{n-t-1}} \leq 1; \\
\frac{d_{t-2,t}}{d_{t,t}} &= \frac{(2^t - 1)(2^{t-1} - 1)}{3(2^n - 2^{t-2})(2^n - 2^{t-1})}.
\end{aligned}$$

Corollary 4.2.6. *Let $q = 2$ and suppose $t = n$. We have the following relations:*

$$\begin{aligned} d_{n,n} &= (2^n-1)(2^n-2)\cdots(2^n-2^{n-1}); \\ \frac{d_{n-1,n}}{d_{n,n}} &= \frac{2^n-1}{2^{n-1}} \approx 2 > 1; \\ \frac{d_{n-2,n}}{d_{n,n}} &= \frac{(2^n-1)(2^{n-1}-1)}{9\cdot 2^{2n-3}}. \end{aligned}$$

In other words, the probability that a $(t \times n)$ random matrix $(t < n)$ with entries in $\mathbb{F}_2$ has rank exactly t is significantly greater than the probability of having rank equal to $t-1$ or $t-2$ or less. Instead, the probability that a square $(n \times n)$ random matrix has rank $n-1$ is the greatest.

Remark 4.2.7. In theory, the previous theorem can not be applied to our case because our construction imposes specific constraints, for example on the row-weight. However, in practice our ratio $\frac{d_{t-1,t}}{d_{t,t}}$ approaches that of the Corollary 4.2.6 for $t = \dim_{\mathbb{F}_2}(\langle \mathrm{Im}(\varepsilon)\rangle)$.

4.2.1 Application to AES

Because of the AES structure, we assign the following values to the parameters we have previously introduced. Let $V = (\mathbb{F}_2)^r$ be our starting vector space with $r = 128$ and $W = (\mathbb{F}_2)^s$, $s > 128$. We need to establish s. We consider the quotient $\mathbb{F}_{256} \cong \mathbb{F}_2[x]/\langle \mathsf{m}\rangle$, where $\mathsf{m} = x^8 + x^4 + x^3 + x + 1 \in \mathbb{F}_2[x]$ is the AES-polynomial. So $m = 8$. According to the previous section, we consider $\varepsilon' : \mathbb{F}_{2^8} \to (\mathbb{F}_2)^{256}$ by means of a primitive element γ of $\mathbb{F}_{256}$, which is a root of the primitive polynomial[3] $\mathsf{n} = x^8 + x^4 + x^3 + x^2 + 1 \in \mathbb{F}_2[x]$, and we define our parallel map $\varepsilon : V \to W$, with $r = mb = 128$ and $s = 2^m b = 4096$, as

$$\varepsilon(v_1, \dots, v_{16}) = (\varepsilon'(v_1), \dots, \varepsilon'(v_{16})).$$

We have that $\dim_{\mathbb{F}_2}\big(\langle \mathrm{Im}(\varepsilon)\rangle\big) = 4081$, by Proposition 4.5.3.
A tipical round function of the AES cryptosystem consists of the composition of two parallel maps (AddRoundKey and SubBytes [Section 2.1.1] operations) and two non-parallel maps (ShiftRows and MixColumns operations [Section 2.1.2]). We view the SubBytes (and AddRoundKey) operation as a parallel map π

$$\begin{aligned} \pi : (\mathbb{F}_{2^8})^{16} &\to (\mathbb{F}_{2^8})^{16} \\ (y_1, \cdots, y_{16}) &\mapsto (\pi_1(y_1), \cdots, \pi_{16}(y_{16})) \end{aligned}$$

where $y_i \in \mathbb{F}_{2^8}$ and $\pi_i \in \mathrm{Sym}(\mathbb{F}_{256})$, for $1 \le i \le 16$. In the SubBytes case, each component π_i, where $1 \le i \le 16$, is composition of inversion operation and an affine

[3]note that $\mathsf{n} \neq \mathsf{m}$; we could not use m because it is not primitive.

map; in the AddRoundKey case, we have a sum with the round-key. By the Theorem 1.1.8 we recalled in the first chapter, we have that $\mathrm{Sym}(\mathbb{F}_{256}) = \langle ax+b, x^{254}\rangle$, where $a, b \in \mathbb{F}_{256}$. We note that a parallel map can be linearized using elementary results from Representation Theory.

Moreover, we claim that ShiftRows is linear over $(\mathbb{F}_2)^{4096}$ and that MixColumns is not linear over $(\mathbb{F}_2)^{4096}$, as follows.
First of all, we recall the map that describes the ShiftRows operation:

$$\begin{array}{lll} \mathsf{sh}: & (\mathbb{F}_{2^8})^{16} & \to (\mathbb{F}_{2^8})^{16} \\ & (y_1, y_2, \cdots, y_{16}) & \mapsto (y_1, y_6, y_{11}, y_{16}, y_5, y_{10}, y_{15}, y_4, y_9, y_{14}, y_3, y_8, y_{13}, y_2, y_7, y_{12}). \end{array}$$

Denoting by $\mathbf{y} = (y_1, \cdots, y_{16})$, we note that

$$\varepsilon(\mathbf{y}) = (\varepsilon'(y_1), \varepsilon'(y_2), \varepsilon'(y_3), \varepsilon'(y_4), \varepsilon'(y_5), \cdots, \varepsilon'(y_{16}))$$

and

$$\varepsilon(\mathsf{sh}(\mathbf{y})) = (\varepsilon'(y_1), \varepsilon'(y_6), \varepsilon'(y_{11}), \varepsilon'(y_{16}), \varepsilon'(y_5), \cdots, \varepsilon'(y_{12})).$$

The map sh is linearly extendible because $\sum_{i\in I} \varepsilon(b^i) = 0$ clearly implies the following equality $\sum_{i\in I} \varepsilon(\mathsf{sh}(b^i)) = 0$. According to Lemma 4.1.6, it is possible to construct the linear map

$$A_{\mathrm{sh}} : (\mathbb{F}_2)^{4096} \to (\mathbb{F}_2)^{4096}$$

and so the ShiftRows operation is linear over $(\mathbb{F}_2)^{4096}$.

Now, we show that the MixColumns operation is not linear over $(\mathbb{F}_2)^{4096}$ using the following counterexample.

Example 4.2.8. Let $w_1, w_2, w_3, w_4 \in W$ such that $w_1 + w_2 + w_3 = w_4$:

$$\begin{array}{rcl} w_1 & = & (\varepsilon'(\gamma^1), \varepsilon'(\gamma^1), \varepsilon'(0), \varepsilon'(0), \varepsilon'(0), \cdots, \varepsilon'(0)) \\ w_2 & = & (\varepsilon'(\gamma^1), \varepsilon'(0), \varepsilon'(\gamma^1), \varepsilon'(0), \varepsilon'(0), \cdots, \varepsilon'(0)) \\ w_3 & = & (\varepsilon'(0), \varepsilon'(0), \varepsilon'(\gamma^1), \varepsilon'(0), \varepsilon'(0), \cdots, \varepsilon'(0)) \\ w_4 & = & (\varepsilon'(0), \varepsilon'(\gamma^1), \varepsilon'(0), \varepsilon'(0), \varepsilon'(0), \cdots, \varepsilon'(0)). \end{array}$$

Now, we apply the MixColumns operation MC to each vector w_1, w_2, w_3, w_4 obtaining the following

$$\begin{array}{rcl} \mathsf{MC}'(w_1) & = & (\varepsilon'(\gamma^1), \varepsilon'(\gamma^3), \varepsilon'(0), \varepsilon'(\gamma^{51}), \varepsilon'(0), \cdots, \varepsilon'(0)) \\ \mathsf{MC}'(w_2) & = & (\varepsilon'(\gamma^3), \varepsilon'(\gamma^{51}), \varepsilon'(\gamma^3), \varepsilon'(\gamma^{51}), \varepsilon'(0), \cdots, \varepsilon'(0)) \\ \mathsf{MC}'(w_3) & = & (\varepsilon'(\gamma^1), \varepsilon'(\gamma^3), \varepsilon'(\gamma^{51}), \varepsilon'(\gamma^1), \varepsilon'(0), \cdots, \varepsilon'(0)) \\ \mathsf{MC}'(w_4) & = & (\varepsilon'(\gamma^3), \varepsilon'(\gamma^{51}), \varepsilon'(\gamma^1), \varepsilon'(\gamma^1), \varepsilon'(0), \cdots, \varepsilon'(0)) \end{array}$$

where

$$\begin{tikzcd} V \arrow[r, "\varepsilon"] \arrow[d, "\mathsf{MC}"'] \arrow[dr, phantom, "\circlearrowright"] & W \arrow[d, "\mathsf{MC}'"] \\ V \arrow[r, "\varepsilon"] & W \end{tikzcd} .$$

Then we have that

$$\mathsf{MC}'(w_1)+\mathsf{MC}'(w_2)+\mathsf{MC}'(w_3) = (\varepsilon'(\gamma^3), \varepsilon'(\gamma^{51}), \varepsilon'(0)+\varepsilon'(\gamma^3)+\varepsilon'(\gamma^{51}), \varepsilon'(\gamma^1), \varepsilon'(0), \cdots, \varepsilon'(0)).$$

The third component of the previous vector is a sum in $(\mathbb{F}_2)^{256}$ and it has weight equal to 3. So, the vector $\mathsf{MC}'(w_1) + \mathsf{MC}'(w_2) + \mathsf{MC}'(w_3)$ is an element of the admissible space but it is a non-admissible vector. Therefore, $\mathsf{MC}'(w_4) = \mathsf{MC}'(w_1 + w_2 + w_3) \neq \mathsf{MC}'(w_1) + \mathsf{MC}'(w_2) + \mathsf{MC}'(w_3)$ and so the MixColumns is not linear over W. It means that the extension of MC is not linearly extendible.

Remark 4.2.9. If all the AES operations were parallel maps, it would be possible to linearize the "full" cryptosystem because the set of the parallel maps is a group with respect to the composition operation.

4.2.2 Application to PRESENT

As for AES, we assign the right values to our parameters, according to PRESENT's structure. Let $V = (\mathbb{F}_2)^r$ be our starting vector space with $r = 64$, and $W = (\mathbb{F}_2)^s$ with $s > 64$. We consider $\varepsilon' : \mathbb{F}_{2^4} \to (\mathbb{F}_2)^{16}$ and we define our parallel map $\varepsilon : V \to W$, with $r = mb = 64$ and $s = 2^m b = 256$, as

$$\varepsilon(v_1, \ldots, v_{16}) = (\varepsilon'(v_1), \ldots, \varepsilon'(v_{16})).$$

We note that $\dim_{\mathbb{F}_2}\left(\langle \mathrm{Im}(\varepsilon)\rangle\right) = 241$ (see Proposition 4.5.3).
A typical round function of the PRESENT cryptosystem consists of the composition of two parallel maps (addRoundKey and sBoxLayer operations [Section 2.3.1]) and one non-parallel map (pLayer operation [Section 2.3.2]). The addRoundKey (and sBoxLayer) operation is a parallel maps π

$$\begin{array}{rcl} \pi : (\mathbb{F}_{2^4})^{16} & \to & (\mathbb{F}_{2^4})^{16} \\ (t_1, \cdots, t_{16}) & \mapsto & (\pi_1(t_1), \cdots, \pi_{16}(t_{16})) \end{array}$$

where $\pi_i \in \mathrm{Sym}(\mathbb{F}_{16})$. In the sBoxLayer case, each component π_i, where $1 \leq i \leq 16$, is given by the table in Section 2.3.1; when π is the addRoundKey operation, we have only a bitwise sum with the round-key.
Moreover, it is easy to see that pLayer is not linear over $(\mathbb{F}_2)^{256}$.

