Sistemi dell'elaborazione: Sicurezza su Reti **Anno Acc. 1998-99**
PROF. ALFREDO DE SANTIS

Advanced Encryption Standard

a cura di G. Di Donato, D. Morzillo, L. Picciuolo *15 Marzo 1999*

Indice

1 Advanced Encryption Standard

Fino ad oggi il DES [1] ha rappresentato lo standard per la cifratura e l'autenticazione di documenti. Approvato la prima volta dal National Institute of Standard and Technology (NIST) nel 1977, recentemente il DES è stato riaffermato quale standard di cifratura nel 1993, con una clausola che ne affermava la validità fino a Dicembre 1998. La clausola specificava inoltre che allo scadere di tale data il NIST avrebbe dovuto prendere in considerazione alternative che offrissero un maggior grado di sicurezza.

Sin dall'inizio, il DES è stato oggetto di molte critiche che hanno messo in discussione la sua forza crittografica. La prima è riferita alla sicurezza dello schema: a dispetto delle 2^{56} chiavi necessarie per avviare un attacco brutale nei confronti dell'algoritmo, la metà di esse permette in media la violazione del DES; ciò ha spinto alla considerazione che la lunghezza della chiave deve poter essere variabile affinchè sia assicurato un minimo grado di sicurezza ed affidabilità. Una seconda critica riguarda l'uso delle S-BOX nelle procedure di cifratura e decifratura [1]; in particolare si crede che ognuna di queste strutture nascondano delle trapdoor, ossia che i valori che le S-BOX contengono siano stati scelti in modo da nascondere qualche tranello che permetta all'ideatore di decifrare un qualunque messaggio.

In vista della scadenza di validità del DES, ed a cause di tali critiche, il NIST ha dovuto considerare delle alternative che offrissero un livello di sicurezza maggiore. Sebbene il DES triplo [3] poteva essere considerato un valido sostituto, il NIST desidera selezionare come nuovo standard un algoritmo di cifratura più sicuro ed efficiente. A tal fine, nel Settembre del 1997, è stato indetto un concorso pubblico per la nomina dell'Advanced Encryption Standard (A.E.S.) [2]. Obiettivo fondamentale del NIST è di stabilire un nuovo standard che diventi il punto di riferimento nel prossimo secolo nel campo della sicurezza.

Per ogni candidato alla nomina, il NIST ha richiesto un completo pacchetto di documentazione, che contenesse quanto segue:

1. una completa descrizione dell'algoritmo, comprendente tutte le equazioni matematiche, le tabelle, i diagrammi ed i parametri necessari all'implementazione dell'algoritmo;

2. una stima dell'efficienza computazionale, comprendente le seguenti informazioni:

   a. una descrizione della piattaforma usata per generare la stima;

   b. un'analisi delle velocità dell'algoritmo sulle suddette piattaforme ;

   c. qualunque altra informazione che possa rivelarsi essenziale per l'analisi dell'algoritmo;

3. un'analisi dell'algoritmo rispetto agli attacchi di crittoanalisi più conosciuti (ad esempio known o chosen plaintext);

4. un elenco dei vantaggi e dei limiti dell'algoritmo;

5. un'implementazione di riferimento per comprendere come l'algoritmo può essere codificato; in particolare il NIST richiede un codice sorgente scritto in ANSI C, con appropriati commenti;

6. un'implementazione ottimizzata matematicamente dell'algoritmo, implementata in ANSI C e Java, in modo che il NIST possa effettuare due diversi test per dimostrarne l'efficienza.

Ciascuno dei candidati verrà sottoposto ad una serie di test ed analisi atti a valutarne le caratteristiche computazionali e di sicurezza. I requisiti richiesti dal NIST sono i seguenti:

1. l'algoritmo deve implementare un cifrario a chiave simmetrica (segreta);
2. l'algoritmo deve appartenere alla classe dei block cipher;
3. l'algoritmo dovrà essere capace di supportare chiavi con taglia compresa tra **128** e **256** bit e messaggi di **128** bit (l'algoritmo può comunque supportare ulteriori taglie, quali ad esempio 64 e 256 bit, che verranno comunque prese in considerazione durante la fase di valutazione);
4. la struttura dell'algoritmo deve essere tale da permetterne l'implementazione su **smart-card**.

La piattaforma adoperata dal NIST per l'analisi dei candidati si basa su un PC IBM-compatibile con processore Pentium Pro 200MHz, 64MB di RAM e sistema operativo WINDOWS 95; i compilatori adoperati nella valutazione delle implementazioni sono il Borland C++ Development Suite 5.0 ed il Java Development Kit (JDK) 1.1.

I fattori che caratterizzano la selezione sono:

SICUREZZA. La sicurezza rappresenta il fattore più importante della valutazione. Ogni algoritmo, per essere definito minimamente sicuro, viene sottoposto alle seguenti prove:

1. la sicurezza dell'algoritmo verrà confrontata con la sicurezza degli altri candidati;
2. l'output dell'algoritmo dovrà essere indistinguibile da una permutazione casuale del blocco in input;
3. la sicurezza dovrà basarsi su una buona base matematica;
4. altri fattori di sicurezza presenti nell'algoritmo in analisi verranno considerati nella valutazione finale.

