Alma Mater Studiorum · Università di Bologna

facoltà di scienze matematiche, fisiche e naturali

Corso di Laurea in Matematica

L'ALGORITMO: ADVANCED ENCRYPTION STANDARD(AES)

Tesi di Laurea in Algoritmi della Teoria dei Numeri e crittografia

Relatore:
Chiar.mo Prof.
Davide Aliffi

Presentata da:
Davide Chieco

II Sessione
Anno Accademico 2013/2014

Indice

Introduzione

Scopo di questa tesi è dare una breve e concisa spiegazione del funzionamento dell' algoritmo crittografico Advanced Encryption Standard. Nei quattro capitoli vedremo, rispettivamente, la storia di AES ed i motivi per i quali venne adottato, le basi matematiche richieste per comprendere al meglio l'implementazione e le operazioni dell'algoritmo, la descrizione dettagliata di tutte le operazioni svolte all'interno di AES e, infine, le considerazione sulla sicurezza.

In Crittografia, l'Advanced Encryption Standard (AES), conosciuto anche come Rijndael, è un algoritmo di cifratura a blocchi utilizzato come standard dal governo degli USA. Nel 1997 il National Institute of Standards and Technology(NIST),agenzia del dipartimento del commercio che ha il compito di approvare gli standard federali per il governo degli Stati Uniti, richiese pubblicamente proposte di algoritmi tra i quali scegliere il sostituto del Data Encryption Standard (DES). Il nuovo algoritmo avrebbe dovuto consentire l'uso di chiavi di 128, 192, 256 bit, avrebbe dovuto operare su blocchi di 128 bit di input e avrebbe dovuto lavorare su vari hardware di diverso tipo. Inoltre avrebbe dovuto essere veloce e crittograficamente robusto. L'AES è stato annunciato dal NIST nel U.S. FIPS PUB 197 il 26 novembre del 2001, dopo un periodo di analisi di confronto durato cinque anni, in cui quindici progetti alternativi sono stati analizzati e studiati pubblicamente,dalla comunità crittografica internazionale, riconoscendo il Rijndael come il piú adatto . L'AES è divenuto uno standard effettivo il 26 maggio 2002. E' disponibile in diversi pacchetti di cifratura. L'AES è il primo algoritmo di cifratura pubblicamente

accessibile ed aperto. L'algoritmo di cifratura Rijndael è stato progettato da due crittografi belgi, Joan Daemen e Vincent Rijmen ed è un'evoluzione di un primo algoritmo precedente, chiamato Square. Il nome Rijndael è una parola composta con i nomi dei due inventori.

Capitolo 1

La scelta del NIST: AES

In questo primo capitolo vedremo i criteri di valutazione utilizzati dal NIST per valutare i potenziali candidati: questi criteri riguardano tutti gli aspetti delle moderne cifrature simmetriche a blocchi. In pratica si sono evoluti due insiemi di criteri.Quando il NIST emise la sua richiesta originaria di proposte di algoritmi nel 1997, stabiliva che gli algoritmi sarebbero stati confrontati sulla base di determinati criteri che elencheremo qui di seguito

1.1 Criteri di valutazione per lo standard AES

- **SICUREZZA:** che comprendeva altri quattro criteri
  - **Effettiva sicurezza:**da confrontare con gli altri algoritmi presenti
  - **Casualità:** il livello di indistinguibilitá dell'output dell'algoritmo da un blocco, generato in modo casuale
  - **Solidità:** delle basi matematiche su cui si fonda l'algoritmo
  - **altri fattori di sicurezza:** sollevati eventualmente durante il processo di valutazione

  L'enfasi nella valutazione era sulle possibilità reali di un attacco rispetto ai metodi di crittografia noti da tempo. Poichè in AES la chiave ha

dimensione minime di 128 bit, gli attacchi a forza bruta con le tecnologie attuali e future erano considerati impraticabili. Per questo motivo si enfatizzava l'analisi della resistenza dell'algoritmo rispetto agli altri attacchi noti.

- **COSTO:**
  - **Requisiti di proprietà intelletuale:** l'algoritmo dovesse essere disponibile globalmente in modo gratuito
  - **Efficienza computazionale:** divisa in 2 fasi, la prima valutava principalmente l'implementazione software, mentre la seconda riguardava la velocitá dell'algoritmo
  - **Requisiti di memoria:** la memoria necessaria per l'esecuzione dell'algoritmo candidato.

  Il NIST richiese che l'algoritmo AES potesse essere impiegato per un'ampia gamma di applicazioni. Di conseguenza, doveva avere un'elevata efficienza computazionale e doveva essere utilizzabile in applicazioni ad alta velocità, come per esempio i collegamenti a banda larga.

- **CARATTERISTICHE DELL'IMPLEMENTAZIONE:**
  - **Flessibilità:** fra i candidati, vennero preferiti, a parità di altre condizioni, gli algoritmi che rispondevano alle esigenze di un maggior numero di utenti
  - **Fattibilità hardware e software:** l'algoritmo candidato non doveva essere restrittivo rispetto alla piattaforma di implentazione hardware o software.
  - **Semplicità:** semplicità del progetto.

  Questa categoria includeva vari aspetti fra cui la flessibilità, la possibilità di impiego in varie implementazioni hardware e software e la semplicità, per facilitare l'analisi della sicurezza.

Utilizzando questi criteri, i 21 algoritmi vennero prima ridotti a 15 e poi a 5.Prima della valutazione finale i criteri di valutazione vennero aggiornati. Ecco i criteri utilizzati nella valutazione finale:

- **Ambienti con spazio limitato:** In alcune applicazioni, come per esempio le smart card, sono disponibili quantità relativamente ridotte di memoria RAM e/o ROM. E' necessario quindi valutare la memoria richiesta per il codice , la rappresentazione di oggetti come S-box e le sottochiavi

- **Implementazione hardware:** come nel caso del software, le implementazioni hardware possono essere ottimizzate in termini di velocità o di dimensioni. Tuttavia nel caso delle implementazioni hardware, più che in quelle software, le dimensioni si traducono direttamente in costi.In un computer di utilizzo generale, dotato di una grande quantità di memoria, raddoppiare le dimensioni di un programma crittografico potrebbe costituire una differenza trascurabile mentre raddoppiare l'area utilizzata in un dispositivo hardware può più che raddoppiare il costo del dispositivo stesso

- **Attacchi alle implementazioni:** Vi è un'altra classe di attacchi, oltre a quella descritta nella prima fase: questi attacchi utilizzano misure fisiche condotte durante l'esecuzione dell'algoritmo, per raccogliere informazioni su parametri(ad esempio la chiave). Tali attacchi sfruttano una combinazione di caratteristiche intrinseche dell'algoritmo e funzionalità dipendenti dall'implementazione.

