Treball final del grau de Matemàtiques

Anàlisi algebraica de l'AES

Aleix Bou Comas

Director: Artur Travesa
Departament de Matemàtiques i Informàtica
Barcelona, 19 de gener de 2018

Abstract

Since 2001, AES symmetric private key criptosystem is being used as one of the standards to encrypt a huge amount of messages. AES is analysed from an algebraic point of view using finite field theory, a less common approach in the available standard literature. To finish, the security of the cryptosystem is assessed.

Resum

Des del 2001 s'està utilitzant com a un dels criptosistemes estàndards per xifrar una gran quantitat de missatges l'AES, un criptosistema de clau privada i simètrica. Utilitzant recursos de cossos finits i d'àlgebra en general, es pretén analitzar l'AES des d'un vessant algebraic, poc habitual a la literatura estàndard. Finalment, es discuteix la seguretat del criptosistema.

Agraïments

Primer de tot m'agradaria donar les gràcies a l'Artur Travesa per la seva dedicació i la seva ajuda constant al llarg de tot el treball. També m'agradaria agrair a la meva família per permetre que hagi pogut arribar fins aquí. I als meus amics per tot el seu suport.

Contingut

Introducció

El criptosistema Rijndael es va adoptar com a criptosistema estàndard pel NIST (Nacional Institute of Standard and Technologies) al 2001 i es va començar a anomenar com AES (Advanced Encryption Standard). Aquest canvi va ser proposat després de veure que la seguretat del DES, anterior criptosistema estàndard, es podia veure compromesa en un futur relativament proper.

L'AES és un critosistema de clau privada i simètrica; és a dir, s'utilitza la mateixa clau per xifrar i per desxifrar el missatge. L'AES va ser escollit pel NIST d'entre 15 altres criptosistemes per la seva seguretat, la seva facilitat d'implementar, la rapidesa en xifrar i desxifrar els missatges. Un dels atributs que més destaca és que es poden utilitzar tres mides diferents de claus.

Aquest treball pretén explicar l'AES des d'un punt de vista algebraic, explicant de forma matemàtica cada transformació, a diferència de la gran part de la literatura que explica l'AES de forma pràctica pensant en la seva implementació. Posteriorment, també es realitza una discussió de les possibles maneres d'atacar el criptosistema, i el resultat obtingut a partir d'aquests atacs fins ara.

La memòria està estructurada en quatre capítols que formen tres blocs. El primer bloc està format per un primer capítol introductori on es repassen certs conceptes necessaris per descriure el criptosistema i es fixen les notacions que es faran servir. El segon bloc és format pels capítols 3 i 4; s'hi descriuen els processos de xifratge i de desxifratge de l'AES. Finalment, el tercer bloc, format per l'últim capítol, on es descriuen diferents maneres d'atacar l'AES i el resultat d'aquests atacs.

Al primer capítol hi ha una petita introducció a la criptogràfia bàsica i els resultats més rellevants que cal saber per tal de poder seguir més endavant els raonament sobre seguretat. Com que l'AES utilitza el cos finit $\mathbb{F}_2$ per representar els bits i també utilitza un cos finit de grau 8 sobre $\mathbb{F}_2$ per expressar els bytes, es fa un resum de la teoria necessària de cossos finits així com una breu discussió sobre com implementar-los a l'ordinador. Per acabar també s'introdueix com cal preparar el missatge per tal de xifrar-lo ja que, com veurem, no totes les comunicacions poden ser xifrades directament; a vegades, cal un tractament previ.

El segon i tercer capítols formen el bloc principal, i s'hi descriu exhaustivament l'AES així com totes les funcions que el conformen i les inverses respectives. En particular, i en tractar-se d'una exposició matemàtica del criptosistema, s'ha cregut convenient incloure una demostració rigorosa del fet que els processos de xifratge i de desxifratge són bijectius i l'un invers de l'altre, una demostració que no està inclosa als estàndards.

En el quart i últim capítol, es justifica que l'AES no és un xifratge afí i per què no funcionen alguns tipus de criptoanàlisis que s'havien utilitzat per atacar criptosistemes

anteriors a l'AES. Seguidament, es fa una breu exposició de les equacions que s'haurien de resoldre per tal de trencar el criptosistema per força bruta i s'arriba a la conclusió que amb l'estat actual de l'àlgebra i la tecnologia, el criptosistema és robust davant dels mètodes criptoanalítics convencionals.

Capítol 1

Preliminars

1.1 Criptografia

La criptogràfia té com a objectiu permetre que dos dispositius qualssevol, que s'anomenen l'emissor i el receptor, es puguin comunicar a través d'un canal insegur de manera que cap tercer pugui entendre la comunicació transmesa.
El missatge preparat per al xifratge que l'emissor vol transmetre al receptor s'anomena text pla, i pot ser de qualsevol naturalesa ja sigui alfabètica, numèrica, en codi binari, etcètera. L'emissor per tal de transmetre el seu missatge pla de forma segura el xifra utilitzant una clau, consensuada amb el receptor, i envia el missatge xifrat a través del canal insegur. L'objectiu de xifrar el missatge és que si un tercer l'intercepta no sigui capaç d'entendre'l, però, en canvi, si el rep el receptor que posseeix la clau, sigui capaç de desxifrar el text xifrat i reconstruir el missatge pla.

Un model matemàtic útil per a l'estudi d'aquets fenòmens es pot descriure a partir de la definició següent.

Definició 1.1.1. Un criptosistema és una 5-pla $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ on

1. $\mathcal{P}$ és un conjunt finit, anomenat de textos plans;
2. $\mathcal{C}$ és un conjunt finit, anomenat de textos xifrats;
3. $\mathcal{K}$, és un conjunt finit, anomenat de claus;
4. $\mathcal{E} = \{e_k\}_{k\in\mathcal{K}}$ i $\mathcal{D} = \{d_k\}_{k\in\mathcal{K}}$ són dues famílies d'aplicacions, $e_k : \mathcal{P} \to \mathcal{C}$ i $d_k : \mathcal{C} \to \mathcal{P}$ anomenades xifratge i desxifratge tals que $\forall k \in \mathcal{K}$ i $\forall x \in \mathcal{P}$ se satisfà que $d_k(e_k(x)) = x$.

La comunicació que es vol transmetre entre l'emissor i el receptor s'ha de transformar en una successió d'elements de $\mathcal{P}$ i, per tant, no pot ser qualsevol. Això fa que moltes comunicacions abans de poder ser xifrades s'hagin de tractar. Aquest és el cas de l'AES que només permet xifrar comunicacions de 128 bits; si aquestes tenen una mida major s'han de trencar en blocs de 128 bits per tal de poder xifrar-los després.

Per altra banda, la propietat clau per tal que el criptosistema funcioni és la número 4 on s'afirma que donada una clau, aplicant les funcions de xifratge i de desxifratge s'obté el missatge pla.

Per tal d'enviar aquesta comunicació, primerament els dispositius que es volen intercanviar el missatge pla s'han de posar en contacte a través d'un canal segur per escollir les claus de manera que no la sàpiga cap tercer.

Suposem que un dispositiu li vol enviar la comunicació següent $x = x_1x_2\cdots x_n$, on cada $x_i \in \mathcal{P}$, a l'altre dispositiu amb el qual, prèviament, han acordat una clau $k \in \mathcal{K}$. Aleshores l'emissor calcula $y_i = e_k(x_i)$ i obté el text xifrat $y = y_1y_2\cdots y_n$. Quan el segon dispositiu rep el missatge només ha de computar $x_i = d_k(y_i)$ per obtenir el conjunt de missatges plans; es a dir, la comunicació. Clarament, la funció e_k ha de ser injectiva, ja que en cas contrari, existirien $x_1, x_2 \in \mathcal{P}$ amb $x_1 \neq x_2$, tals que $y = e_k(x_1) = e_k(x_2)$ i el receptor no tindria manera de saber quin dels dos resultats escollir.

Un tipus d'atac, utilitzat sobretot contra els primers criptosistemes, és l'anàlisi de freqüències. Aquesta anàlisi es basa en el fet que en totes les llengües exsiteix una distribució característica que dóna compte de la freqüència amb què apareix cada lletra o caràcter. Aquesta distribució es manté aproximadament constant en qualsevol entitat lingüística suficientment llarga. El primer estudi matemàtic que es conserva en anglès el van fer *Beker & Piper* utilitzant diverses noveles, revistes i diaris. No només van fer un estudi de freqüència de caràcters sinó que van estendre la distribució de freqüències a successions de lletres com per exemple TH o FF. Un cop es té en compte això es poden desxifrar missatges xifrats a través de criptosistemes senzills comptant el nombre de vegades que es repeteix cada un dels caràcters i subtituint-los en funció de les freqüències. En cas que aquesta primera anàlisi no doni resultat, es pot estendre a successions de lletres o utilitzant algunes variants.

El xifratge afí és un tipus de criptosistema que pren especial rellevància en l'AES ja que aquest inclou un criptosistema afí amb alguns canvis significatius.

El xifratge afí té la següent estructura. Sigui K un anell, i $E = K^n$, amb $n \geq 1$ un nombre natural; una aplicació afí de E en E és una aplicació de la forma $x \longmapsto a(x) + b$, on $a \in Aut_K(E)$ i $b \in E$, fixats. És fàcil veure que conforma un criptosistema, ja que si es consideren, $\mathcal{P} = \mathcal{C} = E$, $\mathcal{K} = \{(a, b) | a \in Aut_K(E), b \in E\}$ aleshores es compleix que

$$\begin{cases} y = e_k(x) = a(x) + b, \\ d_k(y) = inv(a)(y - b), \end{cases} \tag{1.1.1}$$

on inv correspon a l'invers de l'automorfisme a.

Originalment, aquest sistema s'utilitzava amb $n = 1$. Es considerava cada lletra de l'abecedari com un element de $\mathbb{Z}/26\mathbb{Z}$; és a dir, $\mathcal{P} = \mathcal{C} = \mathbb{Z}/26\mathbb{Z}$ i $\mathcal{K} = (\mathbb{Z}/26\mathbb{Z})^* \times \mathbb{Z}/26\mathbb{Z}$. Per xifrar cada lletra de la comunicació d'utilitzava la regla escrita anteriorment $y_i = ax_i + b$; es pot veure que aquest sistema no és segur a través de l'anàlisi de freqüències.

L'AES inclou una funció anomenada *SubBytes* que és parcialment afí: $\mathcal{P} = \mathcal{C} = \mathbb{F}_2^8$, amb $\mathcal{K} = \{(A, B) | A \in GL(\mathbb{F}_2, 8), B \in \mathbb{F}_2^8\}$. De la mateixa manera, es trenca la comunicació en successions de 8 bits i es xifra: $y_i = Ax_i + B$. En aquest cas la funció també és insegura a través de l'anàlisi de freqüències, però a l'AES s'introdueixen una serie de canvis en aquesta funció que la doten de més seguretat.

Claude Shannon va introduir nocions matemàtiques a la teoria de la informació, per tal de poder estudiar la seguretat dels criptosistemes ([Shannon 1949]). En el seu estudi va introduir dos conceptes que qualsevol criptosistema requereix per ser segur, que són: la difusió i la confusió.

La *difusió* implica que l'ordre del text pla ha de ser dispersat per tot el text xifrat. A la pràctica això implica que si es canvia un càracter del text pla (i la clau es manté constant) aleshores molts dels caràcters del text xifrat també canvien. Això matemàticament implica que la freqüència estadística de lletres al text pla estarà difosa al llarg de tot el text xifrat, i per tant, es necessita molta més extensió de missatge xifrat per plantejar un atac estadístic significatiu.

La *confusió* vol dir que la relació entre el text pla i el text xifrat ha de ser tan aleatòria com sigui possible, és a dir; si es canvia un caràcter de la clau (sense alterar el text pla) s'haurien de canviar diversos caràcters del text xifrat.

En la figura 1.1 es pot apreciar la confusió i la difusió en l'AES, i quina es la diferència entre els dos conceptes a la pràctica. Com es pot veure, aparentment, el criptosistema té unes bones propietats ja que no es pot relacionar el text xifrat amb la clau ni el text pla de forma òbvia.

String Representations (N)	
Plaintext:	00112233445566778899AABBCCDDEEFF
Key:	000102030405060708090A0B0C0D0E0F
Ciphertext:	69C4E0D86A7B0430D8CDB78070B4C55A
String Representations (D)	
Plaintext:	000022334455667788 99AABBCCDDEEFF
Key:	000102030405060708090A0B0C0D0E0F
Ciphertext:	12BF03D636F8173D7D71FCA75E072970
String Representations (C)	
Plaintext:	00112233445566778899AABBCCDDEEFF
Key:	000002030405060708090A0B0C0D0E0F
Ciphertext:	CEA3C4E0A352F54875B7E57F03CDFF6D

Figura 1.1: Taula de text pla, clau i text xifrat respectivament per l'AES-128. La primera, N és utilitzant un text pla i clau qualsevol; en D es fa variar el text pla per poder veure l'efecte que té sobre el text xifrat (amb això es pot fer una primera comprovació de la difusió aparent de l'AES); finalment, en C, es mostra com afecta al text xifrat una petita variació de la clau. Per tal de xifrar aquests missatges plans s'ha utilitzat el programa de [Project Nayuki 2013]

1.2 Cossos finits

Un element bàsic per a la descripció de l'AES des del punt de vista algebraic és la teoria dels cossos finits. El primer cos finit que cal presentar és el cos finit de dos elements, $\mathbb{F}_2 \cong \mathbb{Z}/2\mathbb{Z}$, que s'utilitza per representar els bits, caràcters binaris formats per 0 o 1.