COSTO. L'algoritmo candidato dovrà avere insiti una serie di attributi che ne determinino un basso costo. Tale costo verrà valutato in base ai seguenti punti :

requisiti di licenza: Il NIST richiede che l'algoritmo selezionato come AES sia disponibile a livello mondiale, senza esclusive e royalty-free.

efficienza computazionale: la valutazione di tale efficienza sarà applicata sia all'implementazione hardware che software dell'algoritmo. Essenzialmente viene richiesta una certa velocità di computazione dei dati.

requisiti di memoria: vengono valutati il tasso dell'elaborazione dati a livello hardware, la grandezza del codice e infine la richiesta di RAM per l'implementazione software.

Il 20 Agosto 1998, in California, si è tenuta la prima conferenza per la candidatura all'AES, durante la quale il NIST ha annunciato i 15 candidati ufficiali (elencati nella tabella dell'APPENDICE B). Una seconda conferenza si terrà il 22 Marzo 1999 presso l'hotel Quirinale di Roma per discutere i risultati delle analisi e delle valutazioni del primo incontro.

Tutte le informazioni raccolte sono state opportunamente rese pubbliche [2], permettendo a chiunque di seguire attivamente la selezione dell'AES .

2 Crittosistema RC6

RC6 (progettato da Rivest, Robshaw, Sidney ed Yin) è un cifrario a blocchi semplice, compatto e sicuro. Questo nuovo crittosistema offre buone prestazioni ed una considerevole flessibilità, inoltre la sua semplicità permette rifiniture e miglioramenti alla sua sicurezza in modo facile e veloce.

L'idea per questo nuovo crittosistema nasce dalla considerazione che RC5 ([3], [4]) può essere considerato un potenziale candidato per la nomina all'AES.

Per andare incontro alle richieste del NIST riguardo le specifiche tecniche dell'AES, un cifrario a blocchi deve gestire blocchi di 128 bit. Poichè le architetture ed i linguaggi adottati dal NIST per la selezione del candidato non supportano operazioni a 64 bit, RC5 non poteva essere presentato in quanto non avrebbe raggiunto prestazioni soddisfacenti su tali architetture. Pertanto, lo schema è stato modificato opportunamente adoperando quattro registri, di 32 bit ciascuno, nelle operazioni di cifratura/decifratura. Il vantaggio che ne deriva è che, con un tale schema, eseguiamo due rotazioni in ogni round piuttosto che una in ogni half-round (come accade per l'RC5); inoltre nella computazione delle rotazioni viene usata una quantità di bit superiore, il che aumenta il grado di sicurezza dello schema.

Un'altra modifica apportata all'RC5 è l'aggiunta, all'insieme delle operazioni primitive, della moltiplicazione di interi; quest'ultima è adoperata per computare le rotazioni, in modo da renderle dipendenti da tutti i bit di un registro, piuttosto che da una quantità inferiore di bit.

Ciò permette all'RC6 di eseguire un numero inferiore di round, incrementando il grado di sicurezza e di efficienza.

2.1 Dettagli dell'algoritmo

Anche RC6 appartiene alla famiglia degli algoritmi di cifratura parametrizzati [3].

L'algoritmo cifra parole di w bit, eseguendo r iterazioni (o round), utilizzando una chiave di cifratura lunga b bytes. L'algoritmo considera quindi tre parametri fondamentali per il suo funzionamento: w, r, b.

Per rendere RC6 compatibile con le richieste per lo standard AES, la versione presentata fissa $w = 32$ e $r = 20$ ed utilizza una chiave di 16, 24 o 32 bytes.

RC6-$w/r/b$ opera su unità di quattro parole di w-bit usando le seguenti sei operazioni di base:

- $a + b$: addizione di interi modulo 2^w
- $a - b$: sottrazione di interi modulo 2^w
- $a \oplus b$: XOR bit a bit di parole di w bit
- $a \times b$: moltiplicazione di intero modulo 2^w
- $a \lll b$: shift di w bit a sinistra di un numero di bit dato dai $\lg w$ bit meno significativi di b
- $a \ggg b$: shift di w bit a destra di un numero di bit dato dai $\lg w$ bit meno significativi di b

2.1.1 Schedulazione della chiave

La schedulazione della chiave dell' RC6-$w/r/b$ è sostanzialmente identica a quella dell'RC5-$w/r/b$. La sola differenza consiste nel fatto che in RC6, dalla chiave fornita dall'utente, vengono dedotte un numero maggiore di parole chiave utilizzate nella cifratura/decifratura (vengono dedotte $(2r+4)$ parole chiave contro le $(2r+1)$ di RC5).