- **Crittografia e decrittografia:** Se gli algoritmi di cifratura e decifrazione sono differenti sarà necessario ulteriore spazio per la decifrazione. Inoltre vi possono essere differenze nel tempo di esecuzione fra la cifratura e la decifrazione indipendentemente dal fatto che utilizzino algoritmi identici o differenti.

- **Agilità della chiave:** Fa riferimento alla capacità di cambiare rapidamente la chiave utilizzando la minima quantità possibile di risorse. Questo comprende sia il calcolo delle sottochiavi che la possibilità di commutare fra diverse configurazioni di sicurezza quando le chiavi sono già disponbili.

- **Altri elementi di versatilità e semplicità:** Indica due aree che rientrano in questa caratteristica. La flessibilità dei parametri include il supporto chiavi e blocchi di altre dimensioni e la possibilità di aumentare il numero di round per rispondere ai nuovi attacchi eventualmenti scoperti. La flessibilità dell'implementazione fa riferimento alla possibilità di ottimizzare gli elementi di cifratura per determinati ambienti.

- **Potenzialità di sfruttamento del parallelismo a livello delle istruzioni:** fa riferimento alla capacità di sfruttare le funzionalità di esecuzione parallela nei processi attuali e futuri.

1.2 Valutazione finale del NIST relativa all'algoritmo Rijndael

Vediamo ora i motivi per cui l'algoritmo di Rijndael é stato scelto come standard per la crittografia, utilizzando i criteri di valutazione che abbiamo appena descritto

- **Sicurezza generale:** Non esiste nessun attacco noto alla sicurezza Rijndael. Rijndael utilizza delle S-box come componenti non lineari, che dovrebbero fornire un margine di sicurezza adeguato, ma ha ricevuto qualche critica per la semplicità della sua struttura matematica che potrebbe essere soggetta ad attacchi . D'altra parte, la semplicità della struttura dovrebbe aver facilitato l'analisi della sicurezza durante il processo di valutazione.

- **Implementazione software:** Rijdael svolge molto bene le operazioni di cifratura e decifrazione in un'ampia varietà di piattaformea. Tuttavia vi è una riduzione delle prestazioni quando aumentano le dimensioni delle chiavi dato il maggior numero di fasi che devono essere eseguite. L'elevato parallelismo intrinseco di Rijndael facilita l'utilizzo efficiente delle risorse del microprocessore garantendo ottime prestazioni software. Il tempo di impostazione della chiave di Rijndael è rapido.

- **Ambienti con spazio limitato:** In generale, Rijndael è molto adatto ad ambienti con spazio limitato dove vengono implementate la crittografia o la decrittografia(ma non entrambe). Ha ridottissimi requisiti in termini di RAM e ROM.Un difetto è il fatto che i requisiti di memoria ROM aumentano quando la crittografia e la decrittografia sono entrambe implementate, anche se si rimane comunque all'interno dei limiti previsti per questo genere di ambienti. La generazione della chiave per la decrittografia è distinta da quella della crittografia.

- **Agilità della chiave:** Rijndael supporta il calcolo in tempo reale delle sottochiavi di crittografia. Rijndael richiede una sola esecuzione del processo di generazione della chiave per generare tutte le sottochiavi prima della decrittografia con una data chiave. Questo introduce un leggero carico in termini di risorse sull'agilità della chiave rijdael

- **Altri elementi di versatilità e semplicità:** Rijdael supporta blocchi di chiavi di 128 192 e 256 bit in qualsiasi combinazione.In linea di principio, la sruttura di Rijdael può impiegare blocchi e chiavi di qualsiasi dimensione (ma multipli di 32 bit) e anche variazioni nel numero delle fasi.

- **Potenzialità di sfruttamento del parallelismo a livello delle istruzioni:** Rijndael ha eccellenti potenzialità per sfruttare il parallelismo nella crittografia di un singolo blocco.

Capitolo 2

Basi Matematiche

2.1 Gruppi Anelli e Campi

In questo capitolo daremo una veloce definizione delle strutture matematiche che andremo ad utilizzare per descrivere l'algoritmo AES.

Definizione 2.1.1. *Un* **gruppo** G *è un insieme di elementi affiancato da un'operarazione binaria, rappresentata dal simbolo *, che associa a ciascuna coppia (a,b) di elementi di G un elemento (a*b) sempre di G in modo tale che valgono i seguenti assiomi:*

- *Chiusura: se* $(a, b) \in G$ *allora anche* $(a * b) \in G$
- *Associatività:* $a * (b * c) = (a * b) * c \quad \forall a, b, c \in G$
- *Elemento identità: vi è un elemento in G tale che* $a * e = e * a = a \quad \forall a \in G$
- *Elemento inverso: per ciascun a in G vi è un elemento* a' *sempre in G tale che* $a * a' = a' * a = e$

Se un gruppo contiene un numero finito di elementi, si chiama ***gruppo finito*** e il suo ordine è uguale al numero degli elementi. Altrimenti il gruppo è chiamato ***gruppo infinito***.