Example 4.2.10. Let $w_1, w_2, w_3, w_4 \in W$ such that $w_1 + w_2 + w_3 = w_4$ and let $\zeta, \eta, \vartheta, \xi, \mu$ be distinct non-zero elements in $\mathbb{F}_{2^4}$. Suppose that

$$\begin{array}{rcl} w_1 & = & (\varepsilon'(\zeta), \varepsilon'(\zeta), \varepsilon'(0), \varepsilon'(0), \varepsilon'(0), \cdots, \varepsilon'(0)) \\ w_2 & = & (\varepsilon'(\zeta), \varepsilon'(0), \varepsilon'(\zeta), \varepsilon'(0), \varepsilon'(0), \cdots, \varepsilon'(0)) \\ w_3 & = & (\varepsilon'(0), \varepsilon'(0), \varepsilon'(\zeta), \varepsilon'(0), \varepsilon'(0), \cdots, \varepsilon'(0)) \\ w_4 & = & (\varepsilon'(0), \varepsilon'(\zeta), \varepsilon'(0), \varepsilon'(0), \varepsilon'(0), \cdots, \varepsilon'(0)). \end{array}$$

Now, we apply the pLayer transformation pL to each vector w_1, w_2, w_3, w_4 obtaining the following

$$\begin{array}{rcl} \mathsf{pL}'(w_1) & = & (\varepsilon'(\eta), \varepsilon'(0), \varepsilon'(0), \varepsilon'(0), \varepsilon'(\eta), \varepsilon'(0), \varepsilon'(0), \varepsilon'(0), \varepsilon'(\eta), \varepsilon'(0), \varepsilon'(0), \varepsilon'(0), \varepsilon'(\eta), \varepsilon'(0), \varepsilon'(0), \varepsilon'(0)) \\ \mathsf{pL}'(w_2) & = & (\varepsilon'(\vartheta), \varepsilon'(0), \varepsilon'(0), \varepsilon'(0), \varepsilon'(\vartheta), \varepsilon'(0), \varepsilon'(0), \varepsilon'(0), \varepsilon'(\vartheta), \varepsilon'(0), \varepsilon'(0), \varepsilon'(0), \varepsilon'(\vartheta), \varepsilon'(0), \varepsilon'(0), \varepsilon'(0)) \\ \mathsf{pL}'(w_3) & = & (\varepsilon'(\xi), \varepsilon'(0), \varepsilon'(0), \varepsilon'(0), \varepsilon'(\xi), \varepsilon'(0), \varepsilon'(0), \varepsilon'(0), \varepsilon'(\xi), \varepsilon'(0), \varepsilon'(0), \varepsilon'(0), \varepsilon'(\xi), \varepsilon'(0), \varepsilon'(0), \varepsilon'(0)) \\ \mathsf{pL}'(w_4) & = & (\varepsilon'(\mu), \varepsilon'(0), \varepsilon'(0), \varepsilon'(0), \varepsilon'(\mu), \varepsilon'(0), \varepsilon'(0), \varepsilon'(0), \varepsilon'(\mu), \varepsilon'(0), \varepsilon'(0), \varepsilon'(0), \varepsilon'(\mu), \varepsilon'(0), \varepsilon'(0), \varepsilon'(0)) \end{array}$$

Then, we have that

$$\mathsf{pL}'(w_4) = \mathsf{pL}'(w_1 + w_2 + w_3) \neq \mathsf{pL}'(w_1) + \mathsf{pL}'(w_2) + \mathsf{pL}'(w_3) = (\varepsilon'(\eta) + \varepsilon'(\vartheta) + \varepsilon'(\xi), \ldots),$$

where the first component has weight 3, and so the pLayer is not a linear operation over W.

Remark 4.2.11. As in the AES case, if all the PRESENT's operations were parallel maps, it would be possible to linearize the "full" cryptosystem because the set of the parallel maps is a group with respect to the composition operation.

4.2.3 Application to SERPENT

Let $V = (\mathbb{F}_2)^r$ be our starting vector space with $r = 128$. In order to identify the value of $r \geq s$, where $W = (\mathbb{F}_2)^s$, we have to consider the map

$$\varepsilon' : (\mathbb{F}_{2^4}) \to (\mathbb{F}_2)^{2^4}.$$

We define our parallel map $\varepsilon : V \to W$ with $r = mb = 128$ and $s = 2^m b = 512$ as

$$\varepsilon(v_1, \ldots, v_{32}) = (\varepsilon'(v_1), \ldots, \varepsilon'(v_{32})).$$

Note that $\dim_{\mathbb{F}_2}(\langle \mathrm{Im}(\varepsilon) \rangle) = 2^m b - (b - 1) = 481$.
The components of a typical Round function are the parallel S-box, the affine transformation described in Section 2.2.2 and the translation with the round key. Obviously, key translation and S-box are parallel maps of type

$$\begin{array}{rcl} \pi : (\mathbb{F}_{2^4})^{32} & \to & (\mathbb{F}_{2^4})^{32} \\ (t_1, \ldots, t_{32}) & \mapsto & (\pi_1(t_1), \ldots, \pi_{32}(t_{32})) \end{array}$$

where $\pi_i \in \mathrm{Sym}(\mathbb{F}_{2^4})$.

Similarly to what was done for AES and PRESENT, we could provide a counterexample to show that the linear transformation of SERPENT is not linear over $(\mathbb{F}_2)^{512}$.

4.3 An “orbit” representation

Starting from the setting we described in the previous section, we consider our parallel map $\varepsilon : (\mathbb{F}_{2^m})^b \to ((\mathbb{F}_2)^{2^m})^b$ defined as $\varepsilon(v_1, \dots, v_b) = (\varepsilon'(v_1), \dots, \varepsilon'(v_b))$. Now, let $\mathbf{M}$ be a matrix in $\mathrm{GL}((\mathbb{F}_2)^{mb})$ and let t be its order, $\mathbf{M}^t = \mathrm{id}_V$. Let $V = (\mathbb{F}_2)^r$ be a vector space with dimension $r = mb$ and let $W = (\mathbb{F}_2)^s$ be the vector space with dimension $s = 2^m bt$. The space embedding $\alpha : V \to W$ we propose in this section is defined as follows

$$\alpha(v) = (\varepsilon(v), \varepsilon(\mathbf{M}v), \dots, \varepsilon(\mathbf{M}^{t-1}v)). \tag{4.5}$$

From now on, α denotes the map in (4.5). Thanks to Proposition 4.2.3, we can easily prove the following proposition:

Proposition 4.3.1. *Let $V = (\mathbb{F}_2)^r$ be a vector space with dimension $r = mb$ and let $W = (\mathbb{F}_2)^s$ be the vector space with dimension $s = 2^m bt$. Let α be as in (4.5). Then we have*

$$2^m b - (b-1) \le \dim_{\mathbb{F}_2}\big(\langle \mathrm{Im}(\alpha)\rangle\big) \le (2^m b - (b-1))t$$

Proof. By Proposition 4.2.3, $\dim_{\mathbb{F}_2}\big(\langle \mathrm{Im}(\varepsilon)\rangle\big) = 2^m b - (b-1)$. Since

$$\{(\varepsilon(v), \varepsilon(\mathbf{M}v), \dots, \varepsilon(\mathbf{M}^{t-1}v)) \mid v \in V\} \subset \{(\varepsilon(v_1), \dots, \varepsilon(v_t)) \mid v_1, \dots, v_t \in V\},$$

then

$$\dim_{\mathbb{F}_2}\big(\langle \mathrm{Im}(\alpha)\rangle\big) \le (2^m b - (b-1))t.$$

On the other hand, considering the projection of $\{(\varepsilon(v), \varepsilon(\mathbf{M}v), \dots, \varepsilon(\mathbf{M}^{t-1}v))\}$ on the first component (the first b bytes), the lower bound follows immediately, again considering Proposition 4.2.3. □

We can further improve Proposition 4.3.1 for byte-oriented Mixing Layer.

Proposition 4.3.2. *Let $V = (\mathbb{F}_2)^r$ be a vector space with dimension $r = mb$ and let $W = (\mathbb{F}_2)^s$ be the vector space with dimension $s = 2^m bt$. Let $\mathbf{M} \in \mathrm{GL}((\mathbb{F}_{2^m})^b)$. Let α be as in (4.5). Then we have*

$$\dim_{\mathbb{F}_2}\big(\langle \mathrm{Im}(\alpha)\big) \le 2^m bt - (bt-1) - mb(t-1)$$

Proof. Let $T = \langle \mathrm{Im}(\alpha)\rangle$. For any $w_1, w_2 \in W$, let $w_1 \cdot w_2$ denote their scalar product. It is sufficient to show that there exist $(bt-1)+mb(t-1)$ elements in $T^\perp$ that are linearly independent, where $T^\perp = \{w \in W \mid w \cdot \mathbf{t} = 0, \forall \mathbf{t} \in T\}$ is the orthogonal space of T (or the "dual" of T, in coding theory notation). In fact, this means

$$\dim T^\perp \geq (bt-1)+mb(t-1)$$

and since $\dim T = \dim W - \dim T^\perp$, our result could follows.

Consider the following matrix product with $\mathbf{M} = (a_{i,j})$

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & \cdots & a_{1b} \\ a_{21} & a_{22} & \cdots & \cdots & a_{2b} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{b1} & a_{b2} & \cdots & \cdots & a_{bb} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_b \end{pmatrix} = \begin{pmatrix} v'_1 \\ v'_2 \\ \vdots \\ v'_b \end{pmatrix}$$

Obviously, $v'_1 = \sum_{i=1}^b v_i a_{1i}$.
Let S' be a subspace of $(\mathbb{F}_2)^m$ such that $\dim(S') = m-1$. For any $1 \leq i \leq b$, let $S_i = \{\beta \in (\mathbb{F}_{2^m}) \mid \beta a_{1i} \in S'\}$. We note that S_i is a subspace and that

$$\Big\{\sum_{i=1}^b v_i a_{1i} \mid v_i \in S_i,\ 1 \leq i \leq b\Big\} = S'$$

and that $|S_i| = |S'| = 2^{m-1}$. There exists a bijection via orthogonality between the following two sets $\mathcal{S} = \{S < (\mathbb{F}_2)^m \mid \dim(S) = m-1\}$ and $\{S^\perp < (\mathbb{F}_2)^m \mid \dim(S^\perp) = 1\}$; their cardinality is obviously $2^m - 1$. We can choose a linear basis for $\mathcal{S} \cup \{0\}$, i.e. $\mathcal{S} \cup \{0\} = \langle \mathbf{e}_1^\perp, \ldots, \mathbf{e}_m^\perp\rangle$. Therefore, each row of $\mathbf{M}$ generates m linearly independent elements of $T^\perp$.
Two relations coming from two different rows are independent, since the matrix $\mathbf{M}$ has full rank, for a total of mb relations.

Now, we construct the elements of the orthogonal space that correspond to the relations induced by the rows of $\mathbf{M}$. We are considering the case $(v, \mathbf{M}v)$ and we observe that

$$\sum_{i=1}^b v_i a_{1i} = v'_1 = (\mathbf{M}v)_1 \tag{4.6}$$

where $v_i \in S_i$.
Since $\varepsilon'(S_i) \subset (\mathbb{F}_2)^{2^m}$,we consider $w_i = \sum_{\ell \in \varepsilon'(S_i)} \ell$ where $\mathrm{w}(w_i) = |\varepsilon'(S_i)| = 2^{m-1}$ and $w_i \in (\mathbb{F}_2)^{2^m}$. The element of $T^\perp$ coming from (4.6) and S is

$$(w_1, \ldots, w_b, w'_1, \ldots, 0, \ldots 0)$$

where $w_1' = \sum_{\ell \in \varepsilon'(S')} \ell$. Clearly, $m-1$ similar elements come from (4.6) and $\mathcal{S}$.
If we consider the relations given by the h-th row of $\mathbf{M}$, i.e. $\sum_{i=1}^{b} v_i a_{hi} = v_h'$, we obtain the following elements

$$(w_1, \dots, w_b, 0, \dots, w_h', \dots, 0, \dots 0).$$

At this point, we have constructed the mb elements of the orthogonal space corresponding to the previous relations.
Instead of considering $(v, \mathbf{M}v)$, since clearly $\mathbf{M}(\mathbf{M}^i v) = \mathbf{M}^{i+1} v$, we can apply the previous construction to each pair $(\mathbf{M}^i v, \mathbf{M}^{i+1} v)$, for $1 \leq i \leq t-2$, obtaining the corresponding elements

$$(\underbrace{0, \dots, 0}_{b(i-1)}, \underbrace{w_1, \dots, w_b}_{b}, \underbrace{0, \dots, w_h', \dots, 0}_{b}, \underbrace{0, \dots, 0}_{bt-(i+1)b}) \tag{4.7}$$

We have found exactly $mb(t-1)$ vectors in $T^\perp$. Since the pairs $(\mathbf{M}^i v, \mathbf{M}^{i+1} v)$ and $(\mathbf{M}^j v, \mathbf{M}^{j+1} v)$ with $i \neq j$ involve different bytes, the relations given by $(\mathbf{M}^i v, \mathbf{M}^{i+1} v)$ are independent from those given by $(\mathbf{M}^j v, \mathbf{M}^{j+1} v)$. Then we have $mb(t-1)$ independent relations (i.e. linearly independent elements of the orthogonal space).

Thanks to Proposition 4.2.3, we have exactly $(bt-1)$ further relations, corresponding to elements in $T^\perp$ of type

$$(\underbrace{0, \dots, 0}_{k-1}, \underbrace{1, \dots, 1}_{b}, \underbrace{1, \dots, 1}_{b}, \underbrace{0, \dots, 0}_{bt-(k+1)}) \tag{4.8}$$

with $1 \leq k \leq bt$.