L'utente fornisce una chiave di b bytes, con $0 \leq b \leq 255$; tali bytes vengono caricati in un vettore L di c parole (oguna di w bit). In questo modo, in $L[0]$ viene memorizzato il primo byte della chiave ed $L[c-1]$ contiene, se necessari, dei bit nulli, aggiunti per rendere la dimensione della chiave uguale ad un certo numero di parole (se $b=0$ allora $c=1$ e quindi $L[0]=0$).
L'algoritmo di schedulazione genera $2r+4$ parole di w bit, che memorizza nel vettore $S[0, \cdot\cdot, 2r+3]$. L'algoritmo è il seguente:

$S[0] = P_w$
for $i = 1$ **to** $2r+3$ **do**
    $S[i] = S[i+1] + Q_w$
$A = B = i = j = 0$
$v = 3 \times \max(c, 2r+4)$
for $s = 1$ **to** v **do**
    $A = S[i] = (S[i] + A + B) \ll 3$
    $B = L[j] = (L[j] + A + B) \ll (A + B)$
    $i = (i+1) \bmod (2r+4)$
    $j = (j+1) \bmod c$

Le costanti esadecimali P_w e Q_w sono le stesse *costanti magiche* adoperate dall'RC5. In particolare risulta:

$$P_{32} = B7E15163h$$

e

$$Q_{32} = 9E3779B9h.$$

2.1.2 Algoritmi di cifratura e decifratura

L'algoritmo di cifratura prende in input il testo in chiaro, memorizzato nei quattro registri A, B, C, D, ognuno di w bit, il numero r di iterazioni da eseguire e la chiave, memorizzata in $S[0, \cdot\cdot, 2r+3]$ (vedi algoritmo di schedulazione della chiave) e restituisce il testo cifrato nei registri A, B, C, D.
L'algoritmo è il seguente:

$B = B + S[0]$;
$D = D + S[1]$;
for $i = 1$ **to** r **do**
    $t = (B \times (2B+1)) \ll \lg w$;
    $u = (D \times (2D+1)) \ll \lg w$;
    $A = ((A \oplus t) \ll u) + S[2i]$;
    $C = ((C \oplus u) \ll t) + S[2i+1]$;
    $(A, B, C, D) = (B, C, D, A)$;
$A = A + S[2r+2]$;
$C = C + S[2r+3]$;

In modo del tutto analogo, l'algoritmo di decifratura prende in input il testo cifrato, memorizzato nei quattro registri, e restituisce il testo in chiaro nei registri A, B, C, D.
L'algoritmo è il seguente:

$C = C - S[2r+3]$;
$A = A - S[2r+2]$;
for $i = r$ **downto** 1 **do**
    $(A, B, C, D) = (B, C, D, A)$;
    $u = (D \times (2D+1)) \ll \lg w$;
    $t = (B \times (2B+1)) \ll \lg w$;
    $C = ((C - S[2i+1]) \gg t) \oplus u$;
    $A = ((A - s[2i]) \gg u) \oplus t$;
$D = D - S[1]$;
$B = B - S[0]$;

2.2 Performance dello schema RC6

Uno dei requisiti per partecipare alla selezione per la nomina di Advanced Encryption Standard (AES) riguarda l'implementazione dell'algoritmo in diversi linguaggi di sviluppo, quali l'ANSI C ed il Java ed una implementazione che permetta l'utilizzo dell'algoritmo su architetture con processori dalle capacità computazionali limitate, ad esempio le smart-card. Analizziamo, nei successivi paragrafi, il comportamento computazionale delle implementazioni dell'algoritmo in ANSI C e Java, e come lo stesso possa essere adattato ad architetture con processore ad 8 bit senza che le proprietà di sicurezza ed efficienza vengano penalizzate.

2.2.1 Performance dello schema RC6 in ANSI C, Java e Assembly

Nella Tabella 1 riportiamo dei valori che evidenziano le caratteristiche computazionali dell'RC6 nei diversi linguaggi di sviluppo richiesti dal NIST. I valori riportati rappresentano una media dei risultati dei test effettuati su ogni implementazione dell'algoritmo (in particolare ogni test è stato eseguito 10 volte). In particolare, i valori di riferimento per le implementazioni ANSI C ed ASSEMBLY sono stati misurati su un elaboratore basato su processore PENTIUM II 266MHZ con 32MB di RAM e Windows 95 (durante l'esecuzione dei test, gli interrupt mascherabili sono stati disabilitati in modo da aumentare l'accuratezza dei risultati). I risultati riportati per quanto concerne l'implementazione JAVA sono state misurate su processore PENTIUM PRO 180MHZ con 64 MB di RAM e Windows NT 4.0. Inoltre sono stati riportati anche valori riferiti allo schema RC5-32/16/16, in modo da permettere un confronto tra i due crittosistemi.

	schema	cicli/block	block/sec a 200MHz	MByte/sec a 200MHz
ANSI C	RC6 encrypt	616	325000	5.19
ANSI C	RC6 decrypt	566	353000	5.65
Java (JDK)	RC6 encrypt	16200	12300	0.197
Java (JDK)	RC6 decrypt	16500	12100	0.197
Java (JIT)	RC6 encrypt	1010	197000	3.15
Java (JIT)	RC6 decrypt	955	209000	3.35
assembly	RC6 encrypt	254	787000	12.6
assembly	RC6 decrypt	254	788000	12.6
ANSI C	RC5-32/16/16 encrypt	328	610000	4.9
Java (JIT)	RC5-32/16/16 encrypt	1140	175000	1.4
assembly	RC5-32/16/16 encrypt	148	1350000	10.8