Un gruppo è chiamato ***abeliano*** se soddisfa la seguente condizione aggiuntiva: $a * b = b * a$ per ogni $a, b \in G$

Definizione 2.1.2. *Un gruppo G è detto* **ciclico** *se ogni elemento di G è una potenza $a^k (k \in \mathbb{N})$ di un elemento fisso $a \in G$. Si dice che l'elemento a genera il gruppo G.*

Definizione 2.1.3. *Un* **anello** *R è un insieme di elementi con 2 operazioni binarie, la somma e la moltiplicazione, tali che per ogni $a, b, c \in R$ valgono i seguenti assiomi*

- *R è un gruppo abeliano rispetto alla somma, ovvero R soddisfa gli assiomi dei gruppi abeliani*
- *Chiusura rispetto alla moltiplicazione: se $a, b \in R$ allora $ab \in R$*
- *Associatività rispetto alla moltiplicazione: $a(bc) = (ab)c \quad \forall a, b, c \in R$*
- *leggi distributive: $a(b + c) = ab + ac \quad \forall a, b, c \in R$*
  $(a + b)c = ac + bc \quad \forall a, b, c \in R$

Un anello si dice ***commutativo*** se soddisfa la seguente condizione aggiuntiva: $ab = ba \quad \forall a, b \in R$

Un ***dominio d'integrità*** è un anello che obbedisce ai seguenti assiomi:

- Identità moltiplicativa: vi è un elemento 1 in R tale che $a1 = 1a = aa \in R$
- Anullamento del prodotto: se $a, b \in R$ e $ab = 0$ allora o $a = 0$ o $b = 0$

Definizione 2.1.4. *Un* **campo** *F è un insieme di elementi con 2 operazioni binarie chiamate somma e moltiplicazione, tali che per ogni $a, b, c \in F$ valgano i seguenti assiomi:*

- *F è un dominio d'integrità*
- *Inverso moltiplicativo: per ogni $a \in F$, eccetto 0, vi è un elemento $a^{-1} \in F$ tale che $aa^{-1} = a^{-1}a = 1$*

2.2 Campi finiti della forma $GF(p^n)$

Teorema 2.2.1 (di esistenza e unicità dei campi finiti di ordine p^n). *Un campo di Galois K (finito) d'ordine q esiste se e solo se l'intero q è una potenza di un numero primo p, cioè:*

$$q = p^n$$

inoltre due campi finiti di eguale ordine sono necessariamente isomorfi.

Un campo finito di ordine p^n viene generalmente scritto come $GF(p^n)$, da Galois field. Vi sono due casi particolarmente interessanti per la crittografia. Per $n = 1$, si ha il campo finito $GF(p)$, questo campo viene definito come l'insieme $\mathbb{Z}p$ degli interi {0,1,...,p-1} insieme alle operazioni aritmetiche modulo p. Per $p = 2$ si ha il campo finito $GF(2^n)$, un campo molto importante in crittografia, in quanto:
Praticamente tutti gli algoritmi di crittografia, sia quelli simmetrici sia quelli a chiave pubblica, prevedono operazioni aritmetiche sugli interi. Se una delle operazioni utilizzate è la divisione, risulta necessario lavorare in un'aritmetica definita su un campo.Per comodità e per motivi di efficienza dell'implementazione si dovrebbe anche lavorare con interi rappresentabili esattamente con un determinato numero di bit. In pratica si vuole lavorare con interi compresi nell'intervallo da 0 a $2^n - 1$ rappresentabili con una parola di n bit. Quindi se devono essere utilizzate tutte le operazioni matematiche e si vuole rappresentare in n bit un intervallo completo di interi, l'aritmetica in modulo 2^n non funzionerà (perchè $\mathbb{Z}_2$ non è un campo). In modo equivalente, l'insieme di interi modulo 2^n per $n > 1$ non è un campo. Pertanto i campi finiti della forma $GF(2^n)$ sono i più interessanti per gli algoritmi crittografici.

2.3 Il campo dell'algoritmo AES: $GF(2^8)$

Ricordiamo che esiste un particolare isomorfismo:
$GF(2^n) \simeq \mathbb{Z}_2[x]/p(x) \quad \forall p(x)$ irriducibile $\in \mathbb{Z}_2[x]$ $deg(p(x)) = n$.

Nel caso di AES i creatori hanno deciso di utilizzare il campo finito

$$GF(2^8) \simeq \mathbb{Z}_2[x]/(x^8 + x^4 + x^3 + x + 1),$$

scegliendo il polinomio irriducibile $(x^8+x^4+x^3+x+1)$, anche se ci sono altri polinomi irriducibili di grado 8, ognuno dei quali porta a calcoli analoghi. La scelta di utilizzare questo particolare campo finito è stata fatta in quanto ogni elemento può essere rappresentato in modo unico come un polinomio
$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$
dove ogni b_i è 0 o 1. Poichè gli 8 bit $b_7b_6b_5b_4b_3b_2b_1b_0$ rappresentano un byte, è possibile rappresentare gli elementi di $GF(2^8)$ come byte di 8 bit.
Come addizione si utilizza lo XOR bit a bit, indicato con il simbolo $\oplus$. Ad esempio:

$$(X^7 + X^6 + X^3 + X + 1) + (X^4 + X^3 + 1) \rightarrow$$
$$11001011 \oplus 00011001 = 11010010 \rightarrow (X^7 + X^6 + X^4 + 1)$$

La moltiplicazione è più delicata e non ha un'interpretazione così semplice, ciò è dovuto al fatto che si sta lavorando modulo il polinomio $(X^8 + X^4 + X^3 + X + 1)$. In generale, si può moltiplicare per X mediante il seguente algoritmo:

1. Far scorrere i bit verso sinistra e concatenare uno 0 come ultimo bit.
2. Se il primo bit è 0, stop.
3. Se il primo bit è 1, eseguire uno XOR con 100011011, ovvero la rappresentazione binaria di $(X^8 + X^4 + X^3 + X + 1)$.