Com a model del byte, podem pendre un espai vectorial de dimensió 8 sobre $\mathbb{F}_2$, de manera que la suma de bytes es realitzarà component a component. L'AES ho fa prenent

aquest $\mathbb{F}_2$-espai vectorial com el cos finit $\mathbb{F}_{2^8}$ que dota l'AES d'una estructura més rica. Per a això es considera el polinomi $m(x) = x^8 + x^4 + x^3 + x + 1 \in \mathbb{F}_2[x]$ que es irreductible i llavors $\mathbb{K} = \mathbb{F}_{2^8} \cong \frac{\mathbb{F}_2[x]}{\langle m(x) \rangle}$. La tria d'aquest polinomi és característica per a l'AES, però com que tots els cossos finits de 2^8 elements són isomorfs entre ells, en cas d'haver escollit un altre polinomi, sempre es podria trobar un isomorfisme que relacioni els dos cossos finits.

Un element qualsevol del cos finit $\mathbb{K}$ es podrà representar, doncs, com la classe d'algun polinomi de la forma $a = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5 + a_6x_6 + a_7x^7$ on $a_i \in \mathbb{F}_2$. Agafant com a base de $\mathbb{K}$ el conjunt ordenat $\{1, x, x^2, x^3, x^4, x^5, x^6, x^7\}$, es pot expressar qualsevol $a = (a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7)$ ja que es pot mirar $\mathbb{K}$ com un espai vectorial de dimensió 8 sobre $\mathbb{F}_2$. Per fer la notació més compacta, molts cops s'expresen els bytes com un conjunt de dos nombres hexadecimals. Aquesta conversió es fa de dreta a esquerra i usant la taula 1.1. Es a dir, si en notació hexadecimal tenim l'element $\{63\}$ aquest com a element de $\mathbb{K}$ equivaldrà a $1 + x + x^5 + x^6$.

Bit	**Digit**	**Bit**	**Digit**
0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F

Taula 1.1: Taula conversora BIT a notació hexadecimal

La suma es realitza coeficient a coeficient com en qualsevol anell de polinomis, l'única característica afegida que s'ha de tenir en compte es que els coeficients pertanyen a $\mathbb{F}_2$.

Per realitzar la multiplicació s'ha de tenir en compte que el representant sempre ha de tenir grau menor que 8. Aleshores, si després de fer la multiplicació el polinomi és de grau més gran o igual que 8, s'ha de fer la divisió del polinomi resultant $p(x)$ entre $m(x)$; un cop fet això, se satisfà que $p(x) = c(x)m(x) + r(x)$, on $c(x)$ és el quocient i $r(x)$ el residu, i com que estem en un cos que identifica $m(x) = 0$, aleshores $c(x) = r(x)$.

Per dividir en aquest cos, és suficient trobar l'element invers del divisor i multiplicar-lo pel dividend.

L'AES utilitza una funció inversa lleument modificada, la denotem $inv(a)$ que associa l'element invers de a en el cos $\mathbb{K}$ si $a \neq 0$ i l'element neutre respecte la suma amb ell mateix. És a dir:

$$\begin{aligned} inv : \mathbb{K} &\longmapsto \mathbb{K} \\ a &\longmapsto a^{-1}\,, si \quad a \neq 0, \\ 0 &\longmapsto 0. \end{aligned} \tag{1.2.1}$$

De fet, donat que estem treballant en un cos finit $\mathbb{F}_{2^8}$, utilitzant que per a tot $a \in \mathbb{K}$ es compleix $a^{2^8} = a$, si resulta que $a \neq 0$ aleshores, $a^{2^8-1} = 1$ o sigui $a \cdot a^{254} = 1$. Aquest teorema ens dona l'existéncia d'element invers $\forall a \in \mathbb{K}^*$ i una manera de calcular-lo. Per

tant podem definir la funció inversa estenent-la a tots els elements de $\mathbb{K}$ de manera que l'invers del 0 vagi al 0 de la manera següent:

$$\begin{aligned} inv : \mathbb{K} &\longmapsto \mathbb{K} \\ a &\longmapsto a^{254} \quad . \end{aligned} \tag{1.2.2}$$

Per al càlcul efectiu de l'element invers existeixen com a mínim tres maneres viables. D'una banda, es podria guardar una taula d'inversos, es a dir; que cada element de $\mathbb{K}$ tingui el seu invers tabulat. Cal dir que aquest métode no es fa servir i el motiu s'explicarà més endavant.

Els altres dos mètodes de càlcul, que s'explicaran tot seguit, són el de l'algoritme ampliat d'Euclides i l'exponenciació binària.

L'algoritme d'Euclides és un algoritme que permet calcular el màxim comú divisor entre dos polinomis. Aquest, es pot ampliar de manera que aquest màxim comú divisior s'expresa com a combinació lineal dels dos polinomis.

Aquest mètode es pot fer servir en el cas estudiat. Sigui $\hat{a}(x) \in \mathbb{F}_2[x]$ el representant de grau menor que 8 de l'element $a(x) \in \mathbb{K}$ que es vol invertir. Per a calcular l'invers de l'element $a(x)$, s'agafa el seu representant $\hat{a}(x)$ i el polinomi $m(x)$ esmentat prèviament com a polinomis de $\mathbb{F}_2[x]$ i s'utilitza l'algoritme d'Euclides ampliat per trobar $b(x), c(x) \in \mathbb{K}$ tals que $\hat{a}(x)\cdot b(x)+c(x)\cdot m(x) = mcd(\hat{a}, m) = 1$, ja que $m(x)$ és un polinomi irreductible de grau 8, mentre que $\hat{a}(x)$ és per definició de grau menor o igual que 7.

L'algoritme d'Euclides es pot expressar a partir de la funció recursiva següent:

$$\left\{ \begin{array}{lll} r_0 = m(x), & r_1 = \hat{a}(x), \\ s_0 = 1, & s_1 = 0, \\ t_0 = 0, & t_1 = 1, \end{array} \right. \Longrightarrow \left\{ \begin{array}{l} r_{i+1} = r_{i-1} - q_i r_i, \\ s_{i+1} = s_{i-1} - q_i s_i, \\ t_{i+1} = t_{i-1} - q_i t_i, \end{array} \right. \tag{1.2.3}$$

on q_i representa el quocient de la divisió entre r_{i-1} i r_i.

És important veure que el grau del polinomi va disminuint a mesura que es va aplicant la funció recursiva; és a dir, $gr(r_{i-1}) > gr(r_i)$.

L'algoritme acaba quan $r_{k+1} = 0$; aleshores, $r_k = mcd(\hat{a}, m)$, i es se satisfà que $r_k(x) = s_k(x) \cdot m(x) + t_k(x) \cdot \hat{a}(x)$. És a dir, els polinomis $b(x) = t_k(x)$ mentre que $c(x) = s_k(x)$.

Lema 1.2.1. *Sigui k l'índex de l'últim element r_k no nul de la funció recursiva; es compleix que $gr(t_k) < 8$.*

Demostració: Per demostrar-ho cal tenir en compte algunes observacions que es compleixen entre aquests polinomis: $gr(q_i) = gr(r_{i-1}) - gr(r_i)$, ja que q_i està definit com el quocient entre r_{i-1} i r_i. També s'ha de tenir en compte que $gr(t_0) = gr(0) = -\infty$, $gr(t_1) = gr(1) = 0$. Finalment, també se satisfà que $gr(r_{i-1}) > gr(r_i)$, ja que r_i està definit com el residu entre r_{i-2} i r_{i-1}.

Anem a veure que es compleix que $gr(t_i) \leq 8 - gr(r_{i-1})$. Per inducció; si $i = 2$, $t_2 = q_1 t_1$ i, per tant, $gr(t_2) = gr(q_1) + gr(t_1) = gr(r_0) - gr(r_1)$; com que $r_0 = m(x)$, aleshores $gr(t_2) = 8 - gr(r_1)$.

Si suposem que es compleix $\forall i \leq n-1$, aleshores se satisfà per a n. $t_n = t_{n-2}+q_{n-1}t_{n-1}$; aleshores, $gr(t_n) \leq max(gr(t_{n-2}), gr(q_{n-1})+gr(t_{n-1})) = max(8-gr(r_{n-3}), 8-gr(r_{n-2})+gr(r_{n-2}) - gr(r_{n-1})) = max(8 - gr(r_{n-3}), 8 - gr(r_{n-1})) = 8 - gr(r_{n-1})$. Per tant, queda

demostrat que es compleix que $gr(t_i) \leq 8 - gr(r_{i-1})$ i, en conseqüència, que $gr(t_i) < 8$, ja que la funció recursiva acaba quan $r_{i+1} = 0$; per tant, r_i és o 1 o de grau superior a 0. En qualsevol dels dos casos r_{i-1} ha de ser de grau més gran que 0; per tant, $gr(t_k) \leq 8 - gr(r_{k-1}) < 8$. □

Tornant a l'algoritme ampliat d'Euclides, la igualtat trobada esdevé la identitat de Bezout $1 = \hat{a}(x) \cdot b(x) + c(x) \cdot m(x)$. Es pot passar aquesta igualtat al cos finit $\mathbb{K}$ utilitzant que els elements de $\mathbb{F}_2$ de grau menor que 8 són representants de les classes d'equivalència de $\mathbb{K}$; per tant, $[1] = [\hat{a}(x)] \cdot [b(x)] + [c(x)] \cdot [m(x)]$, d'on resulta que $1 = a(x) \cdot b(x) + c(x) \cdot 0$. Per tant, s'obté que $1 = a(x) \cdot b(x)$, cosa que implica que $a^{-1}(x) = b(x)$. Com es volia veure, aquest algoritme permet trobar l'element invers sempre que $a(x) \neq 0$.

L'algoritme binari d'exponenciació aprofita les propietats de les potències per tal de reduir el nombre de multiplicacions a l'hora de realitzar potències d'un determinat element.

Per exemple, si volem expressar la 254a potència d'un element $a \in \mathbb{K}$, la forma naïve de calcular-ho és realitzant 253 multiplicacions. Però si es té en compte el fet que es pot expressar $254 = 2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 0 \cdot 2^0$ aleshores per les propietats de les potencies es pot expressar a^{254} com:

$$a^{254} = a^{2^7} \cdot a^{2^6} \cdot a^{2^5} \cdot a^{2^4} \cdot a^{2^3} \cdot a^{2^2} \cdot a^{2^1} = \left(\left(\left(\left(\left(\left(\left(a^2\right) a\right)^2 a\right)^2 a\right)^2 a\right)^2 a\right)^2 a\right)^2 . \quad (1.2.4)$$

Per tant, s'acaba de reduir el nombre d'operacions que s'han de fer de 253 a 13 (7 quadrats i 6 multiplicacions mòdul $m(x)$), un nombre considerablement més petit.

Aquestes són dues maneres que es pot calcular l'element invers de forma eficient.

Per acabar aquesta secció, convé introduir una notació que es farà servir a la part final del treball, l'automorfisme de Frobenius de $\mathbb{K}$ sobre $\mathbb{F}_2$, que es defineix com

$$\begin{array}{rccc} \phi: & \mathbb{K} & \longrightarrow & \mathbb{K} \\ & a & \longmapsto & a^2. \end{array} \quad (1.2.5)$$

1.3 Codificació

En l'inici del capítol (cf. la secció 1.1), s'ha comentat breument que cal, a partir d'una comunicació que es vol transmetre, crear una successió de missatges plans. Resulta que aquest procés no és tant trivial com es pot pensar, ja que l'AES admet únicament missatges plans de 128 bits. Per tant, en un principi només es podrien xifrar les comunicacions la llargada de les quals fos múltiple de 128 bits. A la pràctica, és possible xifrar comunicacions de qualsevol longitud.

La tècnica utilitzada per aconseguir xifrar comunicacions de qualsevol longitud s'anomena *farcit* i com el seu nom indica consisteix a afegir bits a la comunicació fins a tenir una longitud igual a un múltiple de 128 bits, per així poder crear una successió de missatges plans i poder-los xifrar. Cal destacar que si el farcit no es fa correctament pot provocar la disminució de la seguretat del criptosistema.

És per això que existeixen certs farcits estàndard que es poden utilitzar per preparar la comunicació. Una part vital del farciment és que el dispositiu que s'encarrega de desxifrar els missatges sigui capaç de treure el farcit de forma que no hi hagi ambigüitats.

Seguidament, es presentaran cinc farcits comuns; en la majoria, el missatge original és una successió de bytes; en cas contrari, s'especificarà durant l'explicació.

PKCS5: Consisteix a farcir la comunicació amb una successió d'entre 1 i 16 bytes per fer la longitud total de la comunicació igual a un múltiple de 16 bytes. El valor de cada byte afegit pel farcit és igual a la quantitat de bytes afegits en la comunicació. Per exemple, si tenim una comunicació de 45 bytes, aleshores se l'hi hauran d'afegir 3 bytes de valor {03}.

Farcit d'uns i zeros: Consisteix a afegir a la comunicació una successió d'entre 0 i 15 bytes; si aquesta és multiple exacte de 16 bytes, no s'hi afegeix res. El valor del primer byte afegit és {80}, mentre que la resta de bytes s'omplen amb successions de {00}. Si es torna a l'exemple anterior, els tres bytes que s'afegirien a la successió serien: {80} {00} {00}. És important destacar que aquest farciment es pot utilitzar ja que en ASCII no existeix cap caràcter que correspongui a {80}. En cas contrari, s'hauria de fer algunes modificacions perquè no hi haguessin indeterminacions. Aquest farciment és el que s'utilitza en les targetes de crèdit, per exemple.