Tabella 1

I dati riportati nella Tabella 1 non sono comprensivi dei valori relativi alla schedulazione della chiave (mostrati nella Tabella 2) e sono indipendenti dalla lughezza della chiave fornita dall'utente. I dati riferiti all'implementazione ANSI C e JAVA sono ottenuti, rispettivamente, mediante una cifratura/decifratura di un blocco di 3000 e di 100000 caratteri, in modalità ECB.

	schema	cicli	μsec a 200MHz	Key setups/sec a 200MHz
ANSI C	RC6-32/20/16	4710	23.5	42500
Java (JDK)	RC6-32/20/16	107000	537	1860
Java (JIT)	RC6-32/20/16	14300	71.4	14000
ANSI C	RC6-32/20/24	4710	23.6	42400
Java (JDK)	RC6-32/20/24	108000	542	1840
Java (JIT)	RC6-32/20/24	14300	71.5	14000
ANSI C	RC6-32/20/32	4720	23.6	42400
Java (JDK)	RC6-32/20/32	110000	548	1820
Java (JIT)	RC6-32/20/32	15000	75.1	13300

Tabella 2

2.2.2 Performance dello schema RC6 su piattaforma ad 8 bit

L'algoritmo RC6 è adattabile su architetture con processore ad 8 bit; è infatti possibile definire le operazioni di base dell'algoritmo adoperando operazioni implementabili sulla maggior parte dei processori. Vediamo come sia possibile implementare RC6-32/20/b sul processore ad 8 bit della Intel MCS 51.

- **Cifratura/Decifratura.** Analizzando gli algoritmi descritti nel paragrafo precedente, e considerando che l'espressione

$$B \times (2B + 1) = 2B^2 + B$$

consiste di un quadrato e di due addizioni, ogni round dell'algoritmo di cifratura esegue sei addizioni, due OR esclusivi (XOR), due quadrati, due rotazioni a sinistra di 5 bit ($\log w = \log 32 = 5$) e di due rotazioni a sinistra di una quantità r variabile.
Tali operazioni possono essere implementate su un processore ad 8 bit come segue:

1. un'addizione a 32 bit può essere computata mediante 4 addizioni a 8 bit con riporto (ADDC);
2. un OR esclusivo (XOR) a 32 bit può essere computato con 4 XOR a 8 bit (XRL);
3. un quadrato a 32 bit può essere computato mediante 6 moltiplicazioni di stringhe di 8 bit (MUL) ed 11 addizioni con riporto (ADDC). In questo caso le sei moltiplicazioni sono sufficienti dato che necessitiamo soltanto dei 32 bit meno significativi di tale prodotto.
4. una rotazione ciclica a sinistra di 5 bit su una stringa di 32 bit (vedi Figura 1) può essere computata attraverso una rotazione a destra di un bit su una stringa di 32 bit, eseguita 3 volte, e permutandone i 4 byte risultanti (vedi Figura 2). La rotazione a destra di un bit su stringhe a 32 bit può essere implementata mediante 4 rotazioni ad 8 bit (RRC: vedi Figura 3).
5. una rotazione ciclica a sinistra di r bit su una stringa di 32 bit può essere computata con una rotazione a sinistra o a destra di un bit, eseguita $\overline{r}$ volte (con $\overline{r} \leq 4$ ottenuto dai 5 bit meno significativi di r), seguita da una permutazione dei 4 bytes. Ogni permutazione può essere controllata mediante i salti (JB).

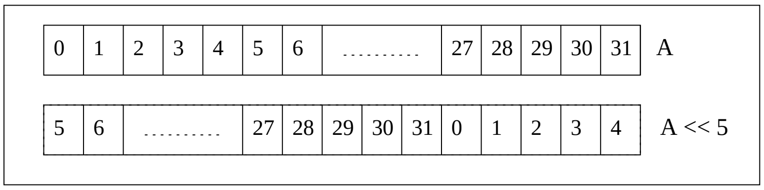

Figura 1: Rotazione a sinistra di 5 bit di una stringa di 32 bit.