The vectors (4.7) and (4.8) form clearly a linearly independent set. □

As we have done in previous section, we can construct the following matrix. Let $\mathcal{D}$ be a subset of the plaintext set $\mathcal{M}$ such that $|\mathcal{D}| = \dim_{\mathbb{F}_2}(\langle \mathrm{Im}(\alpha) \rangle)$. Let $a_i \in \mathcal{D}$, $1 \leq i \leq |\mathcal{D}|$. We construct the $(|\mathcal{D}| \times 2^m bt)$-matrix $\mathbf{D}$ such that the i-th row is the image of the map α applied to the plaintext $a_i \in \mathcal{D}$, for $i \in \{1, \cdots, |\mathcal{D}|\}$:

$$\mathbf{D} = \begin{pmatrix} \alpha(a^1) \\ \alpha(a^2) \\ \vdots \\ \alpha(a^{|\mathcal{D}|}) \end{pmatrix} = \begin{pmatrix} \varepsilon(a^1) & \varepsilon(\mathbf{M}a^1) & \cdots & \varepsilon(\mathbf{M}^{t-1}a^1) \\ \varepsilon(a^2) & \varepsilon(\mathbf{M}a^2) & \cdots & \varepsilon(\mathbf{M}^{t-1}a^2) \\ \vdots & \vdots & \vdots & \vdots \\ \varepsilon(a^{|\mathcal{D}|}) & \varepsilon(\mathbf{M}a^{|\mathcal{D}|}) & \cdots & \varepsilon(M^{t-1}a^{|\mathcal{D}|}) \end{pmatrix}.$$

We expect the rank of this matrix to have a behavior similar to that of matrix $\mathbf{H}$ (4.3), see Remark 4.2.7. Our experiments confirm this.

Let $\tilde{\mathcal{G}}$ be the set of parallel maps $\tilde{\pi} : (\mathbb{F}_{2^m})^b \to (\mathbb{F}_{2^m})^b$, such that, for any $1 \leq j \leq b$, $\tilde{\pi}_j(x) = ax + c$, with $a \neq 0, c \in \mathbb{F}_{2^m}$ (a and c do not depend on j).

Let $\bar{\mathcal{G}}$ be the set of parallel maps $\bar{\pi} : (\mathbb{F}_{2^m})^b \to (\mathbb{F}_{2^m})^b$, such that, for any $1 \le j \le b$, $\bar{\pi}_j(x) = x + d_j$, with $d_j \in \mathbb{F}_{2^m}$.
Note that both $\tilde{\mathcal{G}}$ and $\bar{\mathcal{G}}$ are subgroups of $\mathrm{Sym}((\mathbb{F}_{2^m})^b)$ and we define $\mathcal{G}$ as

$$\mathcal{G} = \left\langle \tilde{\mathcal{G}}, \bar{\mathcal{G}}, \mathbf{M} \right\rangle < \mathrm{Sym}((\mathbb{F}_{2^m})^b).$$

The following result holds:

Proposition 4.3.3. *Let σ be either an element of $\tilde{\mathcal{G}}$ or an element of $\bar{\mathcal{G}}$, then there exists $A_\sigma : W \to W$ which is linear.*

Proof. We want to apply Lemma 4.1.6 and so we must only show that σ is linearly extendible. Let $\{v^i\}_{i\in I} \subset V$ such that $\sum_{i\in I} \alpha(v^i) = 0$, we have to prove that $\sum_{i\in I} \alpha(\sigma(v^i)) = 0$. Note that $\sum_I \alpha(v^i) = 0$ is equivalent to

$$\sum_I \Big(\varepsilon'(v^i)_1, \dots, \varepsilon'(v^i)_b, \varepsilon'(\mathbf{M}v^i)_1, \dots, \varepsilon'(\mathbf{M}v^i)_b, \dots, \varepsilon'(\mathbf{M}^{t-1}v^i)_1, \dots, \varepsilon'(\mathbf{M}^{t-1}v^i)_b\Big) = 0.$$

Then we have the following system S_j for any $1 \le j \le b$

$$S_j = \begin{cases} \sum_I \varepsilon'(v^i_j) = 0 \\ \sum_I \varepsilon'((\mathbf{M}v^i)_j) = 0 \\ \vdots \\ \sum_I \varepsilon'((\mathbf{M}^{t-1}v^i)_j) = 0. \end{cases}$$

Using Lemma 4.2.2, we have that S_j is equivalent to S'_j

$$S'_j = \begin{cases} |\{\ell \mid v^\ell_j = v^i_j\}| \text{ is even } \forall i \in I \\ |\{\ell \mid (\mathbf{M}v^\ell)_j = (\mathbf{M}v^i)_j\}| \text{ is even } \forall i \in I \\ \vdots \\ |\{\ell \mid (\mathbf{M}^{t-1}v^\ell)_j = (\mathbf{M}^{t-1}v^i)_j\}| \text{ is even } \forall i \in I. \end{cases}$$

Suppose $\sigma \in \tilde{\mathcal{G}}$ which means that $\sigma(v) = \sigma(v_1, \cdots, v_b) = (\sigma_1(v_1), \cdots, \sigma_b(v_b))$ where $\sigma_i(v_i) = av_i + c$ for any $1 \le i \le b$ and $a \ne 0, c \in \mathbb{F}_{2^m}$.
Since $\mathbf{M}$ is linear, we have

$$\begin{aligned} (\mathbf{M}^h\sigma(v^\ell))_j &= (\mathbf{M}^h(av^\ell_1 + c, \cdots, av^\ell_b + c))_j \\ &= (a\mathbf{M}^h v^\ell + \mathbf{M}^h(c, \cdots, c))_j \\ &= (a\mathbf{M}^h v^\ell)_j + (\mathbf{M}^h(c, \cdots, c))_j \\ &= a(\mathbf{M}^h v^\ell)_j + \bar{c}, \end{aligned}$$

where $\bar{c}$ is a constant *independent* of ℓ.

We have that,$\forall i \in I$ and for any $1 \leq h \leq t-1$, $|\{\ell \mid (\mathbf{M}^h v^\ell)_j = (\mathbf{M}^h v^i)_j\}|$ is even and so that $|\{\ell \mid a(\mathbf{M}^h v^\ell)_j + \bar{c} = a(\mathbf{M}^h v^i)_j + \bar{c}\}|$ is even. Thanks to Lemma 4.2.2, our thesis follows.

Suppose now that $\sigma \in \bar{\mathcal{G}}$, i.e. $\sigma(v) = v + d$ for some $d \in V$.
Since

$$\begin{aligned}(\mathbf{M}^h \sigma(v^\ell))_j &= (\mathbf{M}^h(av^\ell + d))_j \\ &= (\mathbf{M}^h v^\ell)_j + (\mathbf{M}^h(d))_j \\ &= (\mathbf{M}^h v^\ell)_j + \bar{d},\end{aligned}$$

where $\bar{d}$ is a constant *independent* of ℓ and $|\{\ell \mid (\mathbf{M}^h v^\ell)_j = (\mathbf{M}^h v^i)_j\}|$ is even, we have that

$$|\{\ell \mid (\mathbf{M}^h v^\ell)_j + \bar{d} = (\mathbf{M}^h v^i)_j + \bar{d}\}|$$

is even. By Lemma 4.2.2, our thesis follows. □

4.3.1 Application to AES

Let $V = (\mathbb{F}_2)^r$ be a vector space with dimension $r = 128$ and let $\mathsf{M} : V \to V$ be the $\mathsf{MixingLayer}$ of AES, that is, the composition of $\mathsf{ShiftRows}$ and $\mathsf{MixColumns}$. Since M has order equal to 8 (i.e. $\mathsf{M}^8 = \mathrm{id}_V$), the map $\alpha : V \to W$ we propose is defined as follows

$$\alpha(v) = (\varepsilon(v), \varepsilon(\mathsf{M}v), \ldots, \varepsilon(\mathsf{M}^7 v)), \tag{4.9}$$

where $W = (\mathbb{F}_2)^s$ is the vector space with dimension $s = 2^m bt = 2^{15}$ and ε is the map defined in Subsection 4.2.1: $\varepsilon : (\mathbb{F}_2)^{128} \to (\mathbb{F}_2)^{4096}$.
Let $T = \langle \mathrm{Im}(\alpha) \rangle$ with α in (5.1). We can easily determine $\dim(T)$.

Fact 4.3.4. *In the AES case we have*

$$\dim_{\mathbb{F}_2}(T) = 2^m bt - (bt-1) - mb(t-1) = 31745.$$

Proof. Let $\lambda = 2^m bt - (bt-1) - mb(t-1)$. By computational experiments, we have found a $(\lambda \times 2^m bt)$ full rank matrix for the α representation in the AES case. Which means $\dim_{\mathbb{F}_2} T \geq \lambda$. Thanks to Proposition 4.3.2 we conclude that $\dim_{\mathbb{F}_2} T = \lambda$. □

We note that the group

$$\mathcal{G} = \left\langle \tilde{\mathcal{G}}, \bar{\mathcal{G}}, \mathsf{M} \right\rangle < \mathrm{Sym}((\mathbb{F}_{2^8})^{16}).$$

contains all the permutations of the AES-round function, except notably for the S-box operation.

Proposition 4.3.5. *Let* M *be the* MixingLayer. *Then* α *is a space embedding with respect to* $\mathcal{G} = \left\langle \tilde{\mathcal{G}}, \bar{\mathcal{G}}, \mathsf{M} \right\rangle$.

Proof. According to Proposition 4.3.3, there exists a linear map $A_\sigma : W \to W$ in case σ is $\tilde{\mathcal{G}}$ or $\bar{\mathcal{G}}$. We note that the previous result is independent from $\mathbf{M}$. Let $\mathbf{M}$ be the MixingLayer M. Since $\alpha(v^i) = (\varepsilon(v^i), \varepsilon(\mathsf{M}v^i), \dots, \varepsilon(\mathsf{M}^7 v^i))$ and

$$\begin{aligned} \alpha(\mathsf{M}v^i) &= (\varepsilon(\mathsf{M}v^i), \varepsilon(\mathsf{M}^2 v^i), \dots, \varepsilon(\mathsf{M}^8 v^i)) \\ &= (\varepsilon(\mathsf{M}v^i), \varepsilon(\mathsf{M}^2 v^i), \dots, \varepsilon(v^i)) \end{aligned}$$

$\alpha(\mathsf{M}v^i)$ is a permutation of $\alpha(v^i)$. Obviously, we have that $\sum_{i\in I} \alpha(v^i) = 0$ implies $\sum_{i \in I} \alpha(\mathsf{M}v^i) = 0$. □

With a fixed K, the encryption ϕ_K is the composition of AddRoundKey, Subbytes and MixingLayer. So the only part of ϕ_K which is not linear (with our map α) is the SubBytes operation.

4.3.2 Application to PRESENT

Let $V = (\mathbb{F}_2)^r$ be a vector space with dimension $r = 64$ and let $\mathsf{M} : V \to V$ be the pLayer of PRESENT. Since $\mathsf{M}^3 = \mathrm{id}_V$, the map $\alpha : V \to W$ we propose is defined as follows

$$\alpha(v) = (\varepsilon(v), \varepsilon(\mathsf{M}v), \varepsilon(\mathsf{M}^2 v)), \tag{4.10}$$

where $W = (\mathbb{F}_2)^s$ is the vector space with dimension $s = 2^m bt = 768$. Let α be as in (4.10) and $T = \langle \mathrm{Im}(\alpha) \rangle$. Also in this case it is possible to prove (with a computation) that $\dim_{\mathbb{F}_2}(T) = 2^m bt - (bt-1) - mb(t-1) = 593$

With a fixed K, the encryption ϕ_K is the composition of addRoundKey, sBoxLayer and pLayer. So the only part of ϕ_K which is not linear (with our map α) is the sBoxlayer operation.

4.3.3 Application to SERPENT

Let $V = (\mathbb{F}_2)^r$ be a vector space with dimension $r = 128$ and let $\mathsf{M} : V \to V$ be the affine transformation of SERPENT. Since the order of M is greater than 2^{80} it is huge and impractical to consider the map $\alpha : V \to W$

$$\alpha(v) = (\varepsilon(v), \varepsilon(\mathsf{M}v), \dots, \varepsilon(\mathsf{M}^{2^{80}} v), \dots). \tag{4.11}$$

since $W = (\mathbb{F}_2)^s$ would have $s = 2^m bt > 2^4\, 32\, 2^{80} = 2^{89}$, making the rank computation impossible with nowadays technology.