Zero-longitud: Es tracta de farcir la comunicació de 0 fins a l'últim byte, el valor del qual coincideix amb la longitud del farcit del missatge. Si s'utilitza aquest mètode, es farceix la comunicació encara que la longitud d'aquesta sigui múltiple de 16 bytes. Amb l'exemple anterior, el farcit de la comunicació seria: {00} {00} {03}.

Farcir amb zeros: Es farceixen tots els bytes que falten perquè la comunicació tingui una longitud múltiple a 128 bits amb bytes de valor {00}. El principal problema d'aquest farcit és que no es possible recuperar el missatge original. Si es xifren missatges plans escrits en ASCII, això pot ser intrascendent, ja que el 00 en ASCII representa el caràcter nul, així que sempre que no sigui important el nombre de nuls que tingui el missatge pla aquesta codificació pot ser utilitzada. Tot i així, pot esdevenir un problema si s'està xifrant un missatge pla de dades binàries, com per exemple un document .exe, ja que no es té manera de recuperar el missatge pla de forma exacta.

Si es vol farcir un text ASCII, també és possible farcir-lo amb espais, de manera que els bytes que s'afegiran a la comunicació seran de la forma {02}.

Farcit aleatori: S'afegeixen a la comunicació una successió entre 1 i 16 bytes de valor aleatori excepte l'últim que té la informació amb la quantitat de bytes del farcit.

El *NIST* aconsella utilitzar el *Farcit d'uns i zeros* per codificar la comunicació, tot i que tant aquest com el *PKCS5* són vàlids i els més recomanats per a la codificació. Cal deixar clar que aquestes recomanacions no impliquen que siguin els més segurs, només que són els estàndards; és a dir, els que el NIST considera que més es fan servir i són fins a cert punt segurs.

El més segur, com és lògic, és el facit aleatori; si a més, enlloc de fer que l'últim byte sigui la llargada del farcit, el penséssim en forma de 8 bits, com que només es necessiten 4 bits per representar aquesta longitud, els 4 bits involucrats en la llargada podrien ser el primer, el segon, el quart i el vuitè, o qualssevol altres quatre prefixats, i així es reduiria al màxim la capacitat de fer un atac amb una part del text pla conegut. Tot i així, aquest mètode requereix que els dos dispositius es posin d'acord amb la posició de cada bit que dóna la llargada.

Capítol 2

Xifratge

El criptosistema AES pot ser considerat com a tres criptosistemes diferents, AES-128, AES-192 i AES-256, que comparteixen les mateixes funcions, o similars; per tant, la longitud del missatge pla ha de ser la mateixa (128 bits). L'única cosa que canvia és que varia la longitud de la clau, que és de 128, 192 i 256 bits, respectivament. A partir d'aquestes claus es generen noves successions anomenades claus de ronda; la manera de generar aquestes claus també varia entre els tres criptosistemes.

Una comunicació qualsevol no pot ser xifrada per l'AES; cal fer una codificació prèvia a la comunicació que es vol transmetre per després poder-la xifrar utilitzant el criptosistema. Primer de tot, es transforma, si cal, la comunicació escrita, en un alfabet qualsevol, a un codi binari, utilitzant qualsevol codificació de caràcters com per exemple ASCII i les seves extensions o taules de codis. Un cop realitzada la transformació s'ha de dividir la comunicació en successions de 128 bits, que com s'ha esmentat en el capitol anterior (cf. la secció 1.3) aquestes successions formen el conjunt de missatges plans de l'AES, que també s'anomenen estats.

Les successions de 128 bits es poden dividir cada 32 per formar *paraules*, tot i que el més usual és dividir-los en 16 agrupacions de 8 bits anomenades *bytes*.

El bytes s'interpreten, en aquest cas, com a elements del cos finit $\mathbb{F}_{2^8} = \mathbb{K}$ (cf. la secció 1.2); i, per tant, es pot pensar l'estat com un conjunt de 16 elements de $\mathbb{K}$; és a dir, com un vector del $\mathbb{K}$-espai vectorial de dimensió 16, $\mathbb{K}^{16}$. Es definieix l'espai vectorial $E = \mathbb{K}^{16} = T^4 = U^4$, on $U = \mathbb{K}^4$ i $T = \mathbb{K}^4$; tot seguit s'indicarà la diferència entre aquests dos espais vectorials i com s'utilitzen.

Es pot escriure l'estat com un vector $\vec{s} \in \mathbb{K}^{16}$ o bé com un element de l'espai vectorial format per matrius 4×4 de coficients en el cos $\mathbb{K}$. Per tant es pot escriure l'estat com a mínim de tres maneres diferents: com a matriu,

$$\begin{pmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{pmatrix} \in M(4, \mathbb{K}); \tag{2.0.1}$$

com a vector ordenat per files,

$$(s_{0,0}, s_{0,1}, s_{0,2}, s_{0,3}, s_{1,0}, s_{1,1}, s_{1,2}, s_{1,3}, s_{2,0}, s_{2,1}, s_{2,2}, s_{2,3}, s_{3,0}, s_{3,1}, s_{3,2}, s_{3,3})^t \in T^4,$$

que més endavant serà útil expressar com a $(t_0, t_1, t_2, t_3)^t$ on $t_i = (s_{i,0}, s_{i,1}, s_{i,2}, s_{i,3})^t \in T$; o en forma de vector ordenat per comlumnes

$$(s_{0,0}, s_{1,0}, s_{2,0}, s_{3,0}, s_{0,1}, s_{1,1}, s_{2,1}, s_{3,1}, s_{0,2}, s_{1,2}, s_{2,2}, s_{3,2}, s_{0,3}, s_{1,3}, s_{2,3}, s_{3,3})^t \in U^4;$$

en aquest cas, també és pertinent definir una subestructura per expresar l'estat, $(u_0, u_1, u_2, u_3)^t$ on $u_i = (s_{0,i}, s_{1,i}, s_{2,i}, s_{3,i})^t \in U$, que representen les columnes.

És fàcil veure que per relacionar l'estat ordenat per files i per columnes només cal una permutació, que expressem en forma matricial,

$$\begin{pmatrix} s_{0,0} \\ s_{1,0} \\ s_{2,0} \\ s_{3,0} \\ s_{0,1} \\ s_{1,1} \\ s_{2,1} \\ s_{3,1} \\ s_{0,2} \\ s_{1,2} \\ s_{2,2} \\ s_{3,2} \\ s_{0,3} \\ s_{1,3} \\ s_{2,3} \\ s_{3,3} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} s_{0,0} \\ s_{0,1} \\ s_{0,2} \\ s_{0,3} \\ s_{1,0} \\ s_{1,1} \\ s_{1,2} \\ s_{1,3} \\ s_{2,0} \\ s_{2,1} \\ s_{2,2} \\ s_{2,3} \\ s_{3,0} \\ s_{3,1} \\ s_{3,2} \\ s_{3,3} \end{pmatrix}. \tag{2.0.2}$$

Clarament, la matriu esmentada en 2.0.2, posem P, representa un isomorfisme entre els espais vectorials, $P : T^4 \longrightarrow U^4$.

S'ha de recordar que cada element $s_{i,j} \in \mathbb{K}$, i per això es pot respresentar com un polinomi de grau menor que 8 i de coeficients en $\mathbb{F}_2$; o sigui,

$$s_{i,j} = a_{i,j,0} + a_{i,j,1}x + a_{i,j,2}x^2 + a_{i,j,3}x^3 + a_{i,j,4}x^4 + a_{i,j,5}x^5 + a_{i,j,6}x^6 + a_{i,j,7}x^7, \tag{2.0.3}$$

amb $a_{i,j,k} \in \mathbb{F}_2$, on $0 \leq i, j \leq 3$ i $0 \leq k \leq 7$.

Tot seguit, es presentaran les funcions que més s'utilitzen per xifrar el missatge, així com també la forma amb la qual es generen les claus de ronda que requereix el criptosistema.

2.1 SubBytes

La funció *SubBytes* és una transformació que actua byte a byte de forma independent. Aquesta operació consta de dos passos. El primer consisteix a invertir el byte de sortida (que com que pertany a un cos finit té invers), aquest procés és bàsic per afegir confusió al sistema i evitar que el criptosistema pugui ser considerat afí.

La segona part de la transformació es pot descriure com la següent suma, amb $0 \leq i < 8$:

$$a'_i = a_i + a_{(i+4)mod8} + a_{(i+5)mod8} + a_{(i+6)mod8} + a_{(i+7)mod8} + b_i, \tag{2.1.1}$$

on $b = x^6 + x^5 + x + 1 = (1, 1, 0, 0, 0, 1, 1, 0) \in \mathbb{K}$ que també es pot expressar en notació hexadecimal com $\{63\}$.

Si es tracta el cos finit $\mathbb{K}$ com un espai vectorial de dimensió 8 usant la base esmentada anteriorment, es pot escriure l'equació 2.1.1 en forma matricial:

$$\begin{pmatrix} a'_0 \\ a'_1 \\ a'_2 \\ a'_3 \\ a'_4 \\ a'_5 \\ a'_6 \\ a'_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}. \tag{2.1.2}$$

Per tant, la segona part de la transformació *SubBytes* és una transformació afí.

En resum, sigui $M \in GL(8, \mathbb{F}_2)$, on $GL(8, \mathbb{K})$ representa el *grup lineal general* sobre el cos $\mathbb{F}_2$; és a dir el grup de matrius quadrades i invertibles de rang 8 de coeficients en $\mathbb{F}_2$. Si M representa la matriu associada a la transformació afí i $b = \{63\}$ el vector associat a aquesta, es pot escriure l'operació total com:

$$\begin{aligned} SubBytes : \mathbb{K} &\longrightarrow \mathbb{K} \\ a &\longmapsto M \cdot inv(a) + b; \end{aligned} \tag{2.1.3}$$

equivalentment,

$$\begin{aligned} SubBytes : \mathbb{K} &\longrightarrow \mathbb{K} \\ a &\longmapsto M \cdot a^{254} + b. \end{aligned} \tag{2.1.4}$$

A l'hora de la veritat, si el processador té prou memòria, l'operació es fa amb l'ús de la taula 2.1. Si es vol saber quin és el valor resultant d'aplicar *SubBytes* sobre un element $\{xy\}$ es va a buscar l'element de la fila x i columna y. Per exemple, el resultat de l'operació *SubBytes* a $\{53\}$ és l'element de la fila 5 i la columna 3 que correspon al byte $\{ED\}$. En cas que no es tingués suficient memoria per emmagatzemar tota la taula al processador, s'aplica tota la transformació per cada byte, és una mica més lent.

x/y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	AC	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6B	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9D
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Taula 2.1: Sbox. Taula de substitució de l'operació *SubBytes*

Notem que la raó per la qual en l'AES no es fa servir la tabulació dels inversos és que si es té suficient espai per guardar una taula d'inversos aleshores té prou espai per guardar la *Sbox*, ja que la taula d'inversos i la *Sbox* ocupen exactament el mateix.

Si s'aplica *SubBytes* a tot l'estat en lloc de fer-ho a un únic byte, s'obté la següent expressió que s'anomena *SB*.

Proposició 2.1.1. *La funció resultant d'aplicar SubBytes a tot l'estat és la següent,*

$$\begin{array}{cccc} SB: & U^4 & \longrightarrow & U^4 \\ & \begin{pmatrix} s_{0,0} \\ s_{1,0} \\ s_{2,0} \\ \vdots \\ s_{3,3} \end{pmatrix} & \longmapsto & \begin{pmatrix} M & 0 & 0 & \dots & 0 \\ 0 & M & 0 & \dots & 0 \\ 0 & 0 & M & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & M \end{pmatrix} \begin{pmatrix} s_{0,0}^{254} \\ s_{1,0}^{254} \\ s_{2,0}^{254} \\ \vdots \\ s_{3,3}^{254} \end{pmatrix} + \begin{pmatrix} 63 \\ 63 \\ 63 \\ \vdots \\ 63 \end{pmatrix}, \end{array} \tag{2.1.5}$$

on M representa la matriu esmentada a 2.1.2.