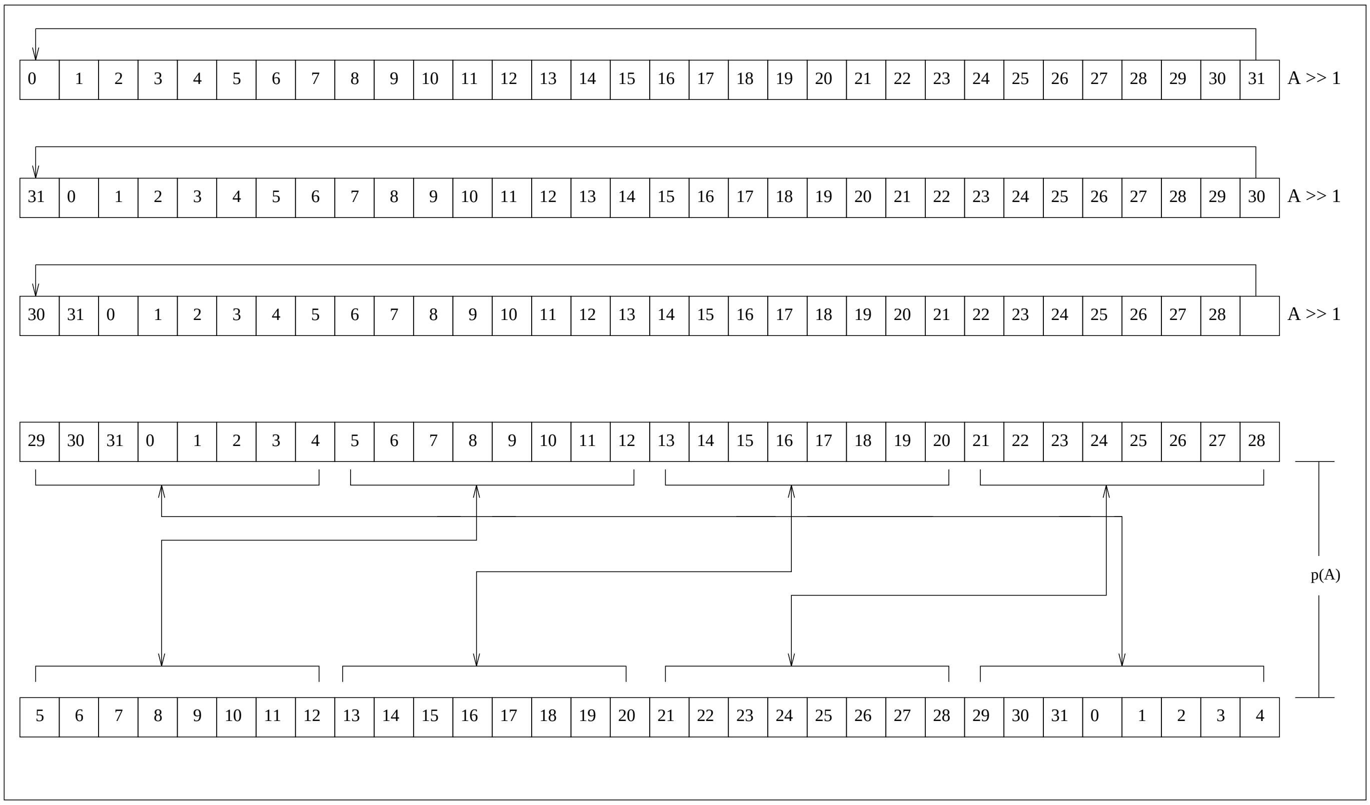

Figura 2: Rotazione a sinistra di 5 bit mediante l'esecuzione di tre rotazioni a destra di 1 bit su stringhe a 32 bit seguita da una opportuna permutazione p.

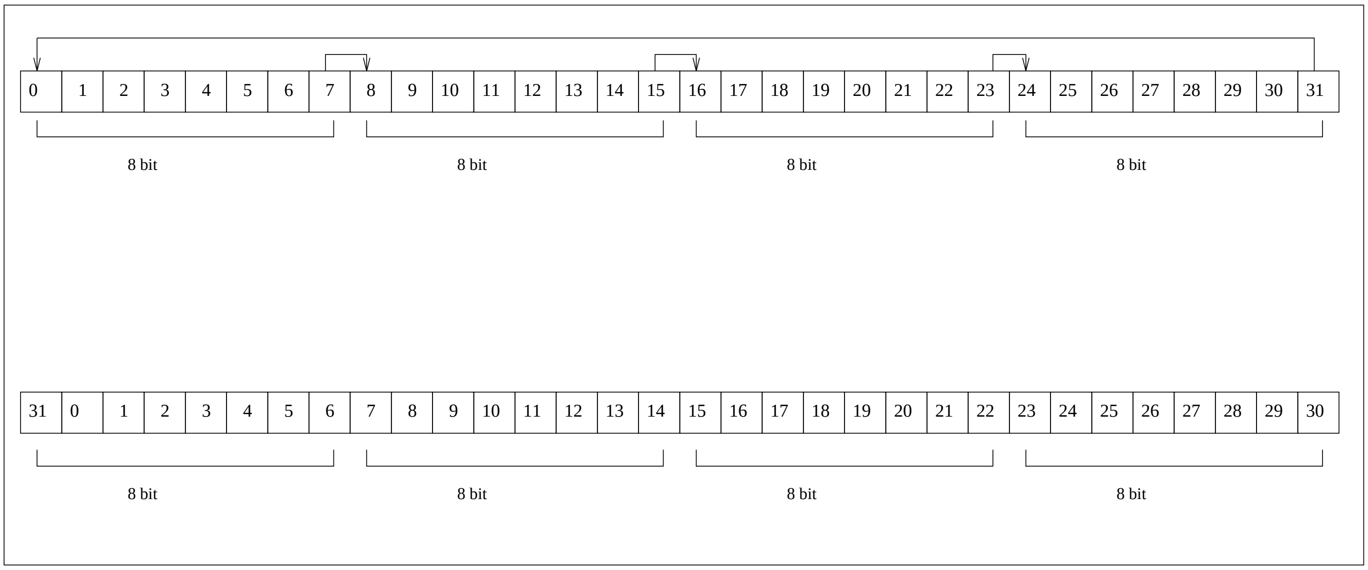

Figura 3: Rotazione a destra di 1 bit mediante rotazione a destra di 1 bit su stringhe di 8 bit con riporto (le freccie rappresentano i carry).