Il motivo per cui si ferma al passo 2 è che se il primo bit è 0, allora il polinomio è di grado inferiore a 8 dopo essere stato moltiplicato per X e quindi non ha bisogno di essere ridotto. Per moltiplicare le potenze maggiori di X, basta moltiplicare per X più volte.
La moltiplicazione per un polinomio arbitrario può essere eseguita moltiplicando per le varie potenze di X che appaiono nel polinomio e poi sommando

i vari risultati.
Vediamo ora un esempio:

$$(X^7 + X^6 + X^3 + X + 1)X = X^8 + X^7 + X^4 + X^2 + X$$
$$= (X^7 + X^3 + X^2 + 1) + (X^8 + X^4 + X^3 + X + 1)$$
$$= X^7 + X^3 + X^2 + 1 \, (mod \; X^8 + X^4 + X^3 + X + 1)$$

La stessa operazione con i bit diventa:

$$11001011 \rightarrow 110010110 \rightarrow 110010110 \oplus 10001101 = 010001101$$

In conclusione, si ha che le operazioni di addizione e moltiplicazione nel campo $GF(2^8)$ possono essere eseguite in modo molto efficiente.

2.4 Polinomi con coefficienti in $GF(2^8)$

Vediamo ora l'aritmetica che si utilizza per i polinomi di grado 3 o inferiore con coefficienti $GF(2^8)$:

- **L'addizione:** anche nei polinomi la somma viene eseguita sommando i coefficienti corrispondenti in $GF(2^8)$, quindi come si è gia detto, la somma diviene equivalente all'operazione XOR:

$$a(x) = a_3x^3 + a_2x^2 + a_1x + a_0$$

$$b(x) = b_3x^3 + b_2x^2 + b_1x + b_0$$

$$a(x) + b(x) = (a_3 \oplus b_3)x^3 + (a_2 \oplus b_2)x^2 + (a_1 \oplus b_1)x + (a_0 \oplus b_0)$$

- **La moltiplicazione:** viene eseguita come una normale moltiplicazione di polinomi con due differenze:

  1. I cointi vengono moltipicati in $GF(2^8)$
  2. Il polinomio risultante viene ridotto modulo $(x^4 + 1)$

Si denota il prodotto modulare di $a(x)b(x)$ con $a(x) \otimes b(x)$.
Per calcolare $d(x) = a(x) \otimes b(x)$, il primo passo è quello di eseguire una moltiplicazione senza l'operazione di modulo e raccogliere i coefficienti di pari potenza. Si può esprimere come $c(x) = a(x)\ X\ b(x)$. Allora:

$$c(x) = c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0$$

dove:

$$c_0 = a_0 \bullet b_0$$

$$c_1 = (a_1 \bullet b_0) \oplus (a_0 \bullet b_1)$$

$$c_2 = (a_2 \bullet b_0) \oplus (a_1 \bullet b_1) \oplus (a_0 \bullet b_2)$$

$$c_3 = (a_3 \bullet b_0) \oplus (a_2 \bullet b_1) \oplus (a_1 \bullet b_2) \oplus (a_0 \bullet b_3)$$

$$c_4 = (a_3 \bullet b_1) \oplus (a_2 \bullet b_2) \oplus (a_1 \bullet b_3)$$

$$c_5 = (a_3 \bullet b_2) \oplus (a_2 \bullet b_3)$$

$$c_6 = a_3 \bullet b_3$$

Il passaggio finale consiste nell'esecuzione dell'operazione di riduzione modulare:

$$d(x) = c(x) mod (x^4 + 1)$$

Quindi $d(x)$ deve soddisfare l'equazione:

$$c(x) = [(x^4 + 1) \times q(x)] \oplus d(x)$$

Capitolo 3

Descrizione dell'algoritmo AES

3.1 La cifratura AES

La proposta Rijndael per AES definiva una cifratura in cui le dimensioni del blocco e della chiave potevano essere specificate in modo indipendente a 128, 192 e 256 bit. Le specifiche di AES prevedono per la chiave queste tre dimensioni alternative, ma vincolano la lunghezza del blocco a 128 bit. Nella descrizione che seguirà presumeremo che la chiave sia a 128 bit, che è probabilmente quella implementata più frequentemente. L'algoritmo Rijndael è stato progettato per avere le seguenti caratteristiche:

- Resistenza contro tutti gli attacchi noti.
- Velocità e compattezza del codice su un'ampia gamma di piattaforme.
- Semplicità progettuale.

Vediamo ora come è strutturato l'algoritmo:
L'input dell'algoritmo è un singolo blocco di 128 bit. Nel documento FIPS PUB 197 questo blocco è rappresentato come una matrice quadrata di bytes. Questo blocco viene copiato nell'array *State* che viene modificato in ogni fase della crittografia. Dopo la fase finale *State* viene copiato in una matrice di output. Analogamente, la chiave di 128 bit è rappresentata come una matrice