2.2 ShiftRows

La transformació *ShiftRows* consisteix bàsicament a aplicar un conjunt de permutacions sobre l'estat inicial. En concret, a la primera fila de l'estat no se l'hi aplica cap transformació; els elements de la segona fila passen a la columna següent, és a dir, $s_{1,j} = s_{1,(j+1)mod4}$; els de la tercera fila salten dues columnes cap a la dreta, $s_{1,j} = s_{1,(j+2)mod4}$; i finalment els de l'última columna reben la següent transformació $s_{1,j} = s_{1,(j+3)mod4}$. Es podria resumir la transformció final de coeficients com $s_{i,j} = s_{i,(j+i)mod4}$ on $s_{i,j}$ són els coeficients de l'estat inicial en forma matricial amb $0 \leq i, j \leq 3$. La transformació de la matriu associada seria la següent:

$$\begin{pmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{pmatrix} \longmapsto \begin{pmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,1} & s_{1,2} & s_{1,3} & s_{1,0} \\ s_{2,2} & s_{2,3} & s_{2,0} & s_{2,1} \\ s_{3,3} & s_{3,0} & s_{3,1} & s_{3,2} \end{pmatrix}. \tag{2.2.1}$$

Si s'agafa cada fila com un vector $(s_{i,0}, s_{i,1}, s_{i,2}, s_{i,3})^t = t_i$, una manera senzilla de tractar aquesta transformació de forma matemàtica és utilitzant les permutacions. La permutació que trasllada quatre coeficients a la columna de la seva dreta, quan s'aplica a un vector t_i, s'expressa mitjançant la matriu

$$\mathfrak{R} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}. \tag{2.2.2}$$

Aleshores, per les propietats de les permutacions, les permutacions que mouen els coeficients a les dues i tres columnes següents seran $\mathfrak{R}^2$ i $\mathfrak{R}^3$ respectivament. Per tant, aquesta operació es pot expressar de la forma següent, si utilitzem la respresentació de l'estat ordenat per files:

$$\begin{pmatrix} t'_0 \\ t'_1 \\ t'_2 \\ t'_3 \end{pmatrix} = \begin{pmatrix} I & 0 & 0 & 0 \\ 0 & \mathfrak{R} & 0 & 0 \\ 0 & 0 & \mathfrak{R}^2 & 0 \\ 0 & 0 & 0 & \mathfrak{R}^3 \end{pmatrix} \begin{pmatrix} t_0 \\ t_1 \\ t_2 \\ t_3 \end{pmatrix}, \tag{2.2.3}$$

on I és la *matriu identitat* 4×4.

En resum, si anomenem L a la matriu 16×16 de l'equació 2.2.3, aleshores podem definir l'operació *ShiftRows*, que a partir d'ara s'anomenarà *SR*, de la forma següent:

$$\begin{array}{rccc} SR: & T^4 & \longrightarrow & T^4 \\ & \begin{pmatrix} t_0 \\ t_1 \\ t_2 \\ t_3 \end{pmatrix} & \longmapsto & L \cdot \begin{pmatrix} t_0 \\ t_1 \\ t_2 \\ t_3 \end{pmatrix} . \end{array} \tag{2.2.4}$$

Lema 2.2.1. *L'aplicació SB i SR commuten entre elles.*

Demostració: Com que l'aplicació *SubBytes* actua sobre els bytes de forma independent als altres bytes i a la posició en què es troba, mentre que la transformació *ShiftRows* altera l'ordre dels bytes però no els fa interactuar entre ells, és clar que les dues operacions *SubBytes* i *ShiftRows* commuten entre elles. □

Proposició 2.2.2. *La funció resultant d'aplicar $ShiftRows$ a tot l'estat és la següent,*

$$\begin{array}{rccc} SR: & T^4 & \longrightarrow & T^4 \\ & \begin{pmatrix} t_0 \\ t_1 \\ t_2 \\ t_3 \end{pmatrix} & \longmapsto & \begin{pmatrix} I & 0 & 0 & 0 \\ 0 & \mathfrak{R} & 0 & 0 \\ 0 & 0 & \mathfrak{R}^2 & 0 \\ 0 & 0 & 0 & \mathfrak{R}^3 \end{pmatrix} \cdot \begin{pmatrix} t_0 \\ t_1 \\ t_2 \\ t_3 \end{pmatrix} , \end{array} \tag{2.2.5}$$

amb $\mathfrak{R}$ la matriu esmentada en 2.1.2.

2.3 MixColumns

Aquesta operació és fa sobre la $\mathbb{K}$-àlgebra definida com $\mathbb{A} = \mathbb{K}[y]/\langle y^4 + 1\rangle$. Aquest anell és commutatiu però no és un cos, ja que y^4+1 no es irreductible; de fet, $y^4+1 = (y+1)^4$.

En aquest cas, es considera un estat ordenat per columnes com un element de $\mathbb{A}^4$ a partir de les immersions

$$\begin{array}{c} i_j : \mathbb{K}^4 \longrightarrow \mathbb{A} \\ u_j = (s_{0,j}, s_{1,j}, s_{2,j}, s_{3,j})^t \longmapsto s_{0,j} + s_{1,j}y + s_{2,j}y^2 + s_{3,j}y^3, \end{array} \quad 0 \leq j \leq 3. \tag{2.3.1}$$

La inclusió 2.3.1 es duu a terme per a les quatre columnes; és a dir, tenim 4 aplicacions, cada una amb una j fixada. A més, convé recordar que els coeficients $s_{i,j} \in \mathbb{K}$.

L'aplicació *MixColumns* consisteix en multiplicar cada un dels elements esmentats per un elements fixat de $\mathbb{A}$, que és el següent:

$$a(y) = \{03\}y^3 + \{01\}y^2 + \{01\}y + \{02\} = (x+1)y^3 + y^2 + y + x. \tag{2.3.2}$$

Així doncs, l'operació *MixColumns* es pot representar com una multiplicació entre elements de l'anell $\mathbb{A}$:

$$\begin{array}{c} MixColumns : \mathbb{A} \longrightarrow \mathbb{A} \\ s_j(y) \longmapsto a(y) \cdot s_j(y), \end{array} \tag{2.3.3}$$

on $s_j(y)$ representa l'element obtingut de la inclusio 2.3.1. L'element resultant d'aquesta operació $c(y)$ serà el següent:

$$c(y) = c_6y^6 + c_5y^5 + c_4y^4 + c_3y^3 + c_2y^2 + c_1y + c_0, \tag{2.3.4}$$

amb $c_6 = s_{3,j} \cdot \{03\}$, $c_5 = s_{2,j} \cdot \{03\} + s_{3,j}$, $c_4 = \{03\} \cdot s_{1,j} + s_{2,j} + s_{1,j}$, $c_3 = \{03\} \cdot s_{0,j} + s_{1,j} + s_{2,j} + s_{3,j} \cdot \{02\}$, $c_2 = s_{0,j} + s_{1,j} + s_{2,j} \cdot \{02\}$, $c_1 = s_{0,j} + s_{1,j} \cdot \{02\}$ i $c_0 = s_{0,j} \cdot \{02\}$. Ara si es recorda que aquest anell té la relació quocient $y^4 + 1 = 0$, per a qualsevol element $P(y)$ se satisfà que $y^i = y^{i \ (mod 4)}$ i per tant, l'element $c(y)$ serà:

$$c(y) = c'_3y^3 + c'_2y^2 + c'_1y + c'_0, \tag{2.3.5}$$

$$\begin{aligned} c'_3 &= \{03\} \cdot s_{0,j} + s_{1,j} + s_{2,j} + \{02\} \cdot s_{3,j}, \\ c'_2 &= s_{0,j} + s_{1,j} + \{02\} \cdot s_{2,j} + \{03\} \cdot s_{3,j}, \\ c'_1 &= s_{0,j} + \{02\} \cdot s_{1,j} + \{03\} \cdot s_{2,j} + s_{3,j}, \\ c'_0 &= \{02\} \cdot s_{0,j} + \{03\} \cdot s_{1,j} + s_{2,j} + s_{3,j}. \end{aligned} \tag{2.3.6}$$

Un cop trobats aquests coeficients, s'ha de tornar l'estat a la seva forma natural a través de:

$$\begin{array}{rccccccc} MixColumns: & \mathbb{K}^4 & \longrightarrow & \mathbb{A} & \longrightarrow & \mathbb{A} & \longrightarrow & \mathbb{K}^4 \\ & u_j & \longmapsto & s_j(y) & \longmapsto & a(y) \cdot s_j(y) & \longmapsto & u'_j. \end{array} \tag{2.3.7}$$

Un cop analitzada l'equació 2.3.6, s'observa que el resultat de l'operació *MixColumns* es pot expressar a través d'una aplicació lineal entre elements de $\mathbb{K}^4$ de la forma següent:

$$u'_j = \begin{pmatrix} s'_{0,j} \\ s'_{1,j} \\ s'_{2,j} \\ s'_{3,j} \end{pmatrix} = \begin{pmatrix} \{02\} & \{03\} & \{01\} & \{01\} \\ \{01\} & \{02\} & \{03\} & \{01\} \\ \{01\} & \{01\} & \{02\} & \{03\} \\ \{03\} & \{01\} & \{01\} & \{02\} \end{pmatrix} \begin{pmatrix} s_{0,j} \\ s_{1,j} \\ s_{2,j} \\ s_{3,j} \end{pmatrix} = S \cdot u_j. \tag{2.3.8}$$

Aquest procés s'ha d'aplicar a les quatre columnes que conformen estat. Per tant si anomenem S la matriu 4×4 representada a l'equació 2.3.8, l'operació *MixColumns* aplicada a tot l'estat s'anomena MC i és de la forma

$$\begin{array}{rccc} MC: & U^4 & \longrightarrow & U^4 \\ & \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \end{pmatrix} & \longmapsto & \begin{pmatrix} S & 0 & 0 & 0 \\ 0 & S & 0 & 0 \\ 0 & 0 & S & 0 \\ 0 & 0 & 0 & S \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \end{pmatrix}. \end{array} \tag{2.3.9}$$

És important destacar que aquesta aplicació és bàsica tant per a la difusió com per a la confusió. Tot i que en principi es podria pensar que és únicament rellevant per a la difusió del sistema, ja que només actua sobre l'estat que està format per modificacions del text pla, però el funcionament per rondes de l'AES fa que al final acabi involucrant també el que s'anomenaran claus de ronda i, per tant, que també aporti confusió.

Proposició 2.3.1. *La funció resultant d'aplicar MixColumns a tot l'estat és la següent,*

$$\begin{array}{rccc} MC: & U^4 & \longrightarrow & U^4 \\ & \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \end{pmatrix} & \longmapsto & \begin{pmatrix} S & 0 & 0 & 0 \\ 0 & S & 0 & 0 \\ 0 & 0 & S & 0 \\ 0 & 0 & 0 & S \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \end{pmatrix}, \end{array} \tag{2.3.10}$$

on S és la matriu definida en l'equació 2.3.8.

2.4 Generació de la Clau de Ronda

A partir de la clau consensuada entre els dispositius (de 128, 192 o 256 bits), com que els estats tenen una longitud de 128 bits i l'estat i la clau se sumen, cal construir successions de 128 bits. A més, l'algoritme de xifratge és iteratiu de manera que cal construir tantes successions de 128 bits com rondes tingui l'algoritme; aquestes successions s'anomenen claus de ronda.

En aquesta secció, explicarà com expandir la clau per al cas més complex de 256 bits, que esta estipulat al [FIPS 2001]; però, en cas que algun dels altres mètodes tingués característiques específiques també es faran les distincions. En el cas de l'AES-256, es repeteix l'aplicació *AddRoundKey* 15 cops; per tant, s'hauran de generar 15 claus de ronda.

Partim d'una *Clau* que és coneguda tant per l'emissor com pel receptor del missatge. Aquesta clau es pot dividir en 8 *paraules* (conjunt de 32-bits):

$$(w_0, w_1, w_2, w_3, w_4, w_5, w_6, w_7)^t. \tag{2.4.1}$$

Com que l'AES-256 té 15 rondes són necessaries 15 claus de ronda. Aleshores el vector clau de ronda serà $cr = (w_0, w_1, \ldots, w_{59})^t$ on cada w_i representa una paraula de la clau de ronda i se satisfà que $0 \leq i \leq 59$.

Hi ha dues noves transformacions que s'utilitzen en el procés de crear les claus de ronda. Per una banda, com que una paraula es pot veure com un conjunt de 4 bytes, $w_i = (a_{i,0}, a_{i,1}, a_{i,2}, a_{i,3})^t$, *RotWord* aplica una permutació a aquesta paraula de manera que $(a_{i,0}, a_{i,1}, a_{i,2}, a_{i,3})^t \longmapsto (a_{i,1}, a_{i,2}, a_{i,3}, a_{i,0})^t$. Per tant, es pot expressar la transformació com:

$$\begin{array}{rccc} RotWord: & \mathbb{K}^4 & \longrightarrow & \mathbb{K}^4 \\ & \begin{pmatrix} a_{i,0} \\ a_{i,1} \\ a_{i,2} \\ a_{i,3} \end{pmatrix} & \longmapsto & \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} a_{i,0} \\ a_{i,1} \\ a_{i,2} \\ a_{i,3} \end{pmatrix}. \end{array} \tag{2.4.2}$$

es pot apreciar que la matriu utilitzada és la mateixa que la matriu $\mathfrak{R}$ utilitzada en la secció 2.2.

Per altra banda, *SubWord* és l'equivalent a aplicar la transformació *SubBytes* a cada byte de la paraula, és a dir;

$$\begin{array}{rccc} SubWord: & \mathbb{K}^4 & \longrightarrow & \mathbb{K}^4 \\ & (a_{i,0}, a_{i,1}, a_{i,2}, a_{i,3}) & \longmapsto & (SB(a_{i,0}), SB(a_{i,1}), SB(a_{i,2}), SB(a_{i,3})). \end{array} \tag{2.4.3}$$

Finalment, *Rcon[i]* genera una paraula tal que $Rcon[i] = (x^{i-1}, \{00\}, \{00\}, \{00\})$. Cal tenir en compte que $x^{i-1} \in \mathbb{K}\ \forall i > 0$, per exemple x s'expressa en notació hexadecimal com $\{02\}$.

Un cop definides les noves funcions, es pot explicar com es generen les claus de ronda. Per tal que resulti més fàcil, es pot fer un seguiment de tot el que es va explicant a través de l'equació 2.4.7.

Es treballa amb les claus com a conjunt de paraules, ja que les funcions usades actuen sobre successions de 4 bytes. Durant tot el procès serà necessari utilitzar una paraula

auxiliar anomenada *pa*. Les anomenades claus de ronda, es posaran totes en un mateix vector, *ClausRonda*, que tindrà una longitud de $4 \cdot 15 = 60$ paraules.