Tutte le istruzioni sopraelencate necessitano di 1 ciclo per essere eseguite, eccetto MUL e JB che richiedono, rispettivamente, 4 e 2 cicli.
Il numero totale di cicli necessari all'esecuzione di un round di RC6 è stimato nella seguente tabella:

Operazione	**Istruzioni**	**Cicli per operazioni**	**Contributo per ogni ciclo**
Addizione	4 ADDC	4	$4 \times 6 = 24$
XOR	4 XRL	4	$4 \times 2 = 8$
Radice quadrata	6 MUL, 11 ADDC	35	$35 \times 2 = 70$
Left Shift di 5	12 RRC	12	$12 \times 2 = 24$
Right Shift di r (media su r)	8 RRC o RLC, 8 JB	24	$24 \times 2 = 48$
Totale			174

Considerando overhead aggiuntivo, determinato dalle istruzioni di assegnazione, pre-whitening, post-whitening ed altre operazioni di inizializzazione, la cifratura di un blocco di dati richiede circa $(174 \times 20) \times 4 = 13920$ cicli. Poichè sul processore MCS 51 ogni ciclo è eseguito in un microsecondo, la velocità dell'algoritmo è circa $(1000000/13920) \times 128 = 9.2$ Kbit/sec.

Recentemente è stata sviluppata un'implementazione di RC6 per processori Intel 8051 che necessita di 13535 cicli per cifrare un blocco di dati.

- **Schedulazione della chiave.** La parte fondamentale della schedulazione della chiave è rappresentata dall'ultimo FOR nel codice dell'algoritmo (vedi paragrafo 2.1.1). Per $b = 16, 24, 32$, il numero di iterazioni di questo ciclo è $v = 3 \times \max(20 \times 2 + 4, b/4) = 132$, che è indipendente da b. In ogni iterazione del ciclo vengono effettuate quattro addizioni a 32 bit, una rotazione a sinistra di 3 bit ed una rotazione a sinistra di $t = (A + B)$ bit. Inoltre ci sono alcune operazioni ad 8 bit che vengono incluse come overheads. Seguendo un'analisi del tutto simile a quella fatta per la cifratura otteniamo, per ogni iterazione, un numero di cicli pari a 52. Pertanto sono necessari $(52 \times 132) \times 4 = 27456$ cicli per schedulare chiavi di 128, 192, 256 bit, impiegando, sul processore MCS 51, un tempo pari a circa 27 millisecondi.

2.3 Regole di implementazione

Come potremo aspettarci dall'algoritmo, RC6 è straordinariamente compatto. Infatti si stima per un processore Intel Pentium PRO che una veloce implementazione in linguaggio ASSEMBLY potrebbe essere scritta con meno di 256 bytes di codice per ognuna delle tre fasi (cifratura, decifratura e schedulazione della chiave). A differenza di molti altri algoritmi, l'RC6 non consulta tabelle durante la fase di cifratura. Ciò significa che il codice dell'algoritmo può essere prontamente adattato all'interno dei moderni chip di memoria cache ed ottenere così un notevole risparmio di spazio. Le funzioni di cifratura e decifratura di RC6 fanno uso dei 176 bytes della chiave schedulata, ed una minima quantità di memoria; per computare tali 176 bytes, il processo di schedulazione del crittosistema richiede poco più di un vettore ausiliare della stessa taglia della chiave fornita dall'utente. Inoltre, vista la dimensione non proibitiva della chiave schedulata, è possibile precomputare e memorizzare centinaia di chiavi schedulate per centinaia di chiavi fornite dall'utente. Allora fare riferimento ad una di queste chiavi richiederebbe soltanto lo spostamento di un puntatore alla chiave corrente. Dato che la famiglia degli algoritmi cui il crittosistema appartiene è completamente parametrizzata, e avendo la possibilità di implementare RC6 in maniera compatta ed efficiente, l'algoritmo risulta particolarmente versatile.

2.4 Progettazione e Motivazione

Durante la progettazione di RC6 si tenne conto soprattutto dei seguenti requisiti:

1. Sicurezza
2. Semplicità
3. Buone prestazioni computazionali

2.4.1 Sicurezza e semplicità di RC6

La semplicità di RC5 ha reso tale algoritmo un punto di attrazione per la ricerca. RC6 è stato progettato sull'esperienza ottenuta da RC5 e sulla sicurezza offerta da un cifrario di notevole semplicità. Possiamo vedere la progressiva progettazione dell'RC6 attraverso i seguenti passi:

1. Partiamo dal ciclo base di RC5:

$$\begin{array}{l} \textbf{for } i = 1 \textbf{ to } r \textbf{ do} \\ \quad A = ((A \oplus B) \lll B) + S[i] \\ \quad (A, B) = (B, A) \end{array}$$

2. Eseguiamo due copie di RC5 in parallelo: una sui registri A e B, l'altra sui registri C e D.

$$\begin{array}{l} \textbf{for } i = 1 \textbf{ to } r \textbf{ do} \\ \quad A = ((A \oplus B) \lll B) + S[2i] \\ \quad C = ((C \oplus D) \lll D) + S[2i+1] \\ \quad (A, B) = (B, A) \\ \quad (C, D) = (D, C) \end{array}$$