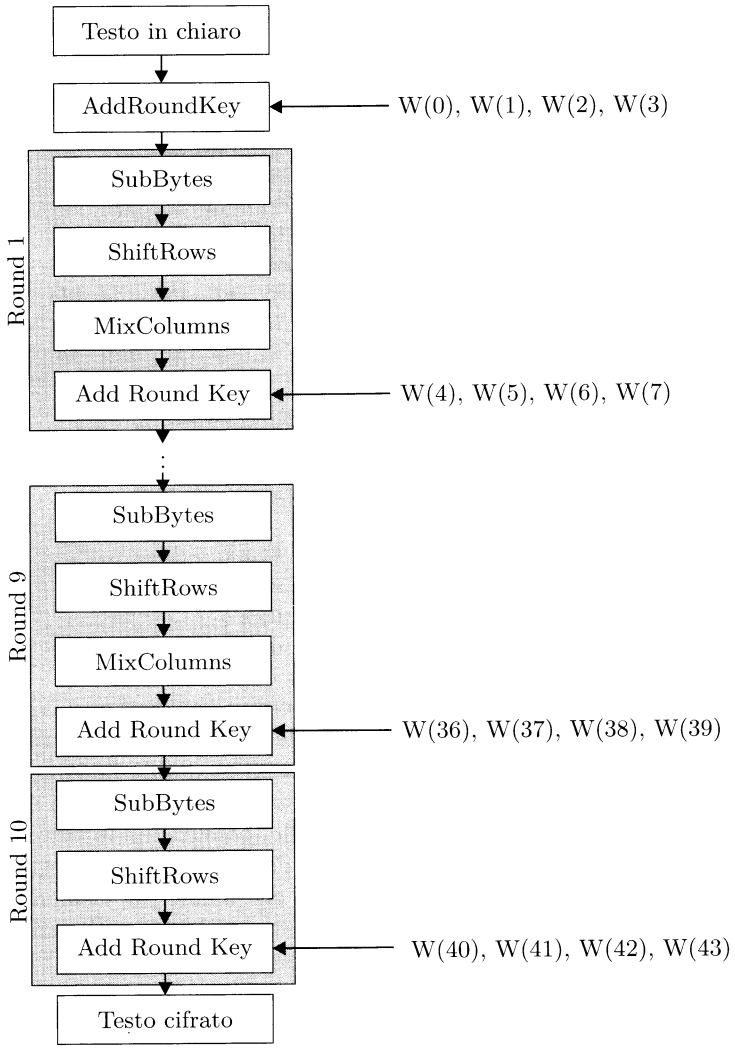

Figura 3.1: L'algoritmo AES

quadrata di bytes. La chiave fornita come input viene espansa in un array di 44 word a 32 bit.
La codifica consiste in 10 *round*(12 se la chiave è di 192, 14 se la chiave è di 256). Ogni *round* inizia con un input di 128 bit e finisce con un output di 128 bit. I *round* sono formati dai seguenti quattro passi fondamentali chiamati ***layer***:

1. **Trasformazione SubBytes**
2. **Trasformazione ShiftRows**
3. **Trasformazione Mix Columns**
4. **AddRoundKey**

Pertanto ogni round è dato da

$$\rightarrow [SubBytes] \rightarrow [ShiftRows] \rightarrow [MixColumns] \rightarrow [AddRoundKey] \rightarrow$$

L'unica eccezione dei 10 round è il *round* finale che non utilizza Mix Columns. Vedremo più avanti che esiste anche un round preliminare detto *round* 0 in cui si utilizza solo AddRoundKey.

3.1.1 Trasformazione SubBytes

La *Trasformazione SubBytes* è una semplice ricerca su tabella. AES definisce una matrice di 16x16 byte chiamata *S-box* che contiene una permutazione di tutti i 256 valori a 8 bit, come possiamo vedere nella figura 3.1.

Ogni singolo byte di *state* viene mappato in un nuovo byte nel seguente modo: i primi 4 bit indicano la riga e i secondi 4 bit indicano la colonna. Questi valori di riga e colonna fungono da indici della S-box per selezionare un valore univoco di output a 8 bit.

La S-box viene costruita nel seguente modo:

		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	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
x	**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	6D	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	9E
	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

Tabella 3.1: S-box

1. Inizializzare la S-box con il valore dei byte in una sequenza ascendente riga per riga. La prima riga contiene {00},{01},{02},...,{0F}; la seconda riga contiene {10},{11} e così via. Pertanto il valore del byte nella riga x e colonnna y sarà $\{xy\}$.

2. Associare a ciascun byte della S-box il suo inverso moltiplicativo nel campo finito $GF(2^8)$; il valore {00} viene mappato su se stesso.

3. Considerare che ciascun byte della S-box è costituito da 8 bit con $(b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0)$. Applicare la seguente trasformazione a ciascun bit di ciascun byte della S-box:

$$b'_i = b_i \oplus b_{(i+4)mod8} \oplus b_{(i+5)mod8} \oplus b_{(i+6)mod8} \oplus b_{(i+7)mod8} \oplus c_i$$

dove c_i è l' i-esimo bit del byte c con il valore {63}; cioè $(c_7c_6c_5c_4c_3c_2c_1c_0) = (01100011)$

Lo standard AES rappresenta questa trasformazione in forma matriciale nel seguente modo:

$$\begin{bmatrix} \boldsymbol{b'_0} \\ \boldsymbol{b'_1} \\ \boldsymbol{b'_2} \\ \boldsymbol{b'_3} \\ \boldsymbol{b'_4} \\ \boldsymbol{b'_5} \\ \boldsymbol{b'_6} \\ \boldsymbol{b'_7} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{b_0} \\ \boldsymbol{b_1} \\ \boldsymbol{b_2} \\ \boldsymbol{b_3} \\ \boldsymbol{b_4} \\ \boldsymbol{b_5} \\ \boldsymbol{b_6} \\ \boldsymbol{b_7} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Per il modo in cui è stato generata la S-box, possiamo anche vedere questa trasformazione come:

$$X \to AX^{-1} + B = AX^{254} + B.$$

Inoltre un fatto molto importante è che questa è l'unica trasformazione non lineare di tutto l'algoritmo.

3.1.2 Trasformazione ShiftRows

La trasformazione ShiftRows diretta consiste semplicemente nell'eseguire uno scorrimento di byte nell'arry *State*, ovvero:

- La prima riga *State* non viene modificata.
- Nella seconda riga viene eseguito uno scorrimento circolare a sinistra di un byte.
- Nella terza riga viene eseguito uno scorrimento circolare a sinistra di 2 byte.
- Nella quarta riga viene eseguito uno scorrimento circolare a sinistra di 3 byte.

Questa fase è più importante di quanto possa sembrare a prima vista per il fatto che *State*, come la cifratura input e output, viene trattato come array di 4 colonne di 4 byte. Pertanto, nella critttografia, i primi 4 byte del testo in chiaro vengono copiati nella prima colonna di *State*. Quindi uno scorrimento di riga sposta un singolo byte da una colonna a un'altra con una distanza lineare multipla di 4 byte.