Les primeres 8 paraules correspondran a la clau decidida pels dos dispositius abans d'establir la comunicació. Seguidament, i mentre el contador i, que s'inicialitza a 8, no sigui igual a 60, es repetiran les següents accions i comprovacions:
S'utilitza la paraula immediatament anterior com a paraula auxiliar $pa = w_{i-1}$.
Si $i(mod 8) = 0$, amb $8 \leq i < 60$, aleshores se li aplica a *pa* la següent transformació:

$$pa = SubWord(RotWord(pa)) + Rcon\left(\frac{i}{8}\right); \tag{2.4.4}$$

en canvi, si $i(mod 8) = 4$, aleshores la transformació és la següent:

$$pa = SubWord(pa). \tag{2.4.5}$$

Finalment, en qualsevol dels casos, $w_i = w_{i-8} + pa$.

Fent aquest procés s'acaba obtenint una clau de ronda prou llarga com per aplicar-la 15 vegades, que és precisament els cops que la necessita el sistema.

En resum, es pot definir l'aplicació *GenereacioClausRonda* com:

$$\begin{array}{rccc} GeneracioClausRonda: & (\mathbb{K}^{16})^2 & \longrightarrow & (\mathbb{K}^{16})^{15} \\ & c & \longmapsto & GeneracioClausRonda(c), \end{array} \tag{2.4.6}$$

on c representa la clau del sistema de 256 bits. I la funció *GeneracioClausRonda* és la següent:

$$\begin{cases} w_i = c_i, & \\ & si \quad 0 \leq i \leq 7, \\ w_i = SubWord(RotWord(w_{i-1})) + & (x^{\frac{i}{8}}, \{00\}, \{00\}, \{00\}) + w_{i-8}, \\ & si \quad 8 \leq i \leq 59 \ \wedge \ i(mod 8) = 0, \\ w_i = SubWord(w_{i-1}) + w_{i-8}, & \\ & si \quad 8 \leq i \leq 59 \ \wedge \ i(mod 8) = 4, \\ w_i = w_{i-1} + w_{i-8}, & \\ & altrament, \end{cases} \tag{2.4.7}$$

on w és la successió de paraules de la clau de ronda, c_i són les 8 paraules de què està formada la clau del sistema, la resta són les funcions definides prèviament i se satisfà que $0 \leq i \leq 59$.

Cal recodar que aquestes funcions estan descrites per a paraules, no per a bytes. Seguidament, per tal d'unificar aquesta secció amb el llenguatge utilitzat durant tot el treball, s'escriurà la funció *Generació de claus de ronda* per a bytes:

$$\begin{array}{rccc} GeneracioClausRonda: & E^2 & \longrightarrow & E^{15} \\ & c & \longmapsto & GC(c), \end{array} \tag{2.4.8}$$

$c = (c_0, c_1, \ldots, c_{31})$ que representa la clau de 32 bytes del sistema; en aquesta ocasió, enlloc d'anomenar cada part de la clau de ronda w_i, numenclatura escollida per designar a la clau de ronda separada per paraules, s'anomenaran a_j, amb $0 \leq j \leq 239$, als bytes

individuals de la clau de ronda. $GC(c)$ consisteix en

$$\begin{cases} a_j = c_j, & si \quad 0 \leq j \leq 31, \\ a_j = SB(a_{j-3}) + a_{j-32} + x^{\frac{j}{32}}, & si \quad 32 \leq j \leq 239 \ \wedge \ j(mod32) = 0, \\ a_j = SB(a_{j-3}) + a_{j-32}, & si \quad 32 \leq j \leq 239 \ \wedge \ j(mod32) = 1, \\ a_j = SB(a_{j-3}) + a_{j-32}, & si \quad 32 \leq j \leq 239 \ \wedge \ j(mod32) = 2, \\ a_j = SB(a_{j-7}) + a_{j-32}, & si \quad 32 \leq j \leq 239 \ \wedge \ j(mod32) = 3, \\ a_j = SB(a_{j-4}) + a_{j-32}, & si \quad 32 \leq j \leq 239 \ \wedge \ j(mod32) = 16, \\ a_j = SB(a_{j-4}) + a_{j-32}, & si \quad 32 \leq j \leq 239 \ \wedge \ j(mod32) = 17, \\ a_j = SB(a_{j-4}) + a_{i-32}, & si \quad 32 \leq j \leq 239 \ \wedge \ j(mod32) = 18, \\ a_j = SB(a_{j-4}) + a_{j-32}, & si \quad 32 \leq j \leq 239 \ \wedge \ j(mod32) = 19, \\ a_j = a_{j-4} + a_{j-32}, & altrament, \end{cases} \tag{2.4.9}$$

que com es pot veure és la traducció literal de la funció *GeneracioClausRonda* en bytes en lloc de paraules.

Proposició 2.4.1. *La funció generació de claus de ronda de l'AES-256 està descrita a 2.4.7, en paraules, i a 2.4.9, en funció dels bytes de la clau.*

Cal destacar que per a les funcions AES-128 i AES-192, la llargada de les claus és de 16 i 24 bytes respectivament i les funcions canvien una mica ja que, en el primer cas, el sistema només necessita 11 claus de ronda i, en el segon, se'n necessiten 13. Per a les altres versions de l'AES els condicionals varien lleugerament, així com la funció *Rcon*: en l'AES-128 es canvia el condicional per si $i(mod4) = 0$; això fa que la funció $Rcon[i/4]$ també varïi, per a l'AES-192 passa el mateix, però amb $i(mod6) = 0$ i la funció s'avalua com $Rcon[i/6]$. El pas on només s'aplica *SubWord* només té lloc en l'AES-256, en cap dels altres dos sistemes s'utilitza. Seguidament, s'expresaran la funció *GeneracióClausRonda* per a l'AES-128 i l'AES-192, en paraules.

$$\begin{array}{rccc} GeneracioClausRonda_{128}: & \mathbb{K}^{16} & \longrightarrow & (\mathbb{K}^{16})^{11} \\ & c & \longmapsto & GeneracioClausRonda_{128}(c), \end{array} \tag{2.4.10}$$

suposant que $w_i \in \mathbb{K}^4$ aleshores $GeneracioClausRonda_{128}$,

$$\begin{cases} w_i = c_i, & si \quad 0 \leq i < 3, \\ w_i = SubWord(RotWord(w_{i-1})) + (x^{\frac{i}{4}}, \{00\}, \{00\}, \{00\}) + w_{i-4}, & si \quad 4 \leq i \leq 43 \ \wedge \ i(mod4) = 0, \\ w_i = w_{i-1} + w_{i-4}, & altrament. \end{cases} \tag{2.4.11}$$

Per últim, $GeneracioClausRonda_{192}$

$$\begin{array}{rccc} GeneracioClausRonda_{192}: & \mathbb{K}^{24} & \longrightarrow & (\mathbb{K}^{16})^{13} \\ & c & \longmapsto & GeneracioClausRonda_{192}(c), \end{array} \tag{2.4.12}$$

amb $GeneracioClausRonda_{192}$,

$$\begin{cases} w_i = c_i, & si \quad 0 \leq i < 5, \\ w_i = SubWord(RotWord(w_{i-1})) + (x^{\frac{i}{6}}, \{00\}, \{00\}, \{00\}) + w_{i-4}, & si \quad 6 \leq i \leq 51 \ \wedge \ i(mod6) = 0, \\ w_i = w_{i-1} + w_{i-4}, & altrament. \end{cases} \tag{2.4.13}$$

Proposició 2.4.2. *La funció generació de claus de ronda de l'AES-128 està descrita a 2.4.11, mentre que la de l'AES-192 està descrita a 2.4.13.*

2.5 AddRoundKey

Un cop generada la clau de ronda, s'anomenarà l'aplicació que als estàndards s'anomena *AddRoundKey* com AR_i. Per tal que la notació sigui congruent amb la de l'apartat següent, primer s'ha de definir una aplicació projecció

$$\begin{array}{rccc} \pi_i: & E^{15} & \longrightarrow & E \\ & cr & \longmapsto & cl_i, \end{array} \tag{2.5.1}$$

on $\mathbb{K}^{16}$ és pensat com a espai vectorial (cf. inici el capítol 2), $cr = (cl_0, cl_1, \ldots, cl_{14})$ representa el vector de totes les claus de ronda, i $cl_i \in E$ la clau de ronda usada en la i-ésima ronda, formada per 16 bytes.

Si utilitzem la mateixa nomenclatura, la funció AR_i es pot escriure com

$$\begin{array}{rccc} AR_i: & E \times E^{15} & \longrightarrow & E \\ & (s, cr) & \longmapsto & s + \pi_i(cr), \end{array} \tag{2.5.2}$$

on s representa l'estat i π_i l'aplicació projecció. Com veurem en el següent apartat en l'AES-256, $0 \leq i \leq 14$.

Proposició 2.5.1. *La funció resultant d'aplicar AddRoundKey a tot l'estat és la següent,*

$$\begin{array}{rccc} AR_i: & E \times E^{15} & \longrightarrow & E \\ & (s, cr) & \longmapsto & s + \pi_i(cr), \end{array} \tag{2.5.3}$$

on s és l'estat i cr el vector clau de ronda.

2.6 Resum

Un cop descrites totes les operacions és descriurà com el criptosistema AES encripta un missatge pla de 128-bits, a través d'una clau consensuada prèviament.

Teorema 2.6.1. *La funció Xifratge descrita a continuació coincideix amb la descripció del xifratge de l'AES-256 als estàndards del [FIPS 2001].*

$$\begin{array}{rccc} Xifratge: & E \times E^{15} & \longrightarrow & E \\ & (p, cr) & \longmapsto & X(p, cr), \end{array} \tag{2.6.1}$$

on p representa el missatge pla, cr representa la clau de ronda i $X(p, cr)$ consisteix en:

$$X(p, cr) = AR_{14} \circ SR \circ SB \circ (AR_{13} \circ MC \circ SR \circ SB \circ \cdots \circ AR_1 \circ MC \circ SR \circ SB \circ (AR_0(p))), \tag{2.6.2}$$

on AR_j representa la funció *AddRoundKey* a la successió de bytes de la clau de ronda $cl_j = (a_{16j}, a_{16j+1}, \ldots, a_{16j+15})$. Mentre que les aplicacions SR, SB i MC representen les transformacions *ShiftRows*, *SubBytes* i *MixColumns* resprectivament.

El primer que es fa és afegir els primers 16 bytes (o 4 paraules) de la clau de ronda a l'estat original, es a dir; al text pla. Seguidament es comença una iteració que es duu a terme 13 vegades i que consisteix a aplicar a l'estat la funció *SubBytes*, *ShiftRows* i *MixColumns*, en aquest ordre, i finalment afegir-li els 16 bytes següents de la clau de

ronda. Un cop fetes les 13 iteracions, s'arriba a l'última part, que consisteix a aplicar un cop *SubBytes* i *ShiftRows* a l'estat per acabar sumant-li les últims 16 bytes que queden de la clau de ronda. Així s'arriba al text xifrat de l'AES.

S'ha explicat com es xifra amb l'AES-256. Per a l'AES-128 i l'AES-192, l'única cosa que canvia és la llargada de la clau, de 60 paraules a 44 i 52 paraules respectivament. Això fa que també s'hagi de restringir la quantitat d'iteracions de 13 a 9 i 11 respectivament; a part d'això, la funció *Xifratge* és exactament la mateixa.

Capítol 3

Desxifratge

En aquest apartat, s'explica com, a partir dels missatge xifrat i la clau, es pot desxifrar el missatge per tal d'obtenir novament el missatge pla. Primerament, es veurà que cada una de les funcions anteriors és invertible, i seguidament s'explicarà com s'aplica el desxifratge per a l'AES, ja que no és exactament el codi invers del xifratge.

3.1 Inversa de SubBytes

La funció *SubBytes*

$$\begin{aligned} SubBytes : \mathbb{K} &\longrightarrow \mathbb{K} \\ x &\longmapsto M \cdot inv(x) + b, \end{aligned} \tag{3.1.1}$$

és invertible, perquè es pot comprovar que $M^4 = Id$; per tant, $M^{-1} = M^3$:

$$M^{-1} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}. \tag{3.1.2}$$

Com que el cos base és $\mathbb{F}_2$ aleshores $-b = b$; per tant,

$$\begin{aligned} (SubBytes)^{-1} : \mathbb{K} &\longrightarrow \mathbb{K} \\ x &\longmapsto inv(\big[M^{-1}(x+b)\big]). \end{aligned} \tag{3.1.3}$$

Clarament, l'element invers de $M^{-1}(x+b)$ existeix ja que pertany al cos finit $\mathbb{K}$ on l'invers de tots els elements es pot calcular com $a \mapsto a^{254}$.

Per tant, s'acaba de demostrar que l'aplicació $(SubBytes)^{-1}$ es la transformació inversa de *SubBytes*. En conseqüència l'aplicació SB també té inversa, que consisteix en

$$SB^{-1}: \begin{array}{ccc} U^4 & \longrightarrow & U^4 \\ \begin{pmatrix} s_{0,0} \\ s_{1,0} \\ s_{2,0} \\ \vdots \\ s_{3,3} \end{pmatrix} & \longmapsto & \begin{pmatrix} (M^{-1}(s_{0,0}+\{63\}))^{254} \\ (M^{-1}(s_{1,0}+\{63\}))^{254} \\ (M^{-1}(s_{2,0}+\{63\}))^{254} \\ \vdots \\ (M^{-1}(s_{3,3}+\{63\}))^{254} \end{pmatrix}; \end{array} \tag{3.1.4}$$

en aquest cas es pot comprovar que no es pot expressar SB^{-1} de forma matricial; aquest fet complica llegurament la criptoanàlisi, com veurem en el capítol següent.