4.4 Other representations of this kind

We can also build other representations similar to those described in previous sections. The main objective in these constructions is to identify the right compromise between computational feasibility and quantity of information that can be obtained. In Section 4.2, we constructed the embedding ε that has been useful to make linear the S-box maps which are the classical non-linear maps of a cryptosystem. We had to abandon the linearity of $\mathsf{MixColumns}$ (for AES) and the pLayer (in case of PRESENT). In order to use some more information about the $\mathsf{MixColumns}$ (or the pLayer for PRESENT), we have considered the embedding given in Section 4.3:

$$\alpha(v) = (\varepsilon(v), \varepsilon(\mathbf{M}v), \ldots, \varepsilon(\mathbf{M}^{t-1}v)),$$

where $\mathbf{M}$ is the full Mixing Layer. The strength of this representation is that we can exploit the low order of $\mathbf{M}$ to force the linearity of $\mathbf{M}$. The disadvantages are that we have lost some computational efficiency and that the S-box is non-linear again (but with a lower non-linearity).

For AES, we considered also the embedding given by

$$\alpha(v) = (\varepsilon(v), \varepsilon(\mathsf{MC}(v)), \ldots, \varepsilon(\mathsf{MC}^3(v))),$$

since the order of the $\mathsf{MixColumns}$ is equal to 4 and the $\mathsf{MixColumns}$ operation was the only to be non-linear in Section 4.2. Unfortunately, in this context both the $\mathsf{ShiftRows}$ and the parallel maps are non-linear and so we put aside this idea.

Although the following two embeddings could provide a lot of information about a cryptosystem,

- $\alpha(v) = (\varepsilon(v), \varepsilon((\mathbf{M} \circ \mathrm{Sbox})v), \ldots, \varepsilon((\mathbf{M} \circ \mathrm{Sbox})^{t-1}v)) \quad (t = \mathsf{o}(\mathbf{M} \circ \mathrm{Sbox}))$
- $\alpha(v) = (\varepsilon(v), \varepsilon((\gamma\lambda\sigma_k)v), \ldots, \varepsilon((\gamma\lambda\sigma_k))^{t-1}v)) \quad (t = \mathsf{o}(\gamma\lambda\sigma_k))$

they are very impractical, since the order of $(\mathbf{M} \circ \mathrm{Sbox})$ and of $(\gamma\lambda\sigma_k)$ is huge.

4.5 Further remarks

Let $\mathcal{C}$ be any block cipher such that the plain-text space $\mathcal{M}$ coincides with the cipher space. Let $\mathcal{K}$ be the key space. Any key $k \in \mathcal{K}$ induces a permutation τ_k on $\mathcal{M}$. Since $\mathcal{M}$ is usually $V = (\mathbb{F}_2)^n$ for some $n \in \mathbb{N}$, we can consider $\tau_k \in \mathrm{Sym}(V)$. We denote by $\Gamma = \Gamma(\mathcal{C})$ the subgroup of $\mathrm{Sym}(V)$ generated by all the τ_k's. Unfortunately, the knowledge of $\Gamma(\mathcal{C})$ is out of reach for the most important block ciphers, such as the AES [Nat01] and the DES [Nat77]. However, researchers have been able to compute another related group. Suppose that $\mathcal{C}$ is the composition of l rounds (the division into rounds is provided in the document describing the cipher). Then any key k would induce l permutations, $\tau_{k,1}, \ldots, \tau_{k,l}$, whose composition is τ_k. For any round h, we can consider $\Gamma_h(\mathcal{C})$ as the subgroup of $\mathrm{Sym}(V)$ generated by the $\tau_{k,h}$'s (with k varying in $\mathcal{K}$). We can thus define the group $\Gamma_\infty = \Gamma_\infty(\mathcal{C})$ as the subgroup of $\mathrm{Sym}(V)$ generated by all the Γ_h's. Obviously, $\Gamma \leq \Gamma_\infty$. Group Γ_∞ is traditionally called the *group generated by the round functions* with independent sub-keys. This group is known for some important ciphers, for example we have

Proposition 4.5.1 ([SW08],[Wer02])**.**

$$\Gamma_\infty(\mathrm{AES}) = \mathrm{Alt}((\mathbb{F}_2)^{128}).$$

It is common belief among researchers that $\Gamma_{AES} = \Gamma_\infty(\mathrm{AES}) = \mathrm{Alt}((\mathbb{F}_2)^{128})$. Assuming this, we show in this section that it is impossible to view Γ_{AES} as a subgroup of $\mathrm{GL}(V)$ with V of small dimension. In Cryptography it is customary to present estimates as powers of two, so our problem becomes to find the smallest ℓ such that Γ_{AES} can be linearized in $\mathrm{GL}((\mathbb{F}_2)^{2^\ell})$.
Using the Classification Theorem of finite simple groups (CFSG, see e.g.[Cam99]), it is possible to show that $\ell = 128$. In fact, the regular representation gives a bound $\ell \leq 128$ and for $n > 8$ the minimal dimension of any nontrivial representation of $\mathrm{Alt}(V)$ (over any field) can be proved to be at least $(n-2)$ ([KL90]). However, CFSG is a very deep and involved result, which arises some doubts in the research community. As Cameron says in [Cam99]

$<$ It is quite impossible for a layman to judge whether a complete proof of the theorem currently exists..... A result which has been proved using CFSG, but which has defined all attempts at an "elementary" proof, probably lies deep.$>$

We feel desirable to obtain a result[4] with an "elementary" proof. A classical proof is given in [Wag76]. There are two obvious ways to show that a finite group A cannot be contained (as isomorphic image) in a finite group B. The first is to show that $|A| > |B|$, the second is to show that there is $\eta \in A$ such that its order is strictly larger than the maximum element order in B. Subsection 4.5.1 presents our result using the first approach and we show that $\ell \geq 67$, which is more than enough to ensure the infeasibility of the linearization attack. This subsection's argument is completely elementary. Subsection 4.5.2 present our result using the second approach and we show again that $\ell \geq 67$. It is interesting that, although here some more advanced argument is needed (results in number theory), we reach the same estimate.

[4]this estimate is weaker, but strong enough to show the linearization infeasibility.

4.5.1 First approach

In this subsection we show that the order of $\mathrm{Alt}((\mathbb{F}_2)^{128})$ is strictly larger than the order of $\mathrm{GL}(V)$, with $V = (\mathbb{F}_2)^{2^{66}}$, so that $\ell \geq 67$.

We begin with showing a lemma.

Lemma 4.5.2. *The following inequality holds*

$$2^{(2^7)^{19}} < 2^{128}! < 2^{(2^7)^{20}}.$$

Proof. Let $n = 2^7$, we have to show $2^{n^{19}} < 2^n! < 2^{n^{20}}$. We first show that $2^{n^{19}} < 2^n!$. The following inequality holds for $1 \leq i \leq n-2$ and $1 \leq h \leq 2^{n-i}$

$$\frac{1}{2^{n-i}} \geq \frac{1}{2^{n-i+1}-h} \,. \tag{4.12}$$

Clearly

$$\begin{aligned} 2^n! > 2^{n^{19}} \quad &\Longleftrightarrow \quad 2^n(2^n-1)! > 2^n \cdot 2^{n^{19}-n} \\ &\Longleftrightarrow \quad (2^n-1)(2^n-2)! > 2^{n^{19}-n} \cdot \frac{2^n-1}{2^n-1} \quad . \end{aligned}$$

We apply (4.12) with $i = 1$ and $h = 1$ and so we must prove

$$(2^n-1)(2^n-2)! > 2^{n^{19}-n} \cdot \frac{2^n-1}{2^{n-1}},$$

i.e. $(2^n-2)! > 2^{n^{19}-n-(n-1)}$. We use the same inequality for all $2 \leq h \leq 2^{n-1}$ and we obtain that we must verify $(2^{n-1}-1)! > 2^{n^{19}-n-2^{n-1}(n-1)}$. Then we proceed by applying (4.12) for all $2 \leq i \leq n-2$ and all $1 \leq h \leq 2^{n-i}$, so that we need only to prove

$$(2^{n-(n-1)}-1)! \geq 2^{n^{19}-n-\sum_{i=1}^{n-1} 2^{n-i}(n-i)} \,.$$

In other words, we have to prove

$$1 > 2^{n^{19}-n-\sum_{i=1}^{n-1} 2^{n-i}(n-i)}, \quad \text{that is,} \quad 0 > n^{19} - n - \sum_{i=1}^{n-1} 2^{n-i}(n-i). \tag{4.13}$$

But a direct check shows that the right-hand size of (4.13) holds when $n = 2^7$.

We are left to demonstrate the following inequality: $2^n! < 2^{n^{20}}$.
We proceed by induction for $2 \leq n \leq 2^7$. In this range a computer computation shows that

$$n^{20} + 2^n n + 2^n < (n+1)^{20}. \tag{4.14}$$

When $n = 2$, we have $2^2! < 2^{2^{20}}$. Suppose that $2^n! < 2^{n^{20}}$ and $n \leq 2^7$. We have to prove that $2^{(n+1)}! < 2^{(n+1)^{20}}$. Since $2^{n+1}! = (2^n \cdot 2)! = 2^n!(2^n+1)\cdots(2^n+2^n)$, we have

$$2^n!(2^n+1)\cdots(2^n+2^n) < 2^{n^{20}+n+1}\cdot(2^n+2)\cdots(2^n+2^n) \leq 2^{n^{20}+2^n(n+1)} = 2^{n^{20}+2^n n+2^n}$$

and, applying (4.14), we get $2^{n^{20}+2^n n+2^n} < 2^{(n+1)^{20}}$.
Then the claimed inequality $2^{n+1}! < 2^{(n+1)^{20}}$ follows. □

Our result is contained in the following proposition.

Proposition 4.5.3. *Let $V = (\mathbb{F}_2)^{2^\ell}$ with $\ell \geq 2$. If $G < \mathrm{GL}(V)$, with G isomorphic to $\mathrm{Alt}((\mathbb{F}_2)^{128})$, then $\ell \geq 67$.*

Proof. If $G < \mathrm{GL}(V)$, then $|G| \leq |\mathrm{GL}(V)|$. But $|\mathrm{Sym}((\mathbb{F}_2)^{128})| = 2^{128}! > 2^{2^{133}}$ thanks to Lemma 4.5.2 and so

$$|G| = |\mathrm{Alt}((\mathbb{F}_2)^{128})| = \frac{|\mathrm{Sym}((\mathbb{F}_2)^{128})|}{2} > \frac{2^{2^{133}}}{2} = 2^{2^{133}-1} > 2^{2^{132}} > |\mathrm{GL}((\mathbb{F}_2)^{2^{66}})|.$$

Therefore, $\ell = 66$ is not large enough. □

An immediate consequence of the previous proposition is that the AES cannot be linearized unless using matrices of size at least 2^{67}, which is obviously impractical.

Remark 4.5.4. The previous proposition could be used also for any other block cipher $\mathcal{C}$ acting on 128−bit messages such that $\Gamma_\infty(\mathcal{C}) = \mathrm{Alt}((\mathbb{F}_2)^{128})$, e.g. the SERPENT ([BAK98]).

4.5.2 Using the order of the elements

In this subsection we compare the maximum order of elements in the two groups $\mathrm{Alt}((\mathbb{F}_2)^{128})$ and $\mathrm{GL}((\mathbb{F}_2)^{2^\ell})$. We use permutations of even order. We denote by $\mathsf{o}(\sigma)$ the order of any permutation σ, with $\sigma \in \mathrm{Alt}((\mathbb{F}_2)^{128})$ or $\sigma \in \mathrm{GL}((\mathbb{F}_2)^{2^\ell})$.

The best available result for $\mathrm{GL}((\mathbb{F}_2)^{2^\ell})$ is given by the following theorem

Theorem 4.5.5 ([Dar08])**.** *Let $\sigma \in GL((\mathbb{F}_2)^N)$, with $\mathsf{o}(\sigma)$ is even and $N \geq 4$. Then*

$$\mathsf{o}(\sigma) \leq 2(2^{N-2}-1) = 2^{N-1}-2.$$

Moreover, there is $\sigma \in \mathrm{GL}((\mathbb{F}_2)^N)$ whose order attains the upper bound.

