Proiect IC
Honey Encryption

Tufa Adriana 343 C4

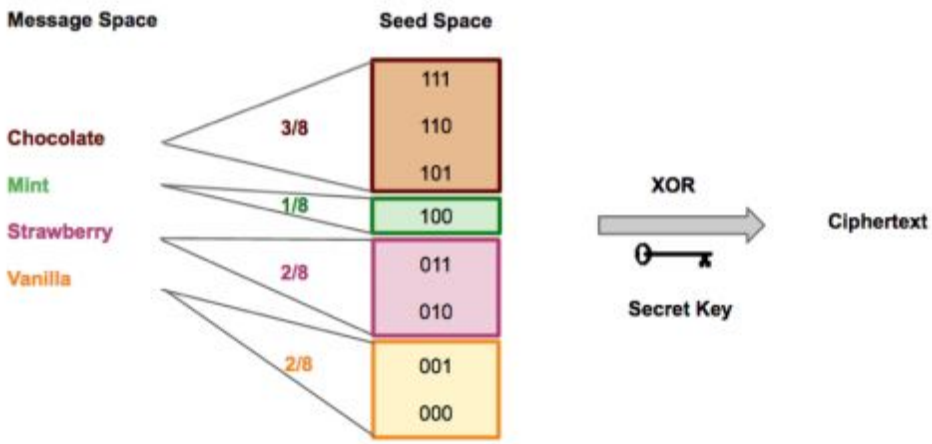
Intr-o schema tipica de criptare atacatorul poate incerca sa ghiceasca cheia folosita prin brute-force. Cum stie cand a gasit cheia buna? Atunci cand mesajul decriptat este unul valid (are sens). Honey encryption este o noua schema de criptare prin care orice cheie folosita va genera o decriptare valida. Astfel, atacatorul nu va sti care din cheile incercate este cea buna.

Honey encryption a fost propus in primul rand ca o modalitate de a adauga mai multa siguranta sistemelor ce folosesc parole, in special parole alese de utilizatori. Acestea sunt in general slabe si au o entropie mica, astfel incat sistemul este vulnerabil. Folosind HE, atacatorul nu va sti care este cheia buna chiar si dupa ce a incercat toate cheile posibile.

HE se bazeaza pe DTE (Distribution-Transforming Encoder), care mapeaza elementele din spatiul mesajelor posibile la un nou spatiu, numit Seed Space, care contine string-uri de lungime n. Pentru a contrui un DTE corect, trebuie sa se cunoasca distributia mesajelor in spatiul initial, adica probabilitatile acestora de aparitie si diferite dependente intre ele (cum este in cazul unei gramatici).

DTE - Criptare + Decriptare:

Sa luam un exemplu in care mesajele posibile sunt aromele de inghetata: chocolate, mint, strawberry si vanilla si seed space este spatiu mesajelor de lungime 3, deci in total 8 elemente. Atribuim probabilitatile de aparitie a mesajelor raportate la numarul de elemente ale seed space, astfel incat suma lor sa fie 1. Folosind ordinea in care am enumerat cuvintele, putem obtine urmatoarea mapare:



Daca plaintext-ul este “Chocolate”, atunci se va alege random un string din intervalul alocat mesajului in seed space si se va face XOR cu cheia secreta.

Pentru a calcula probabilitatile, trebuie sa fie cunoscut CDF-ul (cumulative ditribution function) spatiului de mesaje.

La decriptare, facand XOR intre ciphertext si cheie obtinem seed-ul, care detine o anumita pozitie in seed space. De aici trebuie sa stim cum este mapat spatiul de mesaje, de obicei printr-o tabela inversa celei de la criptare, care contine anumite valori din spatiul de mesaje, avand ca si cheie CDF-ul lor. Folosind binary search se va determina in ce interval se afla seed-ul calculat de noi, si apoi se vor incerca in mod secvential toate variantele din acel interval, pana se gaseste valoarea corecta a CDF-ului.

HE pentru carduri de credit:

Implementarea mea se reduce strict la spatiul mesajelor de tip card de credit, intrucat a fost analizat si se poate obtine usor o functie de distributie valida. Algoritmul poate fi folosit pentru criptarea cardurilor in formularele de pagini web, sau daca se doreste stocarea criptata a acestora in memorie.

Pentru a implementa DTE este util să se ştie formatul numerelor cardurilor de credit. Acestea au intre 12 şi 19 cifre dintre care primele 6 reprezintă un numar de identificare al companiei, iar urmatoarele (mai puţin ultima) reprezintă un numar random unic fiecarui utilizator. Ultima cifra este un checksum calulata cu ajutorul algoritmului Luhn – se face suma cifrelor şi se aduna de 9 ori, iar la sfârşit se aplica modulo 10. Pentru a construi spaţiul mesajelor posibile, s-au adunat date despre numere de carduri valide şi s-au strâns 2039 prefixe. S-a constatat ca lungimea predominanta a numarului este de 16 cifre. Datele colectate au fost puse într-un dicţionar de forma: prefix: [degrees of range, card length, weight]. Degrees of range reprezintă câte cifre „libere” sunt în prefix iar weight reprezintă numărul de reprezentari a prefixului şi depinde de gradul de libertate. (ex pentru 6424** degrees of range = 2 iar weigth = 100).

Având aceste informaţii putem se pot defini functiile folositoare DTE-ului.

PDF – Probability Distribution Function:