3.1.3 Trasformazione Mix Columns

La trasformazione Mix Columns opera su ogni singola colonna. Ciascun byte di una colonna viene mappato in un nuovo valore che è una funzione dei 4 byte presenti nella colonna. La trasformazione può essere definita dalla seguente moltiplicazione di matrici su *State*:

$$\begin{bmatrix} \mathbf{02} & \mathbf{03} & \mathbf{01} & \mathbf{01} \\ \mathbf{01} & \mathbf{02} & \mathbf{03} & \mathbf{01} \\ \mathbf{01} & \mathbf{01} & \mathbf{02} & \mathbf{03} \\ \mathbf{03} & \mathbf{01} & \mathbf{01} & \mathbf{02} \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} \boldsymbol{s'_{0,0}} & \boldsymbol{s'_{0,1}} & \boldsymbol{s'_{0,2}} & \boldsymbol{s'_{0,3}} \\ \boldsymbol{s'_{1,0}} & \boldsymbol{s'_{1,1}} & \boldsymbol{s'_{1,2}} & \boldsymbol{s'_{1,3}} \\ \boldsymbol{s'_{2,0}} & \boldsymbol{s'_{2,1}} & \boldsymbol{s'_{2,2}} & \boldsymbol{s'_{2,3}} \\ \boldsymbol{s'_{3,0}} & \boldsymbol{s'_{3,1}} & \boldsymbol{s'_{3,2}} & \boldsymbol{s'_{3,3}} \end{bmatrix}$$

Ciascun elemento nella matrice è somma dei prodotti degli elementi di una riga e una colonna.
La trasformazione Mix Columns su un'unica colonna j ($0 \leq\ j \leq\ 3$) di *State* può essere espressa come:

$$s'_{0,j} = (2 \bullet s_{0,j}) \oplus (3 \bullet s_{1,j}) \oplus s_{2,j} \oplus s_{3,j}$$

$$s'_{1,j} = s_{0,j} \oplus (2 \bullet s_{1,j}) \oplus (3 \bullet s_{2,j}) \oplus s_{3,j}$$

$$s'_{2,j} = s_{0,j} \oplus s_{1,j} \oplus (2 \bullet s_{2,j}) \oplus (3 \bullet s_{3,j})$$

$$s'_{3,j} = (3 \bullet s_{0,j}) \oplus s_{1,j} \oplus s_{2,j} \oplus (2 \bullet s_{3,j})$$

Dove $\bullet$ si utilizza per indicare la moltiplicazione sul campo finito $GF(2^8)$ e $\oplus$ per indicare l'operazione di XOR bit a bit che corrisponde alla somma in $GF(2^8)$.

Il documento AES decrive anche un altro modo per caratterizzare la trasformazione MixColumns, in termini di aritmetica polinomiale. Nello standard, la trasformazione MixColumns viene definita considerando ciascuna colonna di *State* come un polinomio di quattro termini con coefficienti in $GF(2^8)$. Ciascuna colonna viene moltiplicata modulo $(x^4 + 1)$ per il polinomio fisso $a(x)$ dato da:

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$$

3.1.4 AddRoundKey

Nella trasformazione Add Round Key diretta, i 128 bit di *State* vengono sottoposti a uno XOR bit a bit con i 128 bit della chiave della fase, si tratta di un'operazione a colonne fra i 4 byte della colonna e una word della chiave di fase, in altre parole una somma vettoriale in $\mathbb{Z}_2^{128}$. Questa fase è molto semplice e agisce su tutti i bit di *State*. La sicurezza è garantita dalla complessità dell'espansione della chiave di fase e dalla complessità degli altri stadi di AES.

3.1.5 Chiavi di round

La chiave originale è formata da 128 bit, disposti in una matrice di byte 4x4. Questa matrice è ampliata aggiungendo 40 nuove colonne. Esse sono generate ricorsivamente a partire dalle prime 4 colonne $W(0), W(1), W(2), W(3)$. Se le colonne fino a $W(i-1)$ sono già state definite, allora, se i non è un multiplo di 4 si pone:

$$W(i) = W(i-4) \oplus W(i-1),$$

se i è multiplo di 4, invece:

$$W(i) = W(i-4) \oplus T(W(i-1)),$$

dove $T(W(i-1))$ è la trasformazione di $W(i-1)$ ottenuta nel seguente modo. Gli elementi a, b, c, d della colonna $W(i-1)$ vengono fatti scorrere ciclicamente in modo da avere b, c, d, a. Poi ognuno di questi byte viene sostituito

con il corrispondente elemento nell'S-box del passo SubBytes, ottenendo così 4 byte e, f, g, h. Infine, si calcola la costante di round:

$$r(i) = 00000010^{(i-4)/4}$$

in $GF(2^8)$. Allora $T(W(i-1))$ è il vettore colonna:

$$(e \oplus r(i), f, g, h)$$

La chiave di round per i-esimo round è formata dalle colonne:

$$W(4i), \quad W(4i+1), \quad W(4i+2), \quad W(4i+3).$$

3.2 La decifrazione di AES

Ognuno dei passi SubBytes,ShiftRows, MixColumns e AddRoundKey è invertibile.

1. L'inverso di SubBytes, chiamato InvSubBytes è un'altra tabella che implementa la permutazione inversa.

2. L'inverso di ShiftRows, chiamato InvShiftRows, si ottiene facendo scorrere le righe a destra invece che a sinistra.

3. L'inverso di Mix Columns esiste perchè la matrice 4x4 usata in MixColumns è invertibile. La trasformazione InvMixColumns è data dalla moltiplicazione per la matrice

$$\begin{pmatrix} 00001110 & 00001011 & 00001101 & 00001001 \\ 00001000 & 00001110 & 00001011 & 00001101 \\ 00001101 & 00001001 & 00001110 & 00001011 \\ 00001011 & 00001101 & 00001001 & 00001110 \end{pmatrix}$$

4. AddRoundKey coincide con il suo inverso

La decifrazione ha essenzialmente la medesima struttura della cifratura, dove però i vari *Layer* sono sostituiti dai loro inversi. Naturalmente, le chiavi di round sono usate nell'ordine inverso, cosichè il primo passo utilizza la chiave di round del decimo round e l'ultimo passo usa la chiave dello 0-esimo round.
Quanto precede mostra perchè il passo MixColumns non compare nell'ultimo round. Se comparisse, allora la cifratura partirebbe con AddRoundKey, SubBytes, ShiftRows, MixColumns, AddRoundKey,.... e finirebbe con AddRoundKey, SubBytes, ShiftRows, MixColumns, AddRoundKey. Quindi l'inizio della decifrazione sarebbe: InvMixColumns, InvAddRoundKey, InvSubBytes, InvShiftRows,.... in quanto per motivi di efficienza sono stati scambiati i due passi InvMixColumns e InvAddRoundKey. Questo significa che la decifrazione avrebbe all'inizio un passo InvMixColumns inutile con l'unico effetto di rallentare l'algoritmo.
Sui processori a 8 bit, la decifrazione non è così veloce come la cifratura. Ciò è dovuto al fatto che gli elementi della matrice 4x4 di InvMixColumns sono più complessi di quelli della matrice MixColumns e questo è sufficiente a rendere la decifrazione più lunga della cifratura di circa il 30%.
Tuttavia, in molte applicazioni, la decifrazione non è necessaria, come per esempio quando si usa la modalità CounTeR(CTR).
Vediamo più nel dettaglio questa modalità operativa:

