Lab 1 – Fondamenti di Cybersecurity

GRUPPO – LO HACKER
Manuel Castiglia
Thomas Westerman

GCD and The Euclidean Algorithm

Per risolvere questo problema abbiamo utilizzato l'Algoritmo di Euclide Esteso.
Il primo passo è stato comprendere l'algoritmo euclideo, che si basa sulla ripetuta sottrazione del numero più piccolo dal più grande fino a quando non si ottiene un divisore comune. Successivamente siamo passati alla scrittura in Java dell'algoritmo:

- L'algoritmo prende in input i due numeri (a,b) di cui si vuole calcolare il GCD e un vettore nel quale si vanno a salvare i coefficienti che ci serviranno per la verifica della seguente equazione
    - a * s + b * t = GCD(a,b)
- Inizialmente si vanno ad inizializzare 4 variabili per il calcolo del GCD
- Successivamente si entra in un ciclo while (fino a che non si trova GCD) dove si va a calcolare il quoziente q tra r0 e r e si aggiornano i valori r0 e r per trovare il GCD (come nella versione semplice dell'algoritmo) e in più si aggiornano i coefficienti s e t nel seguente modo
    - s = s0 - q * s
    - t = t0 - q * t
- Così facendo uscendo dal ciclo while, avremo il GCD in r0 e i coefficienti 's' e 't' giusti nel vettore

Una volta scritto l'algoritmo, abbiamo completato lo script andando a recuperare i dati da riga di comando e ci siamo occupati della stampa dei risultati e della gestione degli errori

Multiplicative Inverse and The Extended Euclidean Algorithm

Per risolvere questo problema, abbiamo utilizzato l'Algoritmo Esteso di Euclide con qualche differenza rispetto a quanto fatto prima.
L'obbiettivo era quello di trovare l'inverso moltiplicativo a^{-1} mod n. Per prima cosa abbiamo definito una funzione basata anche questa sull'Algoritmo Esteso di Euclide

- La funzione prende in input i due numeri necessari per il calcolo
- Se il modulo passato è 1, restituisco 0 in quanto ogni numero in modulo 1 è sempre invertibile in modulo 1
- Successivamente fa partire un ciclo while fino a che non si trova l'inverso moltiplicativo (fino a che non diventa 1) dove calcoliamo il quoziente tra 'a' e 'mod' e aggiorniamo i valori di 'a' e 'mod' come nell'algoritmo di Euclide

- Per garantire che l'inverso moltiplicativo sia positivo, aggiungiamo mod a x1 se è negativo e lo restituiamo

Una volta scritto l'algoritmo, abbiamo completato lo script andando a recuperare i dati da riga di comando e ci siamo occupati della stampa dei risultati e della gestione degli errori

The RSA Cryptosystem

Questo problema richiedeva di implementare lo scambio di messaggi tra due persone implementando l'algoritmo RSA.
Per prima cosa, andiamo a definire i vari algoritmi per l'implementazione di RSA:

- Euclide semplice per il calcolo del GCD di due numeri, in quanto RSA richiede che il GCD dei parametri 'p' e 'q' sia pari ad 1 (devono essere coprimi)
- Inverso moltiplicativo per andare a calcolare l'esponente 'a' utilizzato per la decifratura da parte del destinatario
- Algoritmo Square and Multiply per la cifratura del messaggio da parte del mittente Una volta definiti questi algoritmi, si recuperano i dati passati da riga di comando e si dichiarano BigInteger per gestire il caso anche di parametri molto grandi.

Il primo passo è quello di calcolare 'n' (p*q) e pi(n), così da poter poi verificare che il GCD tra pi(n) e b sia effettivamente 1 (in caso così non fosse, si stampa un messaggio e si esce dall'esecuzione del programma).
Se questa prima condizione viene rispettata, va a calcolare l'esponente per cifrare il plaintexr e si chiede all'utente di digitare su tastiera il messaggio (sotto forma di numeri interi) da voler inviare, così da procedere alla sua cifratura con l'algoritmo Square and Multiply.
Infine si va a calcolare e stampare il testo decifrato così da vedere se corrisponde con quanto inserito da tastiera

Istruzioni per il testing

Per testare i file, aprire il terminale, impostare la cartella di riferimento con quella dove sono salvati i dati e procedere alla compilazione ed esecuzione dei file

- Compilazione --> javac nomefile.java
- Esecuzione --> java nomefile <parametri>

```
PS C:\Users\manue\Desktop\Laboratorio_Cybersecurity\Laboratorio1\src> javac RSA.java
PS C:\Users\manue\Desktop\Laboratorio_Cybersecurity\Laboratorio1\src> java RSA 101 113 3533
Il gcd tra b e pi(n) e' 1
L'inverso moltiplicativo di 3533 modulo 11200 e' 6597
Inserisci il plaintext da inviare: 9726
Il messaggio cifrato inviato e' 5761
Il testo decrittato e': 9726
PS C:\Users\manue\Desktop\Laboratorio_Cybersecurity\Laboratorio1\src>
```