Com en el cas de l'aplicació *SubBytes* la informació d'aquesta transformació s'acostuma a tabular en una caixa del tipus Sbox. Tot i així, com que l'aplicació és bijectiva, es pot aprofitar la taula 2.1 i buscar l'element que es desitja invertir en la mateixa taula, un com s'hagi trobat la fila i columna en què estigui determinarà el valor del byte resultant després de la transformació.

Proposició 3.1.1. *L'aplicació inversa de SubBytes aplicada a tot l'estat és la següent,*

$$SB^{-1}: \begin{array}{ccc} U^4 & \longrightarrow & U^4 \\ \begin{pmatrix} s_{0,0} \\ s_{1,0} \\ s_{2,0} \\ \vdots \\ s_{3,3} \end{pmatrix} & \longmapsto & \begin{pmatrix} (M^{-1}(s_{0,0}+\{63\}))^{254} \\ (M^{-1}(s_{1,0}+\{63\}))^{254} \\ (M^{-1}(s_{2,0}+\{63\}))^{254} \\ \vdots \\ (M^{-1}(s_{3,3}+\{63\}))^{254} \end{pmatrix}, \end{array} \tag{3.1.5}$$

on M^{-1} és la matriu representada a 3.1.2.

3.2 Inversa de ShiftRows

Si expressem l'estat com $s_{i,j}$ amb $0 \leq i,j \leq 3$, es pot descriure la transformació *SubBytes* com $s'_{i,j} = s_{i,(j+i)mod4}$, i consisteix a fer correr els coeficients de l'estat i vegades cap a la dreta, on i és la fila en què es troba el coeficient.
Per tant, la seva funció inversa consistirà en córrer els coeficients i vegades cap a l'esquerre, això es pot traduir com $s'_{i,j} = s_{i,(j-i)mod4}$.

Com en el cas de *ShiftRows*, la transformació també es pot expressar de forma matricial si s'ordena l'estat per files ($s = (t_0, t_1, t_2, t_3)^t$):

$$L^3 = \begin{pmatrix} I & 0 & 0 & 0 \\ 0 & \mathfrak{R}^3 & 0 & 0 \\ 0 & 0 & \mathfrak{R}^2 & 0 \\ 0 & 0 & 0 & \mathfrak{R}^1 \end{pmatrix}. \tag{3.2.1}$$

Es fàcil comprovar que s'està davant de la inversa de L ja que si es multiplica L per la matriu de 3.2.1 s'obté:

$$\begin{pmatrix} I & 0 & 0 & 0 \\ 0 & \mathfrak{R}^4 & 0 & 0 \\ 0 & 0 & \mathfrak{R}^4 & 0 \\ 0 & 0 & 0 & \mathfrak{R}^4 \end{pmatrix}. \tag{3.2.2}$$

Però clarament, $\mathfrak{R}^4 = Id$ ja que $\mathfrak{R}^4$ es pot expressar com $s'_{i,j} = s_{i,(j+4)mod4} = s_{i,j}$; per tant, la relació anterior se satisfà i la matriu de 3.2.1 es la inversa de L.

Proposició 3.2.1. *La funció inversa de ShiftRows aplicada a tot l'estat és la següent,*

$$\begin{array}{rccc} SR^{-1}: & T^4 & \longrightarrow & T^4 \\ & \begin{pmatrix} t_0 \\ t_1 \\ t_2 \\ t_3 \end{pmatrix} & \longmapsto & \begin{pmatrix} I & 0 & 0 & 0 \\ 0 & \mathfrak{R}^3 & 0 & 0 \\ 0 & 0 & \mathfrak{R}^2 & 0 \\ 0 & 0 & 0 & \mathfrak{R}^1 \end{pmatrix} \cdot \begin{pmatrix} t_0 \\ t_1 \\ t_2 \\ t_3 \end{pmatrix}, \end{array} \tag{3.2.3}$$

on $\mathfrak{R}$ és la matriu esmentada a la secció 2.2.

3.3 Inversa de MixColumns

Com s'ha vist a la secció 2.3, *MixColumns* consisteix a transformar cada columna de l'estat en un element de $\mathbb{A}$, multiplicar-lo per l'element constant $a(y)$ i, seguidament, tornar a passar l'element a l'estat:

$$\begin{array}{rccccccc} MixColumns_j: & U & \longrightarrow & \mathbb{A} & \longrightarrow & \mathbb{A} & \longrightarrow & U \\ & u_j & \longmapsto & u_j(y) & \longmapsto & a(y)\cdot u_j(y) & \longmapsto & u'_j. \end{array} \tag{3.3.1}$$

Aleshores, clarament l'aplicació inversa existeix si i només si existeix un element invers a $a(y) \in \mathbb{A}$; l'element invers no té perquè existir ja que l'anell $\mathbb{A}$ no és un cos; per tant, no està garantit que existeixi invers de qualsevol element de l'anell. Tot i així, per a l'element determinat $a(y)$ de l'AES, l'element invers existeix i és concretament:

$$\begin{array}{rl} a^{-1}(y) & = \{0B\}y^3 + \{0D\}y^2 + \{09\}y + \{0E\} = \\ & = (x^3+x+1)y^3 + (x^3+x^2+1)y^2 + (x^3+1)y + (x^3+x^2+x). \end{array} \tag{3.3.2}$$

Per tant, es pot definir $MixColumns^{-1}$:

$$\begin{array}{rccccccc} (MixColumns_j)^{-1}: & U & \longrightarrow & \mathbb{A} & \longrightarrow & \mathbb{A} & \longrightarrow & U \\ & u_j & \longmapsto & u_j(y) & \longmapsto & a^{-1}(y)\cdot u_j(y) & \longmapsto & u'_j. \end{array} \tag{3.3.3}$$

Si es segueixen els mateixos passos que en la secció *MixColumns*, s'obté una matriu $S^{-1} \in GL(4, \mathbb{K})$, que es la següent:

$$S^{-1} = \begin{pmatrix} \{0E\} & \{0B\} & \{0D\} & \{09\} \\ \{09\} & \{0E\} & \{0B\} & \{0D\} \\ \{0D\} & \{09\} & \{0E\} & \{0B\} \\ \{0B\} & \{0D\} & \{09\} & \{0E\} \end{pmatrix}. \tag{3.3.4}$$

Per tant, l'aplicació MC^{-1} es pot descriure com

Proposició 3.3.1. *La funció inversa de MixColumns aplicada a tot l'estat és la següent,*

$$\begin{array}{rccc} MC^{-1}: & U^4 & \longrightarrow & U^4 \\ & \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \end{pmatrix} & \longmapsto & \begin{pmatrix} S^{-1} & 0 & 0 & 0 \\ 0 & S^{-1} & 0 & 0 \\ 0 & 0 & S^{-1} & 0 \\ 0 & 0 & 0 & S^{-1} \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \end{pmatrix}, \end{array} \tag{3.3.5}$$

on S^{-1} és la matriu 3.3.4.

Queda demostrat doncs, que existeix funció inversa de l'aplicació *MixColumns*.

3.4 Inversa d'AddRoundKey

Trobar la funció inversa d'*AddRoundKey* és fàcil. Només cal recordar que s'està treballant sobre el cos base $\mathbb{F}_2$; per tant, l'element oposat és tornar a sumar l'element, ja que així s'obté $s_{i,j} + w_{i,j} + w_{i,j} = s_{i,j} + 2w_{i,j} = s_{i,j}$; és a dir, la inversa de l'aplicació *AddRoundKey* és la mateixa funció *AddRoundKey*. O sigui

$$\begin{array}{rccc} AR_i^{-1}: & E \times E^{15} & \longrightarrow & E \\ & (s, cr) & \longmapsto & s + \pi_i(cr), \end{array} \tag{3.4.1}$$

on π_i és l'aplicació projecció per la ronda i-ésima de l'AES (cf. la secció 2.5).

Proposició 3.4.1. *La funció inversa d'AddRoundKey és ella mateixa.*

3.5 Resum

Als estàndards, [FIPS 2001], es presenten diverses maneres d'implementar la funció inversa de l'AES. En algunes, el programa que s'utilitza per implementar la funció inversa de l'AES, no consisteix exactament en aplicar les inverses de les funcions parcials en ordre invers a l'aplicació *Xifratge*. Existeix alguna permutació entre aplicacions parcials que tot seguit s'explicarà.

Com en l'aplicació de xifratge, el desxifratge comença afegint la clau de ronda, però en aquest cas es sumen les 16 últims bytes enlloc dels 16 primers. Seguidament, també s'entra en una iteració, on s'apliquen a l'estat les funcions *Inversa ShiftRows*, *Inversa SubBytes* i se li afegeix la clau de ronda amb ordre invers; és a dir, si ja s'han sumat els 16 últims bytes, doncs segueix amb els 16 penúltims bytes; o sigui, els que estan en posició 223 fins a la 208, i així successivament; per últim, s'aplica la funció *Inversa de MixColumns* a l'estat; aquesta iteració es repeteix 13 vegades. Finalment, s'aplica a l'estat la funció *Inversa ShiftRows* i la funció *Inversa SubBytes*, per acabar sumant-li les primeres 4 paraules de la clau de ronda.

Matemàticament, la funció *Desxifratge* es pot escriure com

$$\begin{array}{rccc} Desxifratge: & E \times E^{15} & \longrightarrow & E \\ & (x, cr) & \longmapsto & DX(x, cr) \end{array}, \tag{3.5.1}$$

on x representa el missatge xifrat, cr representa la clau de ronda i $DX(s)$ consisteix en:

$$DX(x, cr) = AR_0 \circ SB^{-1} \circ SR^{-1} \circ (MC^{-1} \circ AR_1 \circ SB^{-1} \circ SR^{-1} \cdots \circ MC^{-1} \circ AR_{13} \circ SB^{-1} \circ SR^{-1} \circ (AR_{14}(x))). \tag{3.5.2}$$

Teorema 3.5.1. *L'aplicació Desxifratge descrita anteriorment és la funció utilitzada per desxifrar els missatges xifrats de l'AES-256 i és la inversa de la funció Xifratge*

La demostració és directa a partir de les seccions anteriors. □

Com s'ha dit a l'inici, els estàndards també preveuen que les funcions *Inversa SubBytes* i *Inversa ShiftRows* estiguin en ordre invers del que haurien d'estar si realment s'invertís el xifratge. Però com ja s'ha vist aquestes funcions commuten, en conseqüència, aquest fet no suposa un problema.

Capítol 4

Criptoanàlisi

Per tal de realitzar una anàlisi de la seguretat de l'AES, i veure com ajuda cada operació a la seguretat de l'AES, és útil presentar cada ronda de l'AES com la següent aplicació:

$$\begin{array}{rccc} Xifratge: & \mathbb{K}^{16} & \longrightarrow & \mathbb{K}^{16} \\ & a & \longmapsto & \hat{S}\cdot\hat{L}\cdot(\hat{M}\cdot\hat{inv}(a)+\hat{\{63\}})+k_i, \end{array} \tag{4.0.1}$$

on les matrius $\hat{S}$, $\hat{L}$, $\hat{M}$ són molt semblants a les esmentades anteriorment, però amb certes diferències. El mateix passa amb el $\{63\}$ que en aquest cas no representarà un sol byte, sinò un conjunt de 16 bytes del mateix valor.

El canvi de les matrius associades a les transformacions no implica cap canvi en les operacions propiament dites, però està clar que cal que es triï una ordenació determinada per a l'estat, i això causa canvis de les matrius associades.
Per convenció, s'agafa l'estat ordenat per columnes, i en conseqüència, la matriu associada a *MixColumns* no varia, és a dir;

$$\hat{S}=\begin{pmatrix} S & 0 & 0 & 0 \\ 0 & S & 0 & 0 \\ 0 & 0 & S & 0 \\ 0 & 0 & 0 & S \end{pmatrix}. \tag{4.0.2}$$

En canvi, és clar que si la matriu associada a *MixColumns* no canvia, la matriu associada a *ShiftRows* patirà un canvi de base, tot i que conservarà les seves propietats; la matriu resultant d'aquest canvi de base s'anomenarà $\hat{L}$. $\hat{M}$ serà la transformació associada a la matriu M en l'espai $\mathbb{K}^{16}$; aquesta transformació només es pot expressar de forma matricial si en lloc d'expressar tota la transformació en l'espai $\mathbb{K}^{16}$ s'expresa sobre el cos finit $\mathbb{F}_2^{128}$. Aquest fet augmenta la complexitat de l'atac per força bruta ja que impedeix desenvolupar un sistema d'equacions on les incognites siguin els bytes de l'estat, obliga que el sistema d'equacions estigui format per bits.

Si es recorda que la funció $inv:\mathbb{K}\longrightarrow\mathbb{K}$ actua de manera $inv(a)=a^{254}$, obtenim la funció

$$\begin{array}{rccc} \hat{inv}: & \mathbb{K}^{16} & \longrightarrow & \mathbb{K}^{16} \\ & (s_{i,j})_{0\le i,j\le 3} & \longmapsto & \left(s_{i,j}^{254}\right)_{0\le i,j\le 3}. \end{array} \tag{4.0.3}$$

Només queda definir el $\hat{\{63\}}$ i com ja s'ha dit, es el vector que esta format per 16 bytes de valor $\{63\}$.