3.2.1 Modalità CTR

In questa modalità di funzionamento viene impiegato un contatore corrispondente alle dimensioni del blocco di testo in chiaro. L'unico requisito è che il valore del contatore debba essere differente per ciascun blocco di testo in chiaro crittografato. In genere il contatore viene inizializzato con un determinato valore e poi incrementato di un'unità per ogni blocco successivo. Per la crittografia, il contatore viene crittografato e poi viene applicato uno XOR con il blocco di testo in chiaro per produrre il blocco di testo cifrato; non vi è alcun concatenamento. Per la decrittografia viene utilizzata

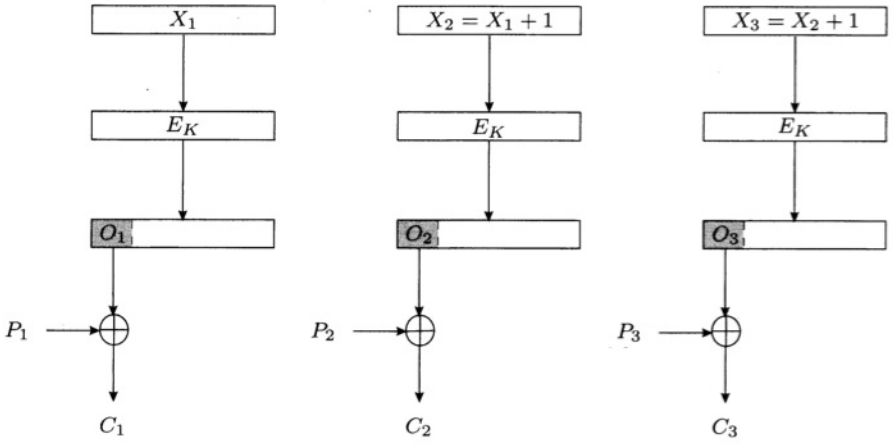

Figura 3.2: La modalità CTR

la stessa sequenza di valori del contatore e a ciascun contatore crittografato viene applicato lo XOR con un blocco di testo cifrato per ottenere il blocco di testo in chiaro corrispondente. Tutto questo quindi permette di utilizzare solo l'implementazione dell'algoritmo di crittografia e non dell'algoritmo di decrittografia, fatto molto interessante quando l'algoritmo di decrittografia è più oneroso di quello di crittografia, proprio come accade in AES.
Vediamo in generale la procedura per la modalità CounTeR:

$$X_j = X_{j-1} + 1$$

$$O_j = (E_K(X_j))$$

$$C_j = P_j \oplus O_j$$

per $j = 2, 3, ...$

Capitolo 4

Considerazioni sulla sicurezza

4.1 L'effettiva sicurezza

Dopo aver visto come agisce l'algoritmo AES, possiamo pensare che esso sia veramente sicuro?
Molti crittografi sono preoccupati che l'algoritmo sia troppo esposto ad attacchi per via dell'estrema semplicità algebrica. Daemen e Rijmen hanno osservato che l'azione del S-box su i bytes b_{ij} può essere scritta come:

$$S(b_{ij}) = \lambda_8 + \sum_{d=0}^{7} \lambda_d \; b_{ij}^{255-2d}$$

Questa semplicità è molto attraente per i crittoanalisti. Richard Schoeppel ha espresso preoccupazione per la struttura di Rijndael e per il fatto che l'algoritmo è quasi tutto lineare, ma la preoccupazione più grande riguarda il fatto che l'unica componente non lineare è x^{-1}, troppo banale, dato che il campo di base è di caratteristica 2. Spostando la costante λ_8 nel passo in cui viene sommata la chiave di round, l'equazione diventerebbe:

$$S(b_{ij}) = \sum_{d=0}^{7} \lambda_d \; b_{ij}^{-2d}$$

dove l'aritmetica è in $GF(2^8)$.
Gli altri passi di Rijndael hanno espressioni algebriche semplici. Niels Ferguson, Doug Whiting and Schroppel hanno proposto il seguente metodo per

attaccare Rijndael. Sia k_{ij}^r la chiave nella posizione (i, j) nel round r e, come prima, l'aritmetica in $GF(2^8)$. Dopodichè:

$$b_{ij}^{r+1} = k_{ij}^{(r)} + \sum_{e_r \in \varepsilon, d_r \in D} \lambda_{i,e_r,d_r} \ (b_{e_r,e_{r+1}}^{(r)})^{-2^{d_r}}$$

dove $\varepsilon = \{0, 1, 2, 3\}$ e $D = \{0, 1, 2, ..., 7\}$.
Consideriamo cosa accade con Rijndael a 128 bit, quando $r = 10$.
In generale ogni round della computazione raddoppia il numero dei termini. Ma la caratteristica 2 cambia tutto ciò. Elaborare gli elementi ad una potenza di 2 in un campo di caratteristica 2 causa la scomparsa di termini intermedi. In questo modo:

$$b_{ij}^3 = k_{ij}^{(2)} + \sum_{e_2 \in \varepsilon, d_2 \in D} \frac{\lambda_{i,e_2,d_2}}{(k_{e_2,e_2+j} + \sum_{e_1 \in \varepsilon, d_1 \in D} \frac{\lambda_{e_2,e_1,d_1}}{(b_{e_1,e_1+e_2+j})^{2^{d_1}}})^{2^{d_2}}}.$$

Togliendo ciò che non è essenziale e sostituendo i bytes della chiave estesa con K, le costanti con C, e i pedici e gli esponenti con gli asterischi, K^* C_* rappresenteranno i valori intermedi. In questo modo la formula diventerà più chiara. Dopo sei round, l'espressione prende la forma:

$$b_{ij}^6 = K^* + \sum_{e_5 \in \varepsilon, d_5 \in D} \frac{C_*}{K^* + \sum_{e_4 \in \varepsilon, d_4 \in D} \frac{C_*}{K^* + ...}}$$

Dopo sei round, ci sono 2^{35} termini $C_*/(K^* + p_*^*)$. Dopo dieci round di Rijndael, ci sono circa 2^{50} termini. Se si potessero risolvere efficacemente 2^{50} equazioni non lineari, sarebbe possibile rompere l'algoritmo Rijndael.