Proof. It comes directly from Theorem 1 in [Dar08], with $p = q = 2$ and $N \geq 4$ (so point (a) and (b) do not apply). □

As regards the order of the elements in $\mathrm{Alt}((\mathbb{F}_2)^{128})$, we would like to use the following theorem

Theorem 4.5.6 ([DM96]). *If $n \geq 7$, then $\mathrm{Alt}((\mathbb{F}_2)^n)$ contains an element η of order (strictly) greater then $e^{\sqrt{(1/4)n\ln n}}$.*

The previous theorem is the special case of Theorem 5.1.A (p.145 in [DM96]) when $q = 2$.
In order to be able to compare the two estimates coming from Theorem 4.5.5 and Theorem 4.5.6, we rewrite Theorem 4.5.6 as follows, in order to have $\mathsf{o}(\eta)$ even. Our proof is an easy adaption of the proof contained in [DM96].

Theorem 4.5.7. *If $n \geq 28$, then $\mathrm{Alt}((\mathbb{F}_2)^n)$ contains an element η with $\mathsf{o}(\eta) > e^{\sqrt{(1/4)n\ln n}}$ and $\mathsf{o}(\eta)$ even.*

Proof. Suppose that $p_1, \dots, p_r$ are distinct odd primes elements such that $2 + p_1 + \cdots + p_r \leq n$. Then $\mathrm{Alt}((\mathbb{F}_2)^n)$ contains an element whose non trivial cycles have length $2, p_1, \dots, p_r$ and whose order is therefore $2p_1 \cdots p_r$.
We are going to show that there is $z \in \mathbb{R}$ such that

$$2 + \sum_{2<p\leq z} p \leq n \qquad \text{and} \qquad (\vartheta(z))^2 > \frac{1}{4} n \ln(n)$$

where $\vartheta(z) = \ln(2) + \sum_{2<p\leq z} \ln(p) = \ln(2) + \vartheta^*(z)$.

Let $f(z) = \frac{z}{\ln(z)}$. Since $f(z)$ is an increasing function for $z > e$, we have

$$\begin{aligned} 2 + \sum_{2<p\leq z} p &= f(2)\ln(2) + \sum_{2<p\leq z} f(p)\ln(p) \\ &= f(2)\ln(2) + f(3)\ln(3) + \sum_{3<p\leq z} f(p)\ln(p) \\ &\leq f(z)\ln(3) + \sum_{3<p\leq z} f(z)\ln(p) \\ &= \sum_{2<p\leq z} f(z)\ln(p) = f(z)\sum_{2<p\leq z}\ln(p) = f(z)\vartheta^*(z). \end{aligned}$$

where we have used that $f(2)\ln(2) + f(3)\ln(3) < f(z)\ln(3)$ if $n \geq 28$ (note that $\vartheta(11) = 28$). We take z such that $f(z)\vartheta^*(z) = n$. Since $\vartheta^*(z) > z/2$ for all $z \geq 11$, we have

$$n = \frac{z\vartheta^*(z)}{\ln(z)} < \frac{2(\vartheta^*(z))^2}{\ln(2\vartheta^*(z))} = f(4(\vartheta^*(z))^2).$$

However we also have $f(n\ln(n)) < n$. Since f is an increasing function, this shows that $n\ln(n) < 4(\vartheta^*(z))^2 < 4(\vartheta(z))^2$. □

Now, we compare the estimate from Theorem 4.5.5 and Theorem 4.5.6.
Take $n = 2^{128}$ and $\eta \in \mathrm{Alt}((\mathbb{F}_2)^{128})$ such that $\mathsf{o}(\eta) \geq e^t$ ($\mathsf{o}(\eta)$ even), where $t = \sqrt{(1/4)n \ln n} = \sqrt{(1/4)2^{128}\ln(2^{128})}$.
Since

$$e^t = e^{\sqrt{2^{126}128\ln 2}} = e^{\sqrt{2^{133}\ln 2}} = (e^{\sqrt{2\ln 2}})^{2^{66}},$$

by replacing e with $2^{\log_2 e}$, we obtain

$$e^t = (2^{(\log_2 e)\sqrt{2\ln 2}})^{2^{66}} = 2^{2^{66}(\log_2 e)\sqrt{2\ln 2}} = 2^{2^{66}\varepsilon},$$

where $\varepsilon \in \mathbb{R}$ is circa 1.69. According to Theorem 4.5.7, the order of η is at least $\mathsf{o}(\eta) \geq e^{2^{66}\varepsilon}$. If $\mathrm{Alt}((\mathbb{F}_2)^{128}) \subset \mathrm{GL}((\mathbb{F}_2)^N)$, we then need the the smallest N such that $\mathsf{o}(\eta) \leq (2^{N-1} - 2)$ (Theorem 4.5.5). In other words we have to see when the following inequality holds

$$\mathsf{o}(\eta) = 2^{2^{66}\varepsilon} \leq 2^{N-1} - 2. \tag{4.15}$$

We observe that

- if $N = 2^{66}$, then (4.15) is false, since $2^{2^{66}\varepsilon} > 2^{2^{66}} > 2^{2^{66}-1} - 2$;
- if $N = 2^{67}$, then (4.15) is true, since $2^{2^{66}\varepsilon} < 2^{2^{66}(1.7)} < 2^{2^{67}-1} - 2$.

Therefore, we need at least $\ell \geq 67$ in order to embed $\mathrm{Alt}((\mathbb{F}_2)^{128}) \subset \mathrm{GL}(V)$, which is exactly the same value as in Proposition 4.5.3.

Remark 4.5.8. It is shown in Landau [Lan03] that the maximum order of an element in $\mathrm{Sym}((\mathbb{F}_2)^n)$ is asymptotic to $e^{\sqrt{n\ln n}}$ as $n \to \infty$. Assuming this, we observe that we could slightly improve the value of k we need to $k \geq 68$.

4.6 Some results on a weaker notion of linearity

The results in this section are jointly with L. Maines and the proofs are contained in her Master's thesis [Mai09], supervised by M. Sala.
The main goal sought in Section 4.1, Section 4.2, Section 4.3, and Section 4.4 is to find practical embedding of $(\mathbb{F}_2)^{128}$ into a larger space where all components of the round function become linear. This is impossible, as shown in Section 4.5, but what we achieve in Section 4.3 is an embedding where the non-linear maps are "not so far" from linear maps. There are many notions of "non-linearity", but none of them can be easily computed in our setting. When we say "not so far from linear", we mean that these functions behave with matrix ranks in a way similar to that of linear maps, as discussed in Chapter 5.

However, we have been able to introduce a new non -linearity notion, that we call *s-extendibility* (Definition 4.6.1). We are not able to apply it in the embedding

$$\alpha : v \to (\varepsilon(v), \varepsilon(\mathbf{M}v), \cdots, \varepsilon(\mathbf{M}^7 v)). \tag{4.16}$$

but we can apply it[5] to

$$\alpha : v \to (\varepsilon(v), \varepsilon(\mathbf{M}v)).$$

and so our definition and our results (the main results of this section is Theorem 4.6.6) should be seen as a step forward the complete understanding of the surviving non-linearity in (4.16).

Definition 4.6.1. *Let $V = (\mathbb{F}_2)^r$ and $W = (\mathbb{F}_2)^s$, with $s > r$. Let $\sigma \in \mathrm{Sym}(V)$ and α be an injective map $\alpha : V \to W$. We say that σ is s-**extendible** (via α) if $\forall \{v^h\}_{1 \le h \le s} \subset V$ we have*

$$\sum_{h=1}^{s} \alpha(v^h) = 0 \iff \sum_{h=1}^{s} \alpha(\sigma(v^h)) = 0.$$

Remark 4.6.2. If $v^1 = v^2$ and $v^3 = v^4$, then $\forall \alpha$ and $\forall \sigma$ we have

$$\alpha(v^i) + \alpha(v^2) + \alpha(v^3) + \alpha(v^4) = 0$$

and

$$\alpha(\sigma(v^i)) + \alpha(\sigma(v^2)) + \alpha(\sigma(v^3)) + \alpha(\sigma(v^4)) = 0.$$

So if we test the 4-extendibility of σ only on these sets of vectors, we will find that any σ is 4-extendible. We call these vectors "coupled vectors".

We note that if σ is s-extendible $\forall s \in \mathbb{N}$, then σ is linearly extendible, according to Definition 4.1.4. Moreover, any linear map is s-extendible for all s. A random map is a 2-extendible but (with high probability) it is not s-extendible for any $s \ge 4$. Therefore, any 4-extendible map can be considered closer to a linear map. We would like to have results on our embedding concerning the s-extendibility of maps. A first result in this direction is obtained using the space embedding

$$\alpha(v) = (\varepsilon(v), \varepsilon(\mathbf{M}v)), \tag{4.17}$$

where $\mathbf{M}$ is a $(n \times n)$-matrix with entries in $\mathbb{F}_{2^m}$, as we are going to explain.

[5]under specific conditions on M

Definition 4.6.3. *Let $i, j, x, y, \alpha, \beta, \dots \in \mathbb{F}_{2^m}$ and $\mathbf{M}$ an $(n \times n)$-matrix with entries in $\mathbb{F}_{2^m}$*

$$\mathbf{M} = \begin{pmatrix} m_{11} & m_{12} & \dots & m_{1n} \\ m_{21} & m_{22} & \dots & \dots \\ \vdots & \vdots & \ddots & \ddots \\ m_{n1} & \dots & \dots & m_{nn} \end{pmatrix}.$$

The vectors $w_1, w_2, w_3, w_4 \in (\mathbb{F}_{2^m})^{2n}$ are 4-related vectors *if they can be permuted in order to have the following form:*

		1	2	$\dots$	$n+1$	$\dots$	$2n$
1	$w_1,$	$(i,$	$x,$	$\dots$	$m_{11}i + m_{12}x + \dots,$	$\dots$	$m_{n1}i + m_{n2}x + \dots)$
2	$w_2,$	$(i,$	$y,$	$\dots$	$m_{11}i + m_{12}y + \dots,$	$\dots$	$m_{n1}i + m_{n2}y + \dots)$
3	$w_3,$	$(j,$	$x,$	$\dots$	$m_{11}j + m_{12}x + \dots,$	$\dots$	$m_{n1}j + m_{n2}x + \dots)$
4	$w_4,$	$(j,$	$y,$	$\dots$	$m_{11}j + m_{12}y + \dots,$	$\dots$	$m_{n1}j + m_{n2}y + \dots)$

Four related vectors $w_1, \dots, w_4$ are actually admissible vectors $\alpha(v_1) = (\varepsilon(v_1), \varepsilon(\mathbf{M}v_1))$, $\alpha(v_2) = (\varepsilon(v_2), \varepsilon(\mathbf{M}v_2))$, $\alpha(v_3) = (\varepsilon(v_3), \varepsilon(\mathbf{M}v_3))$ and $\alpha(v_4) = (\varepsilon(v_4), \varepsilon(\mathbf{M}v_4))$ such that

$$\varepsilon(v_1) + \varepsilon(v_2) + \varepsilon(v_3) + \varepsilon(v_4) = 0\,,$$

but we do not know the sum $\varepsilon(\mathbf{M}v_1) + \varepsilon(\mathbf{M}v_2) + \varepsilon(\mathbf{M}v_3) + \varepsilon(\mathbf{M}v_4)$.

Let σ be a parallel maps over $(\mathbb{F}_{2^m})^{2n}$. The image of 4-related vectors via σ can be seen as

		1	2	$\dots$	$n+1$	$\dots$	$2n$
1	$w_1,$	$(\sigma(i),$	$\sigma(x),$	$\dots$	$m_{11}\sigma(i) + m_{12}\sigma(x) + \dots,$	$\dots$	$m_{n1}\sigma(i) + m_{n2}\sigma(x) + \dots)$
2	$w_2^*,$	$(\sigma(i),$	$\sigma(y),$	$\dots$	$m_{11}\sigma(i) + m_{12}\sigma(y) + \dots,$	$\dots$	$m_{n1}\sigma(i) + m_{n2}\sigma(y) + \dots)$
3	$w_3^*,$	$(\sigma(j),$	$\sigma(x),$	$\dots$	$m_{11}\sigma(j) + m_{12}\sigma(x) + \dots,$	$\dots$	$m_{n1}\sigma(j) + m_{n2}\sigma(x) + \dots)$
4	$w_4^*,$	$(\sigma(j),$	$\sigma(y),$	$\dots$	$m_{11}\sigma(j) + m_{12}\sigma(y) + \dots,$	$\dots$	$m_{n1}\sigma(j) + m_{n2}\sigma(y) + \dots)$

Definition 4.6.4. *4-related vectors $w_1, \dots, w_4$ are* totally *4-related if*

$$w_1 + w_2 + w_3 + w_4 = 0.$$

Definition 4.6.5. *Given $(x, y, z, a, b, c) \in \mathbb{N}^6$ and an $(n \times n)$-matrix $\mathbf{M}$, we say that (x, y, z, a, b, c)* fits $\mathbf{M}$ *if the following sums of elements of $\det(\mathbf{M})$ are non-zero:*

- *the sums having a number of elements equal to*

$$\sum_{i=0}^{x} \binom{n-c}{i}\binom{n-b}{x-i}\binom{b-i}{y} x!y! \qquad \sum_{i=0}^{x} \binom{n-b}{i}\binom{n-c}{x-i}\binom{c-i}{z} x!z! \qquad \sum_{i=0}^{y} \binom{n-a}{i}\binom{n-c}{x-i}\binom{c-i}{z} y!z!$$

when $z = 0, x \neq 0, y \neq 0$ — *when $y = 0, x \neq 0, z \neq 0$* — *when $x = 0, y \neq 0, z \neq 0$*

- *the sums having a number of elements equal to*

$$\sum_{i=0}^{x}\sum_{j=0}^{y}\binom{n-c}{i}\binom{n-b}{x-i}\binom{n-a}{j}\binom{(n-c)-i}{y-j}\binom{c-(x-i)-j}{z}x!y!z!$$

when $x \neq 0, y \neq 0, z \neq 0$.