El primer que cal destacar és que $\hat{S}\hat{L}\{\hat{63}\} = \{\hat{63}\}$, ja que $\hat{L}\{\hat{63}\} = \{\hat{63}\}$, degut a que la reordenació d'un vector amb tots els coeficients iguals deixa el vector invariant. El fet que $\hat{S}\{\hat{63}\} = \{\hat{63}\}$ és una mica més subtil, però no gaire. La transformació es pot expressar com, $\{02\}\cdot\{63\}+\{01\}\cdot\{63\}+\{01\}\cdot\{63\}+\{03\}\cdot\{63\} = x\cdot\{63\}+\{63\}+\{63\}+(x+1)\cdot\{63\} = 2x\cdot\{63\}+2\{63\}+\{63\} = \{63\}$.

Pensant en com el sistema pot esdevenir insegur, si més no sota atacs teòrics, emergeix la idea que aquest sistema sigui una aplicació afí, això passa si i només si l'aplicació *Xifratge* esmentada a l'equació 2.6.1 és afí.

És facil veure que l'aplicació inversa no és afí ja que si escrivim una matriu Y tal que les columnes són els inversos respectius de cada element de la base $\{1, x, x^2, x^3, x^4, x^5, x^6, x^7\}$,

$$Y = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}; \tag{4.0.4}$$

si aleshores apliquem la matriu Y al vector $a = x^5 + x^2$, l'element Ya no és l'invers de a. Per tant, gràcies a gràcies a la inversió dels elements de l'estat, cap de les rondes de l'AES és afí. Això no implica directament que la funció *Xifratge* no sigui afí, ja que la composició de funcions que no són afins pot esdevenir afí.

Teorema 4.0.1. *En general, la funció Xifratge no és afí.*

Demostració: Per tal de demostrar que l'aplicació *Xifratge* no és afí, només s'ha de probar que existeix alguna clau per a la qual no se satisfà que el xifratge és afí.

Anem a comprovar-ho per a la clau $k = \{000102030405060708090A0B0C0D0F\}$. Si el *Xifratge* fos afí es compliria que donat un missatge pla e_1 aleshores $X(e_1) = Me_1 + b$, on $M \in \mathcal{M}(16, \mathbb{K})$ i $b \in \mathbb{K}^{16}$. A més es compliria que un altre text pla e_2 es xifraria com $X(e_2) = Me_2 + b$ i, si se sumen les dues expressions tindríem que $M(e_1 + e_2) = X(e_1) + X(e_2)$, mentre que el xifratge del text pla $e_1 + e_2$ seria $X(e_1 + e_2) = M(e_1 + e_2) + b = X(e_1) + X(e_2) + b$. D'aquesta expressió es pot obtenir el vector $b = X(e_1 + e_2) + X(e_1) + X(e_2)$. Si ara fem el mateix procés amb dos vectors diferents de e_1 i e_2 s'hauria de complir que $X(e_3 + e_4) = X(e_3) + X(e_4) + b$. En el cas estudiat això no és cert. S'ha utilitzat com a $e_1 = \{10000000000000000000000000000000\}$ i $e_2 = \{01000000000000000000000000000000\}$ xifrant-los (cf. la secció A.1 per veure els xifrats explícits), s'obté que

$$\begin{cases} X(e_1 + e_2) = \{346C2D68F8588C329C288F3A2FDCEE08\}, \\ b = \{FE8FB22286C53175F0994AA42DC007EA\}. \end{cases} \tag{4.0.5}$$

Per altra banda, veiem que si $e_3 = \{00100000000000000000000000000000\}$ i $e_4 = \{00010000000000000000000000000000\}$ i es fa el corresponent xifratge, aleshores

$$\begin{cases} X(e_3) + X(e_4) + b = \{9F639C60DEB4871CFB14AE23D33DD955\}, \\ X(e_3 + e_4) = \{5DC5BD977DE02639F80DE2371DBB28F0\}; \end{cases} \tag{4.0.6}$$

com es pot veure, $X(e_3+e_4) \neq X(e_3)+X(e_4)+b$; per tant, l'aplicació *Xifratge* no és afí, com volíem demostrar. □

Els atacs que han provat ser més eficaços en la criptografia moderna són la *criptoanàlisi lineal* i la *criptoanàlisi diferencial*. El primer es basa en buscar aproximacions afins a les funcions de xifratge. Realitzar aquesta tasca a l'AES és pràcticament impossible a causa de l'alt nombre de rondes que té la funció, s'haurien de trobar aproximacions afins de les inverses realitzades consecutivament cosa que és difícil. Per altra banda, la criptoanàlisi diferencial es basa en com diferències en els bytes del text pla afecten als del text xifrat; és a dir, s'utilitzen parelles de missatges plans relacionades amb una diferència constant i es mira la diferència entre les parelles de missatges xifrats per veure si es detecta algun tipus de patrons estadístics que ajudin a trobar la clau (cf. apèndix A per trobar alguns exemples). Aquest atac no funciona a causa de la gran difusió que aporta la funció *MixColumns*. És per això que fins ara es creu que l'atac més eficient per criptoanalitzar l'AES és l'atac a la clau per força bruta.

4.1 Atac per força bruta

Per poder executar un atac per força bruta, primerament s'han de plantejar les equacions del sistema que es vol resoldre per tal de poder veure com encarar la resolució i després veure si aquesta és factible. Com que les aplicacions són totes lineals excepte la part de *SubBytes* on apareix la funció inv, és útil denotar dues variables; per una banda, els estats w_i seran els conjunts de 16 bytes d'arribada de la funció inv, mentre que els estats x_i seran els corresponents conjunts de 16 bytes de sortida de la funció inv; l'índex i representa la ronda de l'AES en què es troba l'estat. El plantejament teòric d'aquest atac es realitzarà per a l'AES-128 i un cop vistos els resultats es faran extensibles als altres dos models.

El primer que cal esbrinar és si aquest model porta a un sistema d'equacions determinat o indeterminat, ja que en cas que fos indeterminat s'hauria de buscar algun altre mètode per resoldre'l.

El sistema té 10 rondes, per tant hi haurà $10 \cdot 128 = 1280$ bits d'arribada i bits de sortida de la funció inversió. Per altra banda, s'ha d'afegir 11 cops la clau de ronda; per tant, el nombre de variables de la clau de ronda serà $11 \cdot 128 = 1408$, els 128 bits que s'utilitzen de la clau de ronda cada cop que passa la funció s'anomenaran k_i. En aquest cas però, no s'ha tingut en compte que en la funció *GeneracioClauRonda* també existeixen bytes que s'inverteixen; per tant, les inversions s'hauran de comptar doble. Si ens ho mirem correctament, hi ha 40 bytes que s'inverteixen, és a dir; tindrem 40 bytes d'entrada i de sortida nous però com que aquests ja s'han comptat una vegada només es tindrà en compte un cop aquest 40, per tant, $1408 + 40 \cdot 8 = 1728$ serà el nombre de bits que s'han de tenir en compte com a variables de la clau de ronda. Es disposa d'un sistema de 4288 variables. Anem a veure de quantes equacions independents determinen l'AES.

Per fer-ho cal disposar d'una matriu que s'anomenarà N que serà $N = \hat{S}\hat{L}\hat{M}$ en l'espai vectorial $\mathbb{F}_2^{128}$, es a dir; n és una matriu 128×128 que surgeix del producte de les matrius utilitzades en el criptosistema. Tornant a les equations, per una banda, es disposa de 9 equacions del tipus $w_i = Nx_{i-1} + \{63\} + k_i$, amb $1 \leq i \leq 9$ de manera semblant es poden treure les equacions relacionades amb el text pla, p, i el text xifrat, c, i són $w_0 = p + k_0$ i $c = \hat{L}\hat{M}x_9 + \{63\} + k_{10}$. És a dir, que s'obtenen $11 \cdot 128 = 1408$ equacions lineals i independents entre elles.

Per tal de tenir en compte la inversió s'ha de recordar que cada pas de ronda de l'estat x_i, w_i o de la clau de ronda k_i està formada per 16 bytes que són sota els que recau l'acció. Per tant, si volem escriure de forma correcta aquestes equacions caldrà introduir un index addicional, $0 \leq k \leq 15$. Primer de tot, cal destacar que l'equació referent a la funció inversió, $w_{i,k}x_{i,k} = 1$, no és sempre certa; però, si considerem que tots els bytes son equiprobables, tenim que existeix una probabilitat del $\frac{255}{256}$ que ho sigui.

També s'ha de tenir en compte com es relacionen x i w a través de la funció *inv*. Aquesta equació s'ha de transformar en una equació de bits; per fer-ho, cal introduir les matrius que consisteixen a multiplicar un element $a \in \mathbb{K}$ per qualsevol element del mateix cos. Que és possible ja que la multiplicació entre elements de $\mathbb{K}$ és $\mathbb{F}_2$-lineal.

Exemple 4.1.1. Sigui $a = x \in \mathbb{K}$ la matriu que resultant de multiplicar qualsevol element per a suposant la base usual en $\mathbb{K}$; és a dir, $\{1, x, x^2, x^3, x^4, x^5, x^6, x^7\}$ s'obté

$$C_x = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}. \tag{4.1.1}$$

De la mateixa manera es podria escriure una matriu per a qualsevol element de $\mathbb{K}$.

Per tant, es pot representar l'equació $x_i w_i = 1$ de la forma següent: $C_{x_i} w_i = (1, 0, 0, 0, 0, 0, 0, 0)$, on 7 de les equacions són sempre certes i una d'elles (la primera) se satisfà amb una alta probabilitat. De la mateixa manera, les següents relacions es compleixen sempre: $w_i x_i^2 = x_i$ i $x_i w_i^2 = w_i$; en aquest cas, també es poden traduir aquestes relacions entre bytes com a relacions entre bits. De fet, existeix la matriu que eleva un element de $\mathbb{K}$ al quadrat; no deixa de ser l'expressió matricial de l'automorfisme de Frobenius.

Exemple 4.1.2. La matriu que eleva un element $a \in \mathbb{K}$ al quadrat és:

$$Q = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}. \tag{4.1.2}$$

Per tant, les anteriors relacions es poden escriure de la forma $(C_{w_i}Q + Id)x_i = 0$ i $(C_{x_i}Q + Id)w_i = 0$, d'on s'obtenen 16 noves equacions. Finalment, es poden trobar dues equacions que relacionen bytes que són independents a les anteriors $w_i x_i^4 = x_i^3$ i $x_i w_i^4 = w_i^3$, que es poden traduir a $(C_{w_i}Q^2 + Q)x_i = 0$ i $(C_{x_i}Q^2 + Q)w_i = 0$. D'on s'obtenen 16 noves equacions. I per tant, d'aquí s'obtenen $10 \cdot 16 \cdot (1 + 7 + 2 \cdot 16) = 6400$ equacions quadrades.

Finalment també s'han de tenir en compte les equacions del cos $\mathbb{F}_2$ que afirmen que qualsevol element d'aquest cos satisfà $z^2 + z = 0$; per tant, d'aquest tipus en tenim $128 \cdot 10 = 1280$ per les a variables d'arribada i de sortida. Es pot fer el mateix recompte

per a les equacions que inclouen les claus i es veu que es tenen $40 \cdot 40$ equacions d'inverses, 1728 equacions del cos, i 1280 equacions lineals (són les funcions amb les quals es genera la clau de ronda).

Si es fa el còmput general, es té un sistema amb 4288 variables i 14976 equacions, que correspon a un sistema d'equacions sobredeterminat. A canvi de fer més complex el sistema d'equacions quadràtic es poden substituir les equacions lineals a les quadràtiques i s'obté un sistema amb 1600 variables i 9600 equacions quadrades. A la pràctica, aquestes equacions són d'una complexitat molt elevada i no són manejables.

És per això que es dota l'AES d'unes noves propietats que simplifiquen les equacions. Aquest nou sistema s'anomena BES (Big Encryption Standard) i és una eina teòrica per per l'estudi de l'AES; en general aquest sistema no s'utilitza per xifrar missatges.

4.2 BES

El BES neix amb l'objectiu de simplificar les equations que relacionen el text pla i la clau amb el missatge xifrat, de manera que amb aquest sistema, totes les equacions actuen sobre bytes, elements de $\mathbb{K}$; cosa que simplifica el sistema.

Recordem que l'AES utilitza un missatge pla de 128 bits; en el BES, aquest mateix missatge pla correspondrà a un estat de 128 bytes; es a dir, serà un element de $\mathbb{K}^{128}$. Tot seguit, s'explicarà com es duu a terme el procés i com es pot descriure l'AES mitjançant el BES.

