Esonero 2020/2021

Programmazione Procedurale con Laboratorio

Canali per le domande: ricevimento, email: francesco.santini@unipg.it, Telegram: @safran

1 Cifrare con operatore XOR

L'operatore XOR (eXclusive OR, o $\oplus$)[1] riveste una notevole importanza negli algoritmi di cifratura. Per esempio, l'algoritmo RC4[2], utilizzato anche per la sicurezza di protocolli wireless come *WPA*, genera un flusso di bit pseudo-casuali (*keystream*). Tale flusso è combinato mediante un'operazione di XOR con il testo in chiaro (*plaintext*) per ottenere il testo cifrato (*ciphertext*). L'operazione di decifratura avviene nella stessa maniera, passando in input il testo cifrato ed ottenndo in output il testo in chiaro (questo perché lo XOR è un'operazione simmetrica). Vedere Figura 1 e Figura 2.

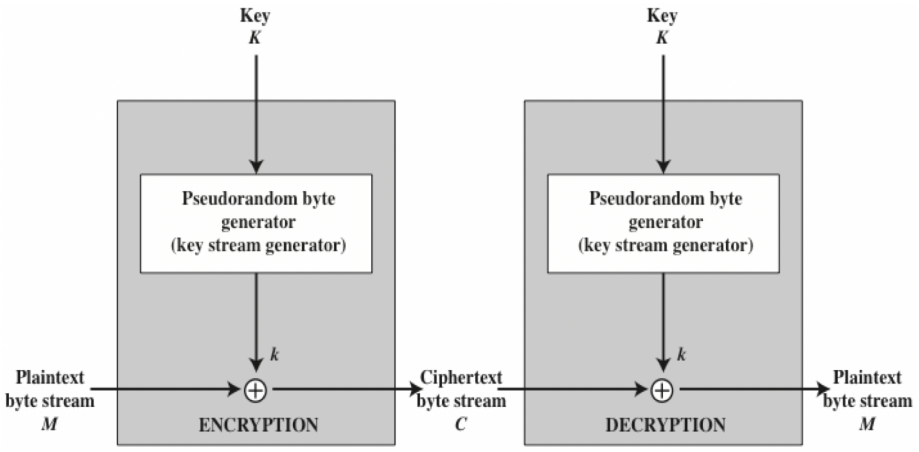

Figura 1. Cifrare con RC4 [“Sicurezza delle Reti - Applicazioni e standard”, William Stallings] .

2 Esercizio

Il programma deve richiedere la stringa di testo da cifrare e salvarla in un array; tale stringa rappresenta il nostro plaintext *M*. Suppore che la lunghezza massima della stringa sia 128 posizioni. Controllare che la lunghezza massima della stringa letta non superi

[1] `https://it.wikipedia.org/wiki/Disgiunzione_esclusiva`.
[2] `https://it.wikipedia.org/wiki/RC4`.

Input		Output
A	B	
0	0	0
0	1	1
1	0	1
1	1	0

Figura 2. Tabella di verità dell'operatore XOR [Wikipedia].

tale dimensione: utilizzare per esempio la funzione *fgets*[3]. Controllare la lunghezza di ciò che si legge da fonti di dati insicure è fondamentale per prevenire attacchi di tipo *buffer overflow*[4].

Dopodiché, è necessario richiedere all'utente in quale maniera vuole procedere. È possibile

1. chiedere all'utente una stringa di lunghezza uguale o superiore (sempre non più di 128 caratteri comunque: utilizzare *fgets*), con la quale cifrare il testo M. Questa stringa rappresenta quindi la nostra chiave k. Se la lunghezza della stringa k è minore stretta della lunghezza M, chiedere nuovamente all'utente di inserire una stringa di lunghezza maggiore o uguale. Oppure,
2. generare una chiave casuale k di lunghezza pari alla lunghezza di M. Stampare la k generata sullo schermo. Attenzione alcuni caratteri della tabella ASCII sono "particolari" una volta stampati sullo schermo, soprattutto da 0 a 31.[5] Ogni carattere di k deve essere generato in modo casuale, ed essendo un carattere ASCII, il suo valore può essere compreso tra 0 e 127. Vedere la Figura 3 per la generazione di un valore in questo intervallo.

Le due possibilità sopra possono essere richieste dall'utente inserendo un numero, 1 o 2 (leggendo un *int* da tastiera). Attenzione la funzione *scanf* è fonte inesauribile di problemi nella lettura di *char*[6], ecco perché si consiglia di leggere un intero.

A questo punto, sia nel caso 1 che nel caso 2 le chiavi k sono fatte da stringhe di caratteri (tipo *char*), essendo le chiavi delle semplici stringhe di testo. Cifrare quindi M utlizzando l'operazione $M \oplus k = C$ (XOR), e stampare su schermo la stringa risultante C, che rappresenta M cifrato con k. L'operazione deve essere effettuata tra caratteri corrispondenti di M e k (primo carattere di M con primo carattere di k, secondo con secondo, etc.). Stampare C sullo schermo.

Infine effettuare $C \oplus k$ e stampare la stringa corrispondente, che per le proprietà dello XOR è esattamente M, cioè il messaggio originario. Stampare M ottenuto in questo modo sullo schermo.

[3] `https://www.educative.io/edpresso/how-to-use-the-fgets-function-in-c`.
[4] `https://it.wikipedia.org/wiki/Buffer_overflow`.
[5] Tabella ASCII: `http://www.asciitable.com`.
[6] `https://stackoverflow.com/questions/20139415/problems-with-scanf`l.

```c
#include <stdio.h>
#include <stdlib.h> // Da includere per utilizzare rand() e srand()
#include <time.h> // Da includere per utilizzare time()

int main () {
time_t t;

/* Inizializza il generatore di numeri casuali utilizzando il tempo attuale */
srand((unsigned) time(&t));  // Funzione da chiamare una volta sola nel programma

/* Ritorna un numero tra 0 e 127*/
printf("%d\n", rand() % 128); // Chiamare quando si ha bisogno di un numero random
}

```

Figura 3. Esempio di come funziona la generazione di un numero casuale.

3 Sottomissione

Si accettano solo sottomissioni attraverso GitHub all'indirizzo `https://classroom.github.com/a/-Tc2j-u-`. È necessario creare un account GitHub quindi. Dato che l'utilizzo di GitHub fa parte dell'esercizio, lo studente deve essere in grado di verificare da solo se l'upload ha avuto successo o meno (non si accettano domande in questo senso). La scadenza per la sottomissione è **Domenica 15 Novembre** (23:59). Nel file README.md del repository aggiungere 1) nome, 2) cognome e 3) matricola (seguire le istruzioni sul file README). L'esonero sarà valutato con SÌ o NO. In caso di mancata sottomissione, o di risposta NO come risultato, in sede di esame orale saranno chieste una o più domande aggiuntive.

4 Note finali

Aggiungere i flag *-std=c11 -Wall* (per esempio `gcc -c crypto.c -std=c11 -Wall`), per essere sicuri di seguire lo standard *C 2011* e farsi segnalare tutti i *warning* rispettivamente. I *warning* vanno TUTTI rimossi.