3. Ad ogni istruzione di swap, invece di scambiare A con B e C con D, permutiamo i registri nel seguete modo: $(A, B, C, D) = (B, C, D, A)$;così la computazione di AB è unita con la computazione di CD. Il ciclo precedente diventa il seguente:

$$\begin{array}{l} \textbf{for } i = 1 \textbf{ to } r \textbf{ do} \\ \quad A = ((A \oplus B) \lll B) + S[2i] \\ \quad C = ((C \oplus D) \lll D) + S[2i+1] \\ \quad (A, B, C, D) = (B, C, D, A) \end{array}$$

4. Unendo ulteriormente le computazioni di AB con CD, attraverso lo scambio delle due quantità che definiscono le due rotazioni, otteniamo quanto segue:

$$
\begin{aligned}
&\textbf{for } i = 1 \textbf{ to } r \textbf{ do}\\
&\quad A = ((A \oplus B) \lll D) + S[2i]\\
&\quad C = ((C \oplus D) \lll B) + S[2i+1]\\
&\quad (A, B, C, D) = (B, C, D, A)
\end{aligned}
$$

5. Piuttosto che usare B e D in maniera semplice, come nel ciclo sopra descritto, usiamo delle versioni di questi registri, trasformate in maniera opportuna. Per ottenere maggiore sicurezza il numero di rotazioni data-dependent, derivato dal risultato di queste trasformazioni, dovrebbe dipendere da tutti i bit, opportunamente mixati, della parola in input. La particolare scelta di queste trasformazioni per RC6 è la funzione

$$f(x) = (x \times (2x+1)) \bmod 2^w \tag{1}$$

seguita da una rotazione a sinistra di cinque posizioni. Questa trasformazione va incontro ai nostri obiettivi di sicurezza, sfruttando operazioni che sono implementate efficientemente sulla maggior parte dei moderni processori. Poichè $f(x)$ è una corrispondenza biunivoca modulo 2^w il numero di rotazioni eseguite dipende da tutti i bit di x. Otteniamo il seguente ciclo:

$$
\begin{aligned}
&\textbf{for } i = 1 \textbf{ to } r \textbf{ do}\\
&\quad t = (B \times (2B+1)) \lll 5;\\
&\quad u = (D \times (2D+1)) \lll 5;\\
&\quad A = ((A \oplus t) \lll u) + S[2i];\\
&\quad C = ((C \oplus u) \lll t) + S[2i+1];\\
&\quad (A, B, C, D) = (B, C, D, A);
\end{aligned}
$$

6. All'inizio ed alla fine delle r iterazioni aggiungiamo delle pre-inizializzazioni e post-inizializzazioni. Senza tali istruzioni, il testo in chiaro rivela parte di input durante la prima iterazione di cifratura ed il testo cifrato rivela parte dell'input nell'ultima iterazione di cifratura. Le due istruzioni di pre-inizializzazione e post-inizializzazione mascherano tali valori; otteniamo così il seguente codice:

$$
\begin{aligned}
&B = B + S[0];\\
&D = D + S[1];\\
&\textbf{for } i = 1 \textbf{ to } r \textbf{ do}\\
&\quad t = (B \times (2B+1)) \lll 5;\\
&\quad u = (D \times (2D+1)) \lll 5;\\
&\quad A = ((A \oplus t) \lll u) + S[2i];\\
&\quad C = ((C \oplus u) \lll t) + S[2i+1];\\
&\quad (A, B, C, D) = (B, C, D, A);\\
&A = A + S[2r+2];\\
&C = C + S[2r+3];
\end{aligned}
$$

L'algoritmo così ottenuto rispecchia i tre obiettivi preposti, ovvero semplicità, sicurezza ed affidabilità.

2.4.2 Prestazioni computazionali di RC6

Fin dalla prima pubblicazione di RC5, ci sono state una serie di articoli che hanno prodotto sostanziali progressi nell'analisi, e le ultime tecniche dimostrano che RC5-32/12/b non può essere idoneo per la sicurezza a lungo termine. La maggior parte delle critiche alla crittoanalisi di RC5 riguardano lo scambio tra i round. L'addizione intera, introdotta in RC6 fornisce un numero ragionevole di scambi (grazie soprattutto al riporto), in modo da ottenere un più elevato livello di protezione, anche in caso di un basso numero di iterazioni.
I cambiamenti significativi introdotti nell'RC6 riguardano l'introduzione della funzione quadratica $B \times (2B+1)$ e la rotazione fissa con 5 bit; la funzione della trasformazione quadratica dei valori di B e D, usata al posto di B e D, per modificare i registri A e C incrementa la non linearità dello schema senza comportare nessuna perdita di entropia, dal momento che la trasformazione è una permutazione.
La rotazione fissa con 5 bit gioca un semplice ma importante ruolo nel complicare sia la crittoanalisi lineare che quella differenziale.