4.2 Big Encryption Standard

Sean Murphy e Matt Robshaw hanno preso una direzione diversa allo stesso approccio generale, loro erano in grado di mettere l'algoritmo di nuovo insieme. La loro revisione mostrò che era possibile un'altra direzione per un attacco. Poichè la stessa costante viene aggiunta ad ogni byte nel S-box,

mentre il resto delle funzioni del round consistono nello shiftare le righe, moltiplicare per una matrice in $GF(2^8)$ e sommare la chiave, Murphy e Robshaw suggerirono che le operazioni di round di Rijndael potessero essere raggruppate mettendo un'aggiunta, opportunamente modificata, all'interno dello step dell'addizione della chiave. Hanno pensato che questo arrangiamento fosse più naturale: con questo, il passo di diffusione lineare del S-box può essere parte di un layer di diffusione lineare. Murphy e Robshaw considerarono l'intero layer di diffusione lineare come $GF(2)$-lineare, dato da una matrice M, 128×128 in $GF(2)$. Notarono due cose sorprendenti su M. La prima fu il polinomio caratteristico di M: $c(x) = (x+1)^{128} = x^{128} + 1$, e il suo polinomio minimo, $m(x) = (x+1)^{15}$, entrambi particolarmente semplici. La seconda scoperta fu che $m(x)$ aveva un grado molto basso. Inoltre notarono che dopo sedici iterazioni la matrice M dava l'identità. Daemen e Rijmenm replicarono che non erano affatto sorpresi del fatto che $M^{16} = I$ e che la loro scelta era basata sulla semplicità dell'algoritmo senza comunque rinunciare alla sicurezza: le funzioni all'interno del round non vanno analizzate una ad una ma vanno viste nel complesso, la sicurezza non deriva dalla somma delle parti del round ma dalla funzione round che comprende tutte le funzioni al suo interno. Così Murphy and Robshaw proposero una versione avanzata di Rijndael. Sono due gli aspetti di Rijndael che aggiungono complessità nell'analizzare l'algoritmo: la non linearità del S- box e la diffusione del S-box introdotta dalla parte affine su $GF(2)$. Murphy e Robshaw videro un modo per incorporare AES in un sistema più grande che , fatta eccezione per l'inversione, avrebbe avuto tutti i passi lineari su $GF(2^8)$. Visualizzando lo spazio A della versione di AES a 128 bit come uno spazio vettoriale su $GF(2^8)$, definiamo 2 insiemi: B è un cifrario con un blocco a 128 byte e uno spazio delle chiavi che è uno spazio vettoriale di dimensione 128 su $GF(2^8)$, e B_A, un sottoinsieme di B che corrisponde ad A. L'insieme B sarà lo spazio di Big Encryption Standard(BES). Ci sono numerosi fatti importanti riguardanti questa nuova versione di AES :

- La diffusione lineare della matrice M_B di BES è sparsa e il suo polinomio

minimo è $(x+1)^{15}$, inoltre le operazioni su M_B sono in $GF(2^8)$.

- Non mostra particolari vulnerabilità alla crittoanalisi differenziale o lineare.

L'analisi precedente mostra che scoprire una chiave di AES è equivalente a risolvere un sistema di equazioni quadratiche in più variabili estremamente sparso (in cui molti coefficienti sono nulli). L'analisi di Murphy e Robshaw mostra che, con qualche ipotesi semplificatrice, una cifratura AES si può vedere come un sistema in più variabili di 2688 equazioni su $GF(2^8)$, delle quali 1280 sono quadratiche sparse e le altre sono lineri. Per questo motivo si potrebbe cercare di forzare AES risolvendo tali sistemi di equazioni purtroppo (o per fortuna) le tecniche efficienti per la risoluzione di sistemi non lineari (basi di Grobner) non sono in grado di risolvere sistemi con più di 15 variabili.

4.3 Obiettivi di progettazione particolari

Per i progettisti di Rijndael, la sicurezza era la preoccupazione primaria seguita immediatamente dall'efficienza. Ma Daemen e Rijmen cercavano anche una compenente che rese unico questo algoritmo: la semplicità , sia specifica sia di analisi. Nonostante la maggior parte dei crittografi non veda la semplicità come un fattore importante, essa fu la caratteristica che fece spiccare Rijndael rispetto agli altri algoritmi finalisti, che semplicemente avevano un design troppo complesso per essere analizzato fino in fondo. Un altro obiettivo non dichiarato dei creatori di Rijndael fu la trasparenza. Infatti utilizzarono una lista pubblica per scegliere i loro parametri, la stessa semplicità di $(x^8 + x^4 + x^3 + x + 1)$ e $(x^4 + 1)$ denota la voluta mancanza di parametri nascosti. Questa caratteristica ebbe la funzione di eliminare la possibilità di trovare delle trapdoor, che se ci fossero state sarebbero state individuate subito grazie proprio alla possibilità di analizzare a fondo tutto l'algoritmo.

Bibliografia

[1] Susan Landau, Polynomials in the Nationâs Service: Using Algebra to Design the Advanced Encryption Standard

[2] Federal Information Processing Standards Publication 197, Announcing the ADVANCED ENCRYPTION STANDARD (AES), November 26, 2001

[3] William Stallings, Crittografia e sicurezza delle reti, McGraw-

[4] Wade trappe - Lawrence C. Washington, Crittografia con elementi di teoria dei codici