La funció en què es basa el BES és la funció *Conjugació* en què els components s'obtenen per aplicació successiva de l'automorfisme de Frobenius

$$\begin{array}{rccc} \phi: & \mathbb{K} & \longrightarrow & \mathbb{K}^8 \\ & a & \longmapsto & (a^{2^0}, a^{2^1}, a^{2^2}, a^{2^3}, a^{2^4}, a^{2^5}, a^{2^6}, a^{2^7}). \end{array} \tag{4.2.1}$$

Per tant, un estat qualsevol de l'AES, s'incrusta al BES a través de:

$$\begin{array}{rccc} \tilde{\phi}: & \mathbb{K}^{16} & \longrightarrow & \mathbb{K}^{128} \\ & (s_{0,0}, s_{1,0}, \cdots, s_{3,3}) & \longmapsto & (\phi(s_{0,0}), \phi(s_{1,0}), \cdots, \phi(s_{3,3})). \end{array} \tag{4.2.2}$$

L'espai $\mathbb{B} = \tilde{\phi}(\mathbb{K}^{16}) \subset \mathbb{K}^{128}$ és on pertany el nou estat del BES, també hi pertany el text pla i el text xifrat. A més, és relativament fàcil transformar les claus de ronda de l'AES en claus de ronda del BES utilitzant la funció ϕ. També és convenient destacar que la funció ϕ compleix que $\phi(inv(a)) = inv(\phi(a))$.
Aquest nou estat s'etiquetarà de la forma següent:

$$(b_{0,0,0}, ..., b_{0,0,7}, b_{1,0,0}, ...b_{3,3,7}), \tag{4.2.3}$$

on $b_{i,j,k} = s_{i,j}^{2^k}$. És fàcil escriure les funcions *MixColumns* i *ShiftRows* adaptades a aquest nou sistema, per una banda *ShiftRows* consisteix a substituir cada 1 de la matriu L per una matriu identitat 8×8, $\mathbb{I}_{8\times 8}$.

D'altra banda, la funció *MixColumns* actuarà sobre $u_i^k = (s_{0,i}^{2^k}, s_{1,i}^{2^k}, s_{2,i}^{2^k}, s_{3,i}^{2^k})$ i les matrius que s'hauran d'utilitzar seran les següents:

$$S = \begin{pmatrix} x^{2^k} & (x+1)^{2^k} & 1 & 1 \\ 1 & x^{2^k} & (x+1)^{2^k} & 1 \\ 1 & 1 & x^{2^k} & (x+1)^{2^k} \\ (x+1)^{2^k} & 1 & 1 & x^{2^k} \end{pmatrix}. \tag{4.2.4}$$

Per tal de expressar *SubBytes* per al del BES, cal tenir en ment que qualsevol transofrmació lineal que actui sobre $\mathbb{K}$ es pot expressar com el polinomi $f(x) = a_0x^{2^0} + a_1x^{2^1} + a_2x^{2^2} + a_3x^{2^3} + a_4x^{2^4} + a_5x^{2^5} + a_6x^{2^6} + a_7x^{2^7}$, on $a_i \in \mathbb{K}$. Per tant, es pot expressar la part lineal de la funció *SubBytes* a través del següent polinomi de coeficients a $\mathbb{K}$:

$$\begin{array}{ccc} \mathbb{K} & \longrightarrow & \mathbb{K} \\ a & \longmapsto & \{05\}a^{2^0} + \{09\}a^{2^1} + \{F9\}a^{2^2} + \{25\}a^{2^3} + \{F4\}a^{2^4} + \{01\}a^{2^5} + \{B5\}a^{2^6} + \{8F\}a^{2^7}. \end{array} \tag{4.2.5}$$

Es pot escriure el global de l'aplicació *SubBytes* a partir de

$$\begin{array}{rccl} SB: & \mathbb{K} & \longrightarrow & \mathbb{K} \\ & a & \longmapsto & \{05\}a^{255-2^0} + \{09\}a^{255-2^1} + \{F9\}a^{255-2^2} + \{25\}a^{255-2^3} + \\ & & & +\{F4\}a^{255-2^4} + \{01\}a^{255-2^5} + \{B5\}a^{255-2^6} + \{8F\}a^{255-2^7}. \end{array} \tag{4.2.6}$$

En aquest cas, si es considera que els respectiu conjugats de cada element formen una base, es pot escriure la transformació en forma de matriu, cosa que resulta útil per realitzar els següents passos.

De la mateixa manera, es pot estendre la funció *GeneracióClausRonda* per representar-la en el sistema BES, aprofitant el que s'ha dit anteriorment.

Per tant, es pot expressar cada ronda del xifratge de la següent manera:

$$\begin{array}{rccl} Xifratge: & \mathbb{B} & \longrightarrow & \mathbb{B} \\ & s_{\mathbb{B}} & \longmapsto & H_{\mathbb{B}} inv(s_{\mathbb{B}}) + k_{i\mathbb{B}}, \end{array} \tag{4.2.7}$$

on $H_{\mathbb{B}}$ representa la matriu que engloba totes les transformacions lineals i $s_{\mathbb{B}}$ es com s'ha anomenat l'estat incrustat en el BES. En el cas de l'AES no es podia representar aquesta matriu ja que en el cas de la matriu M aquesta involucrava els bits, mentre que la resta de matrius involucraven bytes. Aquest és el gran avantatge de la representació del BES respecte la representació de l'AES.

Ara es mostrarà el sistema d'equacions obtingut amb el BES i s'anomenaran les maneres que actualment s'han utilitzat per resoldre'l. Per tant, seguint la lògica del sistema d'equacions de l'AES, es té, per una banda, que

$$w_{0,j,k} = p_{j,k} + k_{0,j,k}, \tag{4.2.8}$$

on es considera que $0 \leq j \leq 15$; és a dir, denota cada byte de l'estat, mentre que $0 \leq k \leq 7$ i que denota cada un dels elements conjugats del byte amb $k = 0$. Com ja s'ha dit en l'apartat anterior, p representa el text pla.
A part d'aquestes equacions, tenim les següents, corresponents a les 9 pròximes rondes:

$$w_{i,j,k} = H_{\mathbb{B}} x_{(i-1),j,k} + k_i, \tag{4.2.9}$$

on $1 \leq i \leq 9$.

Com ja s'ha justificat anteriorment, l'última ronda es lleugerament diferent ja que en aquest cas no es duu a terme l'aplicació *MixColumns*, $H^*_{\mathbb{B}}$. Per tant s'obté que

$$c_{j,k} = H^*_{\mathbb{B}} x_{9,j,k} + k_{10,j,k}. \tag{4.2.10}$$

Finalment, si es considera que no hi ha inversió del 0, també tenim que

$$x_{i,j,k} w_{i,j,k} + 1 = 0. \tag{4.2.11}$$

Ja només falten les equacions que relacionen els estats de conjugació

$$x^2_{i,j,k} + x_{i,j,k+1} = 0 \ ; \quad w^2_{i,j,k} + w_{i,j,k+1} = 0, \tag{4.2.12}$$

on $0 \leq k \leq 7$.

Es poden trobar de manera similar les equacions per a la generació de claus de ronda. Aquestes equacions, com es pot apreciar, són molt similars a les equacions per a l'AES; l'única característica especial que tenen és que actuen totes sobre $\mathbb{K}$ i, per tant, en aquest sentit la seva resolució és més simple.

El mètode més utilitzat per resoldre aquestes equacions és mitjançant bases de Gröbner i tota la teoria desenvolupada en aquest sentit. Tot i així, els últim resultats estimen que amb aquest mètode caldrien de l'ordre de 2^{330} operacions, [Cid, Murphy & Robshaw 2006], en el cos $\mathbb{K}$ que és una quantitat totalment prohibitiva per realitzar el càlcul en l'estat de la tecnologia actual. En aquest treball no s'ha estudiat com resoldre el sistema d'equacions utilitzant el mètode de les bases de Gröbner i, per tant, no s'ha comprovat el resultat esmentat prèviament. Dit això, podria ser un bon punt de partida per a la seva continuació.

Apèndix A

Exemples

A.1 L'aplicació Xifratge no és afí

En aquest apèndix es podran veure exemples de xifratge per a diferents claus o textos plans. Aquests exemples resulten útils per veure alguns conceptes que s'han desenvolupat al llarg d'aquest treball. Altres xifratges s'utilitzen en algunes demostracions, com és el cas d'aquesta primera figura on es veu el xifratge que s'utilitza per demostrar el teorema 4.0.1.

Plaintext:	10000000000000000000000000000000	Plaintext:	00100000000000000000000000000000
Key:	000102030405060708090A0B0C0D0E0F	Key:	000102030405060708090A0B0C0D0E0F
Ciphertext:	299F7C29A3E13AE7F64ECBA062FC7560	Ciphertext:	FF5D987E268F87A0AFE65D004452E7B7
Plaintext:	01000000000000000000000000000000	Plaintext:	00010000000000000000000000000000
Key:	000102030405060708090A0B0C0D0E0F	Key:	000102030405060708090A0B0C0D0E0F
Ciphertext:	E37CD363DD7C87A09AFF0E3E60E09C82	Ciphertext:	9EB1B63C7EFE31C9A46BB987BAAF3908
Plaintext:	11000000000000000000000000000000	Plaintext:	00110000000000000000000000000000
Key:	000102030405060708090A0B0C0D0E0F	Key:	000102030405060708090A0B0C0D0E0F
Ciphertext:	346C2D68F8588C329C288F3A2FDCEE08	Ciphertext:	5DC5BD977DE02639F80DE2371DBB28F0

Figura A.1: Xifratges dels missatges plans involucrats en la demostració del teorema 4.0.1. S'ha utilitzat el programa proporcionat per [Project Nayuki 2013], utilitzant l'AES-128.

A.2 Criptoanàlisi Diferencial

Quatre exemples de criptoanàlisi diferencial. Per tal de plantejar un atac raonable calen molts més textos plans i els corresponents textos xifrats per tal de fer una anàlisi estadística. Aquests exemples s'utilitzen simplement per exemplificar la criptoanàlisi diferencial.

Plaintext:	00112233445566778899AABBCCDDEEFF	Plaintext:	5C6A879F31022A45189E4A6780AA4E5F
Key:	000102030405060708090A0B0C0D0E0F	Key:	000102030405060708090A0B0C0D0E0F
Ciphertext:	69C4E0D86A7B0430D8CDB78070B4C55A	Ciphertext:	4D5B41C888E207D626BDD584CAE7881E
Plaintext:	00002233445566778899AABBCCDDEEFF	Plaintext:	5C7B879F31022A45189E4A6780AA4E5F
Key:	000102030405060708090A0B0C0D0E0F	Key:	000102030405060708090A0B0C0D0E0F
Ciphertext:	12BF03D636F8173D7D71FCA75E072970	Ciphertext:	30B73F81FAA3DB1CAF7D8AACDC631C3A
RestaPlain	00110000000000000000000000000000.	RestaPlain	00110000000000000000000000000000.
RestaCiph	7B7BE30E5C83130DA5BC4B272EB3EC2A	RestaCiph	7DEC7E497241DCCA89C05F2816849424
Plaintext:	153AF6855D6E7C98345A7BBA0A10E2C4	Plaintext:	11054A8FF7E552B2050E597C7BC8A66F
Key:	000102030405060708090A0B0C0D0E0F	Key:	000102030405060708090A0B0C0D0E0F
Ciphertext:	345382CBC62ABC8A48A254E5F23F46F1	Ciphertext:	4B674CF219D971765EF54FB20CA67139
Plaintext:	152BF6855D6E7C98345A7BBA0A10E2C4	Plaintext:	11144A8FF7E552B2050E597C7BC8A66F
Key:	000102030405060708090A0B0C0D0E0F	Key:	000102030405060708090A0B0C0D0E0F
Ciphertext:	5170A8B4A9A2DB318160D345F597E8DF	Ciphertext:	D15B801C88550DD74A0DDDFC11532BA2
RestaPlain	00110000000000000000000000000000.	RestaPlain	00110000000000000000000000000000.
RestaCiph	65232A6F6F8867BBC9C287A007A8AE2E	RestaCiph	9A3CCCEE918D7CA114F8924E1DF55A9B

Figura A.2: Quatre exemples de la criptoanàlisi diferencial, les primeres sis files corresponen als missatges plans, la clau i els missatges xifratgs respectivament, mentre que les dues últimes files hi tenim la resta entre els dos missatges plans i la resta entre els missatges xifrats. S'ha utilitzat el programa proporcionat per [Project Nayuki 2013], utilitzant l'AES-128.

Referències

[Cid, Murphy & Robshaw 2006] Cid, Carlos; Murphy, Sean; Robshaw, Matthew: *Algebraic Aspects of Advanced Encryption Standard*. Springer, USA, 2006. ISBN: 0-387-24363-1.

[Murphy & Robshaw 2002] Murphy, Sean; Robshaw, Matthew: *Essential Algebraic Structure Within the AES*. Unversity of London, 2002.

[Dworkin, 2001] Dworkin, Morris: Recommendation for Block Cipher Modes of Operation. *NIST Special Publication* **800**-38A (2001).

[Frankel, Kent, Lewkowski, Orenaugh, Ritchey & Sharma, 2005] Frankel, Sheila; Kent, Karen; Lewkowski, Ryan; Orenaugh, Angela D.; Ritchey, Ronald W.; Sharma, Steven R.: Guide to IPsec VPNs. *NIST Special Publication* **800**-77 (2005).

[Daemen & Rijmen 2002] Daemen, Joan; Rijmen, Vincent: *The Design of Rijndael: AES - The Advanced Encryption Standard.* Springer, 2002.

[Shannon 1949] Shannon, Claude: Communication Theory of Secrecy Systems. *Bell System Technical Journal*, 1949.

[FIPS 2001] Federal Information Processing Standards: Announcing the ADVANCED ENCYPTION STANDARD (AES). *Federal Information Processing Standards Publication* **197**, november 26, 2001.

[Stinson] Stinson, Doug R.: *Criptography Theory and Practice*. Champan & Hall/CRC, 1995.

[Project Nayuki 2013] Project Nayuki: AES Cipher Internals in Excel, `https://www.nayuki.io/page/aes-cipher-internals-in-excel`, 2013.

Índex terminològic