2.4.3 Sicurezza di RC6

Il miglior approccio (in riferimento ad RC6-32/20/b) che un crittoanalista ha a disposizione per attaccare RC6 consiste in una ricerca esaustiva della chiave di cifratura di b bytes (o sul vettore $S[0,\cdot\cdot,43]$ espanso, quando la chiave fornita dall'utente è particolarmente lunga). Per riuscire in questo intento sono richieste un numero di operazioni determinato dal min(2^{8b},2^{1408}).
Gli attacchi più avanzati di crittoanalisi differenziale [5] e lineare [6], ammissibili su versioni dell'algoritmo con un basso numero di iterazioni, non sono adatti ad attaccare versioni dell'RC6 con 20 iterazioni. La principale difficoltà nasce dal fatto che è difficile trovare caratteristiche iterative o buone approssimazioni lineari su cui basare l'attacco. In conclusione, sulla sicurezza dell'RC6 possiamo affermare quanto segue:

1. il miglior attacco sul cifrario risulta essere una ricerca esaustiva su tutti i possibili valori della chiave di cifratura fornita dall'utente;

2. la quantità di dati necessaria ad attacchi sofisticati, quale la crittoanalisi differenziale e lineare, supera di gran lunga la quantità di dati a disposizione;

3. non esistono esempi conosciuti che abbiano sfruttato le proprietà della *weak-key*.

3 Appendice A

in questo paragrafo mostriamo alcuni esempi dell'applicazione dell'RC6 a particolari testi da cifrare ed il corrispondente testo cifrato:

PLAINTEXT:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
USER KEY:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
CIPHERTEXT:	8f c3 a5 36 56 b1 f7 78 c1 29 df 4e 98 48 a4 1e
PLAINTEXT:	02 13 24 35 46 57 68 79 8a 9b ac bd ce df e0 f1
USER KEY:	01 23 45 67 89 ab cd ef 01 12 23 34 45 56 67 78
CIPHERTEXT:	52 4e 19 2f 47 15 c6 23 1f 51 f6 36 7e a4 3f 18
PLAINTEXT:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
USER KEY:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
	00 00 00 00 00 00 00 00
CIPHERTEXT:	6c d6 1b cb 19 0b 30 38 4e 8a 3f 16 86 90 ae 82
PLAINTEXT:	02 13 24 35 46 57 68 79 8a 9b ac bd ce df e0 f1
USER KEY:	01 23 45 67 89 ab cd ef 01 12 23 34 45 56 67 78
	89 9a ab bc cd de ef f0
CIPHERTEXT:	68 83 29 d0 19 e5 05 04 1e 52 e9 2a f9 52 91 d4
PLAINTEXT:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
USER KEY:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
CIPHERTEXT:	8f 5f bd 05 10 d1 5f a8 93 fa 3f da 6e 85 7e c2
PLAINTEXT:	02 13 24 35 46 57 68 79 8a 9b ac bd ce df e0 f1
USER KEY:	01 23 45 67 89 ab cd ef 01 12 23 34 45 56 67 78
	89 9a ab bc cd de ef f0 10 32 54 76 98 ba dc fe
CIPHERTEXT:	c8 24 18 16 f0 d7 e4 89 20 ad 16 a1 67 4e 5d 48

4 Appendice B

Nella seguente tabella sono esposti tutti i candidati alla nomina di A.E.S.:

Nome dell'algoritmo	*Proponente*
CAST-256	Entrust Technologies,INC.
CRYPTON	Future System, INC.
DEAL	R. Outerbridge, L.Knudsen
DFC	CNRS
E2	Nippon Telegraph and Telephone Corp.
FROG	TecApro Internacional S.A.
HPC	Rich Schroeppel
LOKI97	L.Brown, J.Pieprzyk, J.Seberry
MAGENTA	Detsche Telekom AG
MARS	IBM
RC6	RSA Laboratories
RIJNDAEL	J.Daemen, V. Rijmen
SAFER+	Cylink Corp.
SERPENT	R.Anderson, E. Biham, L. Knudsen
TWOFISH	B.Schneider,J.Kelsey,D.Whiting,D.Wagner,C.Hall,N.Ferguson

Tabella 1: Candidati alla nomina di Advanced Encryption Standard(AES)

Riferimenti

[1] A. Tobiello, C. Pezzella. *Data Encryption Standard.*
Tesina per il corso di Sicurezza su Reti, Corso di Laurea in Informatica, Università di Salerno.

[2] Il bando di concorso e tutte le informazioni riguardanti l'AES sono reperibili presso il sito del NIST: www.nist.gov/AES.

[3] M. Coppola, O. Panza. *Cifrari a blocchi.*
Tesina per il corso di Sicurezza su Reti, Corso di Laurea in Informatica, Università di Salerno.

[4] R.L. Rivest. The RC5 encryption algorithm. In B. Preneel, editor, *Fast Software Encryption*, volume 1008 of *Lecture Notes in Computer Science*, pag. 86-96, 1995, Springer-Verlag.

[5] E. Biham, A. Shamir. *Differential Cryptanalysis of the Data Encryption Standard.* Springer-Verlag, New York, 1993.

[6] M. Matsui. Linear cryptanalysis method for DES cipher. In T. Helleseth, editor, *Advanced in Cryptology - Eurocrypt '93*, volume 765 of *Lecture Notes in Computer Science*, pag. 386-397, 1994, Springer-Verlag.