Prova in Itinere del 24 Aprile 2013

1) Definire formalmente il concetto di perfetta sicurezza

Sia $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ un cifrario simmetrico. Questo si definisce perfettamente sicuro se e solo se verifica la seguente proprietà:

$$\forall m_1, m_2 \in \mathcal{M}$$
$$\forall c \in \mathcal{C}$$
$$k \xleftarrow{R} \mathcal{K}$$
$$Pr[\mathcal{E}_k(m_0) = c] = Pr[\mathcal{E}_k(m_1) = c]$$

con $\mathcal{M} = plaintexts, \mathcal{C} = ciphertexts, \mathcal{K} = chiavi.$

In altre parole, presi due messaggi m_0 ed m_1 qualsiasi dall'insieme di tutti i plaintext validi, la probabilità che la funzione $\mathcal{E}_k$ produca c è la stessa. Questo tipo di sicurezza è garantito dal One Time Pad (OTP).

2) Sia $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ un cifrario simmetrico e siano M, K, C gli insiemi dei messaggi, delle chiavi e dei crittotesti, rispettivamente. Si considerino adesso i seguenti insiemi $M = \{1, 2, 3\}$ e $K = \{1, 2, 3\}$. Supponiamo di voler cifrare un solo messaggio $m \in M$, utilizzando una chiave (random) $k \in K$, come segue $c = (k \cdot m) \mod 7$. E' tale sistema sicuro in senso perfetto? Giustificare la risposta fornita.

Per essere sicuro in senso perfetto, un cifrario deve verificare la seguente proprietà:

$$\forall m_1, m_2 \in \mathcal{M}$$
$$\forall c \in \mathcal{C}$$
$$k \xleftarrow{R} \mathcal{K}$$
$$Pr[\mathcal{E}_k(m_0) = c] = Pr[\mathcal{E}_k(m_1) = c]$$

Il cifrario proposto **non** verifica questa proprietà, e lo si può provare con un semplice contro-esempio. Basta considerare il ciphertext 1: solo il messaggio 1 può produrre questo risultato tramite l'operazione $(1 \cdot 1) \mod 7$.

$$Pr[\mathcal{E}_k(1) = 1] \neq Pr[\mathcal{E}_k(2) = 1]$$
$$\frac{1}{3} \neq 0$$

3) Supponendo di avere a disposizione le procedure Espandi Chiave, S, Shift Rows e mix cols discusse a lezione (delle quali non è richiesta, in questa sede, la descrizione algoritmica) si descriva dettagliatamente il funzionamento di AES, fornendo, inoltre, lo pseudocodice dell’algoritmo.

```
AES_k(m):
    (k0, k1, ... k10) <- EspandiChiave(k) // |ki| = 128
    s <- m ⊕ k0 // s rappresenta lo stato
    for j = 1 to 10 do:
        s <- SBox(s)
        s <- ShiftRows(s)
        if j <= 9 then: s <- MixColumns(s)
        s <- s ⊕ kj
    return s
```

L'algoritmo parte con una fase di espansione della chiave, che dai 128 bit (o 192 o 256) della chiave iniziale produce 10 (o 12 o 14) parole con lunghezza 128 bit.

Il messaggio in bytes viene organizzato in una matrice 4 x 4. Successivamente, si eseguono in ordine: uno XOR fra il messaggio e la prima delle chiavi, al risultato viene applicata una trasformazione non lineare con la S-Box e infine la matrice di output viene sottoposta alle operazioni di ShiftRows e MixColumns per "diffondere" la trasformazione su tutti i bytes.

Questa sequenza di operazioni viene ripetuta per tutte le chiavi prodotte dall'EspandiChiave, anche se generalmente l'ultima passata ignora il MixColumns, che non influirebbe sulla sicurezza dell'output.

4) Definire formalmente il concetto di funzione pseudocasuale sicura

Una funzione pseudocasuale è considerata sicura se il suo comportamento è indistinguibile da quello di una vera funzione casuale.

Per definire più formalmente il significato di indistinguibilità, si immagini il seguente esperimento:

```
ESP_f^prf-1(A):
    k <-R K
    b <- A^{F_k}
    return b

ESP_f^prf-0(A):
    g <-R Func(D, R)
    b <- A^g
    return b
```

Nel primo caso, la funzione che l'avversario A interroga (in stile blackbox) è una funzione pseudocasuale. Nel secondo caso, si tratta di una vera funzione casuale. Si definisce quindi il vantaggio di A nel seguente modo:

$$Adv(A) = |Pr[ESP_f^{prf-1}(A) = 1] - Pr[ESP_f^{prf-0}(A) = 1]|$$

Si tratta del valore assoluto della probabilità che A abbia ragione pensando di trovarsi nel primo esperimento meno la probabilità che A pensi erroneamente di trovarsi sempre nel primo caso.

Le due funzioni sono indistinguibili, e quindi la funzione pseudocasuale è sicusa se $ADV(A)$ è una quantità trascurabile.

5) Sia $F:\{0,1\}^k \times \{0,1\}^l \rightarrow \{0,1\}^l$ una funzione pseudocasuale sicura. Siano inoltre $\alpha, \beta \in \{0,1\}^l$ due stringhe di bit note. Vogliamo utilizzare F per costruire una funzione $G:\{0,1\}^k \times \{0,1\}^{2l} \rightarrow \{0,1\}^l$, nel seguente modo (il simbolo || denota l'operazione di concatenazione):

```
G_k(x):
    xL || xR <- x  // |xL| = |xR| = L
    y1 <- F_k(xL ⊕ α)
    y2 <- F_k(xR ⊕ β)
    y <- y1 ⊕ y2
    return y
```

Dimostrare formalmente che G non è una funzione pseudo-casuale sicura.

Per dimostrare il fatto che G_k non sia una funzione pseudo-casuale sicura deve essere possibile costruire un attacco che permetta ad un avversario con potenza di calcolo limitata di avere un vantaggio non trascurabile. Immaginiamo il seguente attacco:

```
A(G_k):
    xL <- α
    xR <- β
    y <- O_{G_k}(xL || xR)
    yL || yR = y // |yL| = |yR| = L
    if yL = yR: return 1
    else: return 0
```

Grazie a questa costruzione, sappiamo che la funzione G_k chiamerà F_k passando la come input la stringa 0^l entrambe le volte, ottentendo quindi una stringa del tipo x || x, con $x \in \{0,1\}^l$.

- **L'oracolo usa G_k:** la stringa possiede la caratterisca cercata ed A indovinerà sicuramente.
- **L'oracolo usa una funzione casuale:** la probabilità di essere tratti in inganno è $\frac{1}{2^l}$, che rappresenta la probabilità che l'output della funzione abbia la caratteristica che cerchiamo.

$$Adv(A) = |Pr[ESP^{prf-1}_{G_k}(A) = 1] - Pr[ESP^{prf-0}_{G_k}(A) = 1]| = 1 - \frac{1}{2^l} \gg 0$$

La funzione G_k **non** è una funzione pseudo-casuale sicura.