The main result of this section is the next theorem that gives sufficient conditions on $\mathbf{M}$ in order to make all $\sigma : V \to V$ into 4-exendible maps.

Theorem 4.6.6. *Let* $\mathbf{M}$ *be an* $(n \times n)$*-matrix, with entries in* $\mathbb{F}_{2^m}$ *such that:*

1. $\det(\mathbf{M}) \neq 0$;
2. *all the* $k \times k$ *minors are non-zero* $(0 < k < n)$;
3. *all sextuple* (x, y, z, a, b, c) *such that*
   - $0 < a,\ b,\ c \leq n$;
   - $a + b + c = 2n$;
   - $a \geq b \geq c$;
   - $0 \leq x, y, z \leq n$;
   - $x + y + z = n$;
   - $x < a,\ y < b,\ z < c$;

   fit $\mathbf{M}$.

Then any 4*-related vectors are totally related if and only if they are coupled.*

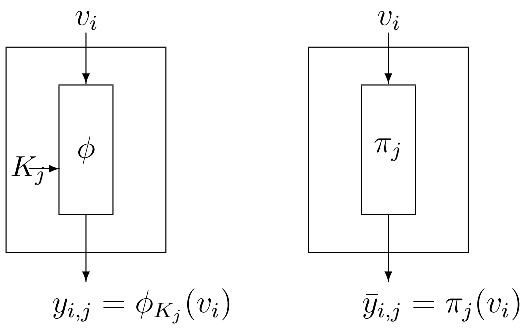
Thanks to Theorem 4.6.6 and Remark 4.6.2, we have the following

Corollary 4.6.7. *In the hypothesis of Theorem 4.6.6, any map is* 4*-extendible.*

A related-key attack

5.1 Related-key distinguishing attacks

Let $v_1, \dots, v_\rho$ be some related plaintexts, let $K_1, \dots, K_\tau$ be some related keys. With *related* vectors we mean that they satisfy a prescribed algebraic relations, e.g. they share half of their bytes. For any key K let ϕ_K be the encryption function associated to K. We denote by $\pi_1, \dots, \pi_\tau$ some random permutations in the message/cipher space V.



A (related-key) distinguishing attack on $\mathcal{C}$ is any algorithm able to distinguish the ciphertexts $\{y_{i,j}\}_{1\le i\le\rho, 1\le j\le\tau}$ from the random ciphertexts $\{\bar{y}_{i,j}\}_{1\le i\le\rho, 1\le j\le\tau}$.
For the full AES, no effective distinguishing attack is present in literature.

5.2 Our setting

Let $\mathcal{M} = V = (\mathbb{F}_2)^r$ be a vector space with dimension $r = mb$. Let $\mathbf{M} \in \mathrm{GL}(V)$ of order t. Let $W = (\mathbb{F}_2)^s$ be the vector space with dimension $s = 2^m bt$. Let $\alpha : V \to W$ be the space embedding defined as in (4.5)

$$\alpha(v) = (\varepsilon(v), \varepsilon(\mathbf{M}v), \dots, \varepsilon(\mathbf{M}^{t-1}v)).$$

Let $\lambda = \dim_{\mathbb{F}_2}(\langle \mathrm{Im}(\alpha)\rangle)$. We choose a small set $\mathcal{D} \subset \mathcal{M}$. For any $d \in \mathcal{D}$, we construct many plaintexts sharing most of their bytes, so that in total we obtain λ distinct plaintexts $\{v^i\}_{1\le i\le\lambda}$. We construct a $(\lambda \times r)$-matrix H with $\{v^i\}_{1\le i\le\lambda}$ as rows. We can consider $\alpha(H)$ as the $(\lambda \times 2^m bt)$-matrix with $\{\alpha(v^i)\}_{1\le i\le\lambda}$ as rows (see Section 4.3).

Remark 5.2.1. Clearly the rows of H are strongly related.

We compute the rank of $\alpha(H)$ and we check that it is very low thanks to the strong correlation among the rows. (It would be equal to $\lambda - 1$ with high probability if the rows were random.) Starting from H we construct many other matrices with the same low rank as $\alpha(H)$, having a similar correlation among their rows. In order to do this, we first construct r $(\lambda \times r)$-matrices $\{H_j\}_{1 \geq j \geq r}$ by changing the $j-$th bit of all rows of H.

From any $1 \leq j \leq r$ we construct L matrices $\{H_j^h\}$ of the same size, as follows. The i-th row of H_j can be written as $(v_1, \ldots, v_b) \in (\mathbb{F}_{2^m})^b$. For any $1 \leq h \leq L$, the i-th row of H_i^h will be $(v_1\gamma^{h-1}, \ldots, v_b\gamma^h)$, where γ is a primitive element of $\mathbb{F}_{2^m}$. If we apply α to all (rows of our) matrices, we will get $\{\alpha(H)_j^h\}_{1 \leq h \leq L\, 1 \leq j \leq r}$-matrices with size $(\lambda \times 2^m bt)$ all sharing the same rank. Note that $\bar{G}$ and $\tilde{G}$ are linearly extendible (Section 4.3).

Fixed a key $k \in \mathcal{K}$, we can encrypt all H_j^h matrices (by encrypting their rows), thus obtaining a set of Lr matrices $\{\phi_k(H_j^h)\}$. We then apply α to each of these matrices and we get a set of Lr $(\lambda \times 2^m bt)$-binary matrices $\{\mathsf{H}_1, \ldots, \mathsf{H}_{Lr}\}$.

We compute their ranks $\{\mathrm{rk}(\mathsf{H}_j)\}_{1 \leq j \leq rh}$ and consider the following four integers:

1. $R_0 = |\{j \mid \mathrm{rk}(\mathsf{H}_j) = \lambda\}|$;
2. $R_1 = |\{j \mid \mathrm{rk}(\mathsf{H}_j) = \lambda - 1\}|$;
3. $R_2 = |\{j \mid \mathrm{rk}(\mathsf{H}_j) = \lambda - 2\}|$;
4. $R_3 = |\{j \mid \mathrm{rk}(\mathsf{H}_j) \leq \lambda - 3\}|$.

Clearly $R_0 + R_1 + R_2 + R_3 = rL$.
What we have done up to now is to associate a set of numerical values (the rank distributions) to any key $k \in \mathcal{K}$. Let $\mathbb{S}_k$ be such numerical set. We now take any key k and view it as $\mathsf{k} = (k_1, \ldots, k_b) \in (\mathbb{F}_{2^m})^b$. We consider the 2^m keys built by changing the first component of k, that it, $K = \{(\beta, k_2, \ldots, k_b) \mid \beta \in \mathbb{F}_{2^m}\}$. These keys are closely related, since they differ in only one component. Our attack consists in

provide statistical evidence that the sets $\mathbb{S}_k$ for $k \in K$ are significantly different from sets obtained using random keys and similar[1] random matrices .

In the next section we see how we deploy this attack on AES-128.

[1]In the same space $\alpha(v)$, that is, the rows are admissible vectors.

5.3 The AES case

As seen in subection 4.3.1, in the AES case, we have $\mathcal{M} = V = (\mathbb{F}_2)^{128}$, we consider $\mathsf{M} : V \to V$ to be the MixingLayer of AES (its order is equal to 8) and the space embedding $\alpha : V \to W$ with $W = (\mathbb{F}_2)^{2^{15}}$ $(s = 2^m bt = 2^{15})$ is

$$\alpha(v) = (\varepsilon(v), \varepsilon(\mathsf{M}v), \dots, \varepsilon(\mathsf{M}^7 v)). \tag{5.1}$$

According to Fact 4.3.4, $\lambda = 2^m bt - (bt-1) - mb(t-1) = 31745$.
We choose a set $\mathcal{D} = \{u^1, \dots, u^5\}$ of plaintexts, $|\mathcal{D}| = 5$. For any $u^i \in \mathcal{D}$ we construct exactly $\lambda/5$ (related) rows of H as follows:

1. let $u^1 = (u_1, \dots, u_{16})$, for $1 \le i \le \lambda/5$, we construct the rows
$$v^i = (u_1, \dots, u_{13}, v_{14}(i), v_{15}(i), v_{16}(i))$$
where $v_{14}(i)$, $v_{15}(i)$ and $v_{16}(i)$ are random elements of $\mathbb{F}_{256}$ (but we enforce all rows to be distinct).

2. let $u^2 = (u_1, \dots, u_{16})$, for $\lambda/5 + 1 \le i \le 2(\lambda/5)$, we construct the rows
$$v^i = (v_1(i), u_2, \dots, u_{14}, v_{15}(i), v_{16}(i))$$
where $v_1(i)$, $v_{15}(i)$ and $v_{16}(i)$ are random elements of $\mathbb{F}_{256}$ (but we enforce all rows to be distinct).

3. let $u^3 = (u_1, \dots, u_{16})$, for $2(\lambda/5) + 1 \le i \le 3(\lambda/5)$, we construct the rows
$$v^i = (v_1(i), v_2(i), u_3 \dots, u_{15}, v_{16}(i))$$
where $v_1(i)$, $v_2(i)$ and $v_{16}(i)$ are random elements of $\mathbb{F}_{256}$ (but we enforce all rows to be distinct).

4. let $u^4 = (u_1, \dots, u_{16})$, for $3(\lambda/5) + 1 \le i \le 4(\lambda/5)$, we construct the rows
$$v^i = (v_1(i), v_2(i), v_3(i), u_4, \dots, u_{16})$$
where $v_1(i)$, $v_2(i)$ and $v_3(i)$ are random elements of $\mathbb{F}_{256}$ (but we enforce all rows to be distinct).

5. let $u^5 = (u_1, \dots, u_{16})$, for $4(\lambda/5) + 1 \le i \le \lambda)$, we construct the rows
$$v^i = (u_1, v_2(i), v_3(i), v_4(i), u_5, \dots, u_{16})$$
where $v_2(i)$, $v_3(i)$ and $v_4(i)$ are random elements of $\mathbb{F}_{256}$ (but we enforce all rows to be distinct).

If we take $\mathcal{D}$ random in $\mathcal{M}$ and we construct the corresponding (31745×32768)-matrix $\alpha(H)$, we note that its rank ranges from 23000 to 24000. In particular, our experiments it is convenient to consider an $\alpha(H)$ matrix with rank equal to 23551. We then proceed to construct $\{H_j^h\}_{1\le j\le 128,\, 1\le h\le 60}$, where we take γ as the root of the polynomial $x^8+x^4+x^3+x^2+1$, with $L=60$. This is a total of 7680 matrices whose images via α share the same rank 23551.

Fixed a key $k \in \mathcal{K} = (\mathbb{F}_2)^{128}$, we can encrypt all H_j^h matrices (by encrypting their rows), thus obtaining a set of 7680 matrices $\{\phi_k(H_j^h)\}$. We then apply α to each of these matrices and we get a set of 7680 (31745×32768)-binary matrices $\{\mathsf{H}_1,\dots,\mathsf{H}_{7680}\}$. We compute their ranks $\{\mathrm{rk}(\mathsf{H}_j)\}_{1\le j\le 7680}$ and consider the following four integers:

1. $R_0 = |\{j \mid \mathrm{rk}(\mathsf{H}_j) = 31745\}|$;
2. $R_1 = |\{j \mid \mathrm{rk}(\mathsf{H}_j) = 31744\}|$;
3. $R_2 = |\{j \mid \mathrm{rk}(\mathsf{H}_j) = 31743\}|$;
4. $R_3 = |\{j \mid \mathrm{rk}(\mathsf{H}_j) \le 31742\}|$.

Clearly $R_0+R_1+R_2+R_3 = 7680$. So, for any key $k \in (\mathbb{F}_2)^{128}$, we get our set $\mathbb{S}_k$. We have chosen $\mathsf{k} = (0,\dots,0) \in (\mathbb{F}_{256})^{16}$. We consider the 256 keys built by changing the first byte of k, that it, $K = \{(\beta, \underbrace{0,\dots,0}_{15}) \mid \beta \in \mathbb{F}_{256}\}$.

We then get in principle 256 sets $\mathbb{S}_k$, $k \in K$. However, in practice we have been able to obtain 37 sets $\mathbb{S}_k$, since for each we need a computational effort of 15 days with a core processor (Intel Xeon X5460, 3.16 GHz). These 37 sets $\mathbb{S}_k$ form a sample that we test against a random sample of 37 value sets.

5.4 Numerical results

In the following figures we plot values coming from $\{\mathbb{S}_k\}$ (stars) and the random values (dots), as follows

1. in Fig. 5.1 we plot R_3 on the x-axis and R_1 on the y-axis;
2. in Fig. 5.2 we plot R_0 on the x-axis and R_1 on the y-axis;
3. in Fig. 5.3 we plot R_0 on the x-axis and R_3 on the y-axis;
4. in Fig. 5.4 we plot R_2 on the x-axis and R_3 on the y-axis.

In Fig. 5.3 and Fig. 5.4 dots and stars scatter in a undistinguishable way, but in Fig. 5.1 and Fig. 5.2 we note that the stars tend to huddle together in a horizontal strip.

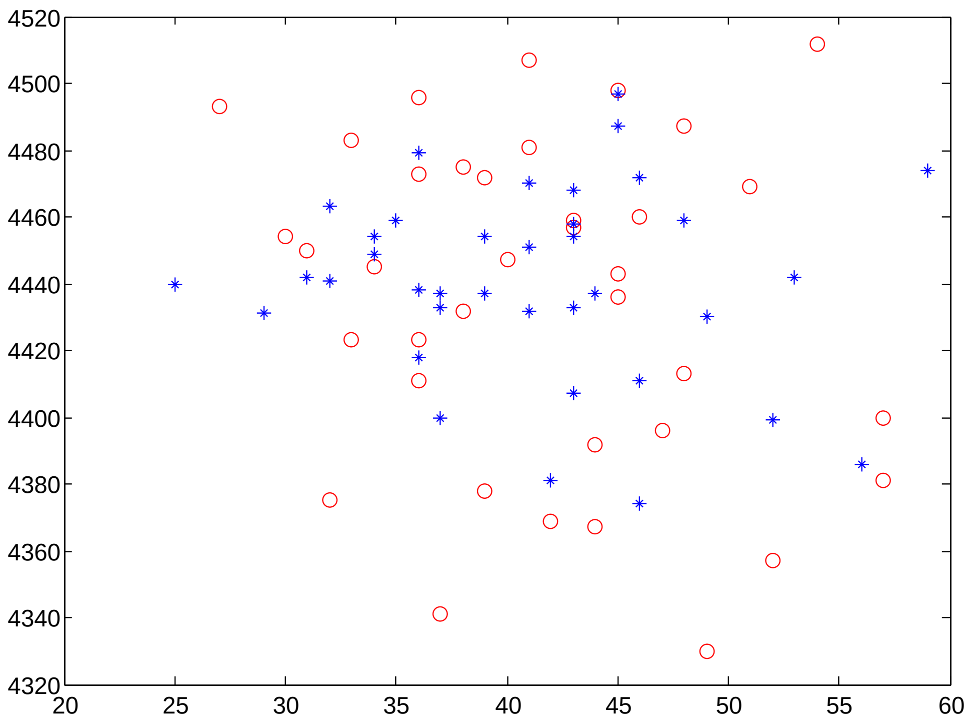


Figure 5.1: R_3 vs R_1

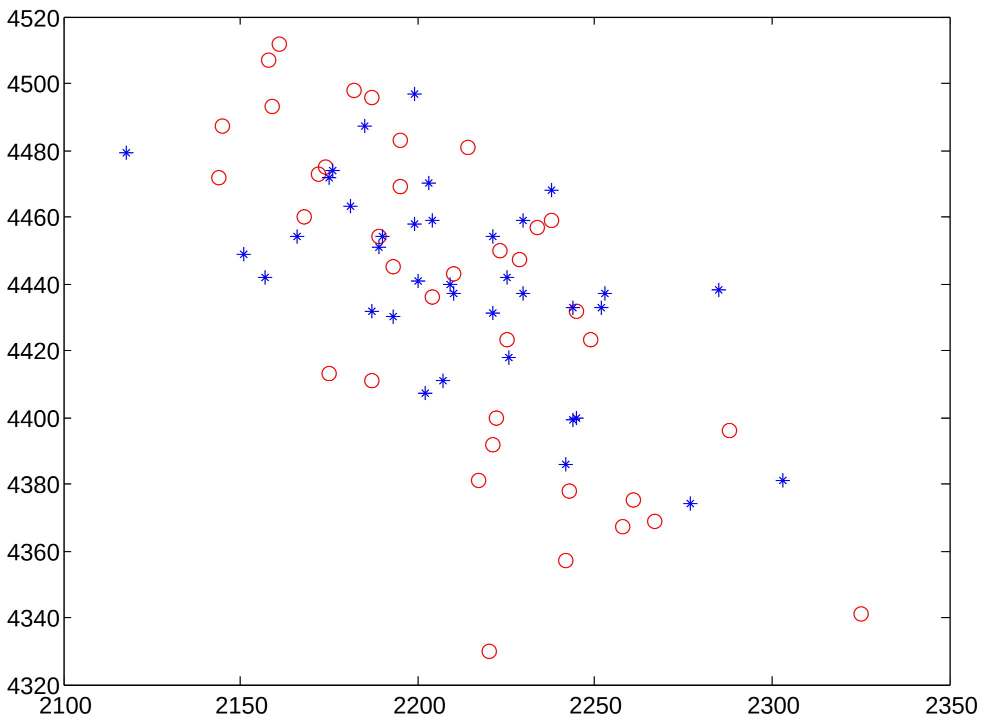


Figure 5.2: R_0 vs R_1

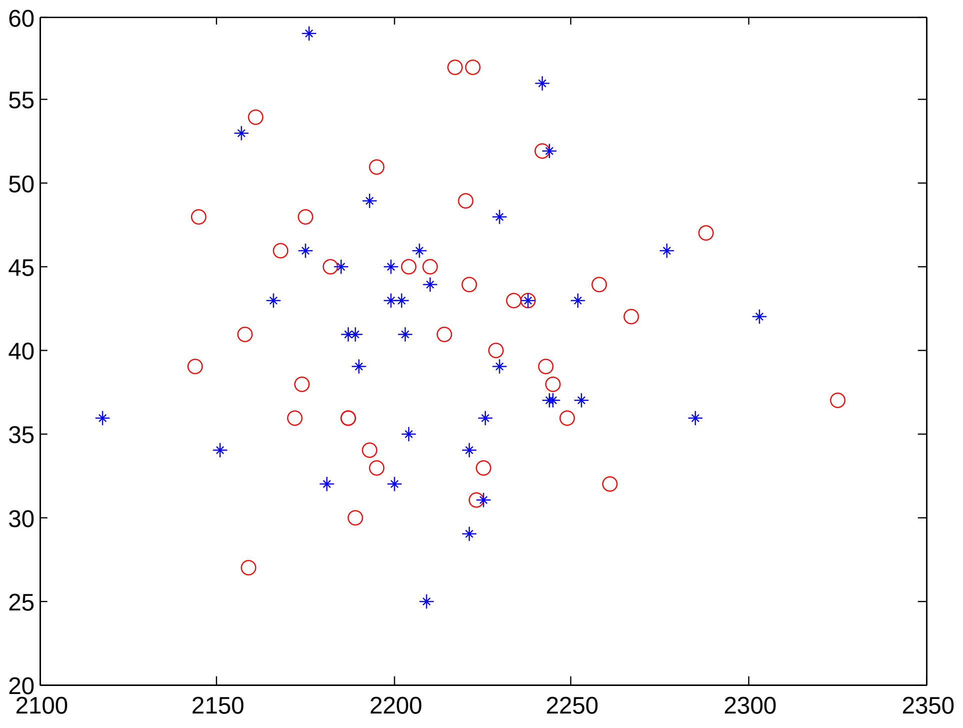


Figure 5.3: R_0 vs R_3

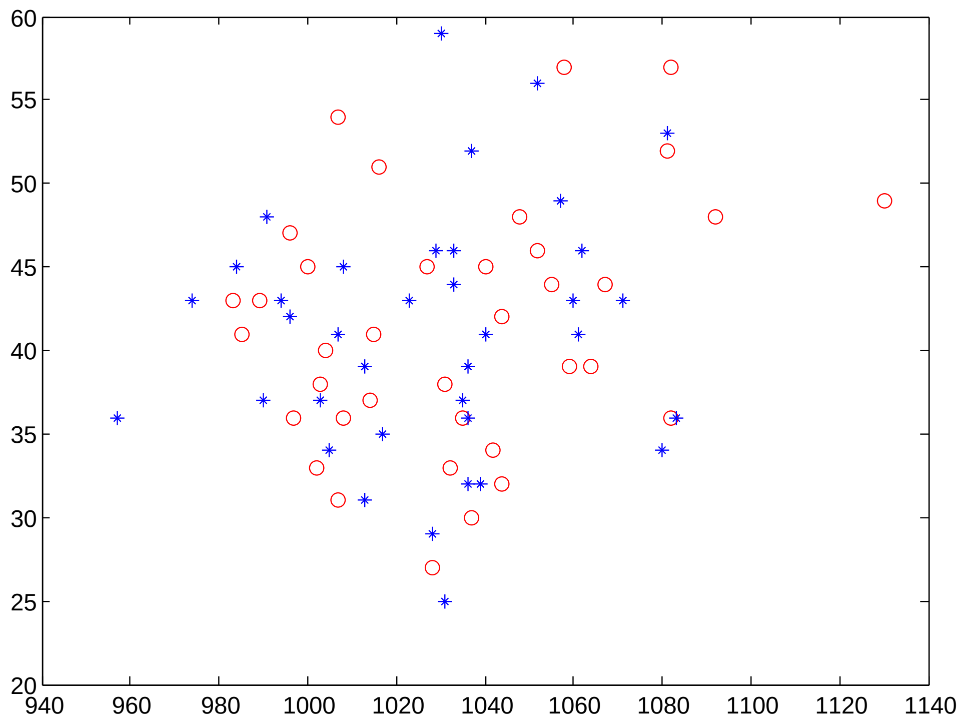


Figure 5.4: $R_2 + R_3$ vs R_3

This suggests that we would need to look at R_1 in order to find some statistical evidence. The expected R_1-mean is 4436, the mean of dots is 4435 and the mean of stars is 4440. Although our mean is further from the expected mean than the dots mean (which is nearly the same as the expected), the difference is not statistical significant and a Marsaglia-type test fails. However, it is not the mean that concerns us, but the fact that the stars are closer to their own mean. To point out the difference in statistical terms, we consider the number of stars within the expected standard deviation range ($4386 \leq R_1 \leq 4486$) and we compare it with the expected number of values within the same range. The χ^2 test gives in this case a probability of 0.5% that the stars come from a random sample. This is a very strong confidence margin. To check the effectiveness of our test, we took four random samples and we applied it. From the χ^2 test we obtain probabilities of 63%, 64%, 28%, 24%, which do not show any significant difference and are much higher than the probability obtained by our related-key sample. Therefore, our test is validated.

5.5 Comments

There are several comments and remarks that we feel are relevant for our attack:

- The matrix H that we constructed in the AES case is very closely correlated, since all rows come from just five original rows, with only three bytes varying; we did try with less closely correlated matrices (for instance, with more free bytes, or with more starting rows), but we failed to note any significant deviation from the random behaviour.
  We also tried to increase further the correlation (by reducing the number of free bytes and/or the number of rows in D), but as a result we had a large number of collisions (i.e. of rows which were the same) and we could not make a working matrix.
  Therefore, our choice of $|D| = 5$ and 3 free bytes appear optimal, given our present data.

- We tried to transform our related-key distinguishing attack into a single-key distinguishing attack, but taking related matrices with a random key did not show any statistical deviation.
  However, we tried a single-key version of our attack on reduced versions of AES128 and it does show some statistical deviation for a few rounds. Since we were not so interested in reduced versions of AES, we do not report here there results.

- We have mounted our attack also for the PRESENT cipher. Although the cipher is much smaller, the attack is not much simpler, because we cannot exploit the byte-oriented structure held by AES. Our preliminary results are encouraging, but we do not report them here.
- Although algebraic setting for the attack is rather involved, the statistical test itself is very easy. We can design more sophisticated test, but we need more $\mathbb{S}_k$ to apply them with significance.

Appendix

Appendix

Here we put the S-boxes of DES.

S_1															
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2															
15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	7	6	12	0	5	14	9

S_3															
10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_4															
7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S_5															
2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	5	5	3

S_6															
12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7															
4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_8															
13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Bibliography

[ABKL$^+$07] A. Andrey Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe, *PRESENT: An ultra-lightweight block cipher*, Proc. of CHES 2007, LNCS, vol. 4727, Springer, 2007, pp. 450–466.

[AC08] M. Albrecht and C. Cid, *Algebraic techniques in differential cryptanalysis*, Crypto. ePrint Arch., Rep. 2008/177, 2008, http://eprint.iacr.org/.

[BAK98] E. Biham, R.J. Anderson, and L.R. Knudsen, *Serpent: A new block cipher proposal*, Proc. of FSE 1998, LNCS, vol. 1372, Springer, 1998, pp. 222–238.

[BB02a] E. Barkan and E. Biham, *The Book of Rijndaels*, Tech. report, IACR ePrint Report, 2002/158, 2002.

[BB02b] ———, *In how many ways can you write Rijndael?*, Proc. of ASIACRYPT 2002, LNCS, vol. 2501, 2002, pp. 160–175.

[BCD$^+$98] C. Burwick, D. Coppersmith, E. D'Avignon, R. Gennaro, S. Halevi, C. Jutla, S. M. Matyas, L. O'Connor, M. Peyravian, D. Safford, and Zunic N., *MARS — A candidate cipher for AES*, NIST AES Proposal, 1998.

[BD07] M. Brickenstein and A. Dreyer, *PolyBoRi: A framework for Gröbner basis computations with Boolean polynomials*, Elec. Proc. of MEGA 2007, 2007, http://www.ricam.oeaw.ac.at/mega2007/electronic/26.pdf.

[BDCBP03] A. Biryukov, C. De Canniere, A. Braeken, and B. Preneel, *A toolbox for cryptanalysis: Linear and affine equivalence algorithms*, Proc.of EUROCRYPT 2003, LNCS, vol. 2656, 2003, pp. 33–50.

[BDK05] E. Biham, O. Dunkelman, and N. Keller, *Related-key boomerang and rectangle attacks*, Proc. of EUROCRYPT 2005, LNCS, vol. 3494, 2005, pp. 507–525.

[BK00] E. Biham and N. Keller, *Cryptanalysis of reduced variants of Rijndael*, Proc. of AES3, 2000.

[BK09] A. Biryukov and D. Khovratovich, *Related-key Cryptanalysis of the Full AES-192 and AES-256*, Tech. report, IACR, 2009, http://eprint.iacr.org/2009/317.

[BPW06a] J. Buchmann, A. Pyshkin, and R. P. Weinmann, *Block ciphers sensitive to Gröbner basis attacks*, Proc. of CT-RSA 2006, LNCS, vol. 3860, Springer, 2006, pp. 313–331.

[BPW06b] ______, *A zero-dimensional Gröbner basis for AES-128*, Proc. of FSE 2006, LNCS, vol. 4047, Springer, 2006, pp. 78–88.

[BS93] E. Biham and A. Shamir, *Differential cryptanalysis of DES-like cryptosystems*, J. of Cryptology **4** (1993), 3–72.

[Buc65] Bruno Buchberger, *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal*, Ph.D. thesis, Innsbruck, 1965.

[Buc06] B. Buchberger, *Bruno Buchberger's PhD thesis 1965: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal*, J. Symb. Comput. **41** (2006), no. 3-4, 475–511.

[Cam99] P. J. Cameron, *Permutation groups*, London Mathematical Society Student Texts, vol. 45, Cambridge University Press, Cambridge, 1999.

[Carar] C. Carlet, *Boolean methods and models*, ch. Boolean Functions for Cryptography and Error Correcting Codes, Cambridge University Press, to appear.

[CDVSar] A. Caranti, F. Dalla Volta, and M. Sala, *On some block ciphers and imprimitive groups*, AAECC (to appear), 10.

[CGC03] *NESSIE D20 - NESSIE security report*, 2003, http://citeseer.ist.psu.edu/568219.html.

[CKK$^+$01] J.H. Cheon, M. Kim, K. Kim, J.Y. Lee, and S. Kang, *Improved impossible differential cryptanalysis of Rijndael and Crypton*, Proc. of ICISC 2001, LNCS, vol. 2288, 2001, pp. 39–49.

[CKPS00] N. Courtois, A. Klimov, J. Patarin, and A. Shamir, *Efficient algorithms for solving overdefined systems of multivariate polynomial equations*, Proc. of EUROCRYPT 2000, LNCS, vol. 1807, Springer, 2000, pp. 392–407.

[CMR05] C. Cid, S. Murphy, and M. J. B. Robshaw, *Small scale variants of the AES*, Proc. of FSE 2005, LNCS, vol. 3557, Springer, 2005, pp. 145–162.

[CMR07] ______, *Algebraic aspects of the Advanced Encryption Standard*, Springer, 2007.

[CP02] N. Courtois and J. Pieprzyk, *Cryptanalysis of block ciphers with overdefined systems of equations*, Proc. of ASIACRYPT 2002, LNCS, vol. 2501, Springer, 2002, pp. 267–287.

[CW09] C. Cid and R. P. Weinmann, *Block ciphers: algebraic cryptanalysis and Gröbner bases*, Gröbner Bases, Coding, and Cryptography (M. Sala, T. Mora, L. Perret, S. Sakata, and C. Traverso, eds.), RISC Book Series, Springer, Heidelberg, 2009, p. to appear.

[CYK09] D. L. Cook, M. Yung, and A. D. Keromytis, *Elastic block ciphers: method, security and instantiations*, Int. J. Inf. Sec **8** (2009), no. 3, 211–231.

[Dar08] M. R. Darafsheh, *The maximum element order in the groups related to the linear groups which is a multiple of the defining characteristic*, Finite Fields Appl. **14** (2008), no. 4, 992–1001.

[DKR97] J. Deamen, L. Knudsen, and V. Rijmen, *The block cipher Square*, Proc. of FSE 97, LNCS, vol. 1267, 1997, pp. 149–165.

[DM96] J. D. Dixon and B. Mortimer, *Permutation groups*, vol. 163, Springer-Verlag, 1996.

[DR98] J. Daemen and V. Rijmen, *AES proposal: Rijndael*, Tech. report, NIST, 1998.

[DR02] ______, *The Design of Rijndael*, Springer, 2002.

[FKL$^+$00] N. Ferguson, J. Kesley, S. Lucks, B. Schneier, M. Stay, D. Wagner, and D. Whitinf, *Improved cryptanalysis of Rijndael*, Proc. of FSE 2000, LNCS, vol. 1978, Springer, 2000, pp. 213–230.

[FSW01] N. Ferguson, R. Schroeppel, and D. Whitinf, *A simple algebraic representation of Rijndael*, Proc. of SAC 2001, LNCS, vol. 2259, 2001, pp. 103–111.

[GM00] H. Gilbert and M. Minier, *A collision attack on seven rounds of Rijndael*, Proc. of AES3, 2000.

[KKS00] J. Kelsey, T. Kohno, and B. Schneier, *Amplified Boomerang attack against reduced-round MARS and Serpent*, Proc. of FSE 2000, LNCS, vol. 1978, 2000, pp. 75–93.

[KL90] P. Kleidman and M. Liebeck, *The subgroup structure of the finite classical groups*, London Math. Soc. LNS, vol. 129, Cambridge University Press, 1990.

[Knu99] L. Knudsen, *Contemporary block ciphers*, LNCS **1561** (1999), 105–126.

[Lan03] E. Landau, *Ueber die maximalordung der permutation gegbenen grades*, Arch. der Math. und Phys. **5** (1903), no. 3, 92–103.

[LK07] C. Lim and K. Khoo, *An Analysis of XSL applied to BES*, Proc. of FSE 2007 (A. Biryukov, ed.), LNCS, vol. 4593, Springer, 2007, pp. 242–253.

[LN97] R. Lidl and H. Niederreiter, *Finite fields*, Encyclopedia of Mathematics and its Applications, Cambridge University Press, 1997.

[Mai09] Lara Maines, *Una debole rappresentazione del gruppo simmetrico*, Master's thesis (laurea specialistica), University of Trento, Department of Mathematics, 2009.

[Mat93] M. Matsui, *Linear cryptanalysis method for DES cipher*, Proc. of EUROCRYPT 93, LNCS, vol. 765, 1993, pp. 386–397.

[MMM04] T. Migler, K. E. Morrison, and O. Mitchell, *Weight and rank of matrices over finite fields*, Tech. report, arxiv, 2004, http://arxiv.org/abs/math/0403314.

[MR02] S. Murphy and M. J. B. Robshaw, *Essential algebraic structure within the AES*, Proc. of CRYPTO 2002, LNCS, vol. 2442, Springer, 2002, pp. 1–16.

[Nat77] National Bureau of Standards, *The Data Encryption Standard*, Federal Information Processing Standards Publication (FIPS) 46, 1977.

[Nat01] National Institute of Standards and Technology, *The Advanced Encryption Standard*, Federal Information Processing Standards Publication (FIPS) 197, 2001.

[NIS00] *A statistical test suite for random and pseudorandom number generators for cryptographic applications*, Special Publication SP 800-22, NIST, 2000, http://csrc.nist.gov/publications/nistpubs/800-22/sp-800-22-051501.pdf.

[PGC98] J. Patarin, L. Goubin, and N. Courtois, *Improved algorithms for isomorphisms of polynomials*, Proc. of EUROCRYPT 1998, LNCS, vol. 1403, 1998, pp. 184–200.

[Pha04] R. C. W. Phan, *Impossible differential cryptanalysis of 7-round advanced encryption standard (AES)*, Inform. Process. Lett. **91** (2004), no. 1, 33–38.

[pol] *The software package PolyBori - Polynomials over Boolean Rings*, http://polybori.sourceforge.net/.

[Riv92] R. L. Rivest, *The MD5 message-digest algorithm*, Internet RFC 1321, 1992.

[RRY00] R. L. Rivest, M. J. B. Robshaw, and Y. L. Yin, *RC6 as the AES*, Proc. of AES III, 2000, pp. 337–342.

[Rue92] R. Rueppel, *Stream ciphers*, Contemporary cryptology - The science of information integrity, IEEE Press, 1992, pp. 65–134.

[Sch98] B. Schneier, *The Twofish encryption algorithm*, Dr. Dobb's Journal of Software Tools **23** (1998), no. 12, 30–38.

[Sha49] C. E. Shannon, *Communication theory of secrecy systems*, Bell System Tech. J. **28** (1949), 656–715.

[Sot98] J. J. Soto, *Randomness testing of the AES candidate algorithms*, Proc. of AES candidate conference I (National Institute of Standards and Technology, ed.), NIST, 1998, http://csrc.nist.gov/encryption/aes/round1/r1-rand.pdf, p. 9.

[Sti95] D. R. Stinson, *Cryptography, Theory and Practice*, CRC Press, 1995.

[SW08] R. Sparr and R. Wernsdorf, *Group theoretic properties of Rijndael-like ciphers*, Discrete Appl. Math. **156** (2008), no. 16, 3139–3149.

[TZ05] I. Toli and A. Zanoni, *An algebraic interpretation of AES-128*, Proc. of AES 2004, LNCS, vol. 3373, Springer, 2005, pp. 84–97.

[Wag76] A. Wagner, *The faithful linear representation of least degree of* s_n *and* a_n *over field of characteristic* 2., Math. Z. **151** (1976), no. 2, 127–137.

[Wag99] D. Wagner, *The Boomerang attack*, Proc. of FSE 1999, LNCS, vol. 1636, 1999, pp. 156–170.

[Wer02] R. Wernsdorf, *The round functions of Rijndael generate the alternating group*, Proc. of FSE 2002, LNCS, vol. 2365, Springer, 2002, pp. 143–148.

[WLL04] S. Wu, S. Lu, and C. Laih, *Design of AES based on dual cipher and composite field*, Proc. of CT-RSA 2004, LNCS, vol. 2964, 2004, pp. 25–38.

[WOL02] J. Wolkerstorfer, E. Oswald, and M. Lamberger, *An ASIC implementation of the AES SBoxes*, CT-RSA: The Cryptographers' Track at RSA Conference, LNCS, vol. 2271, 2002, pp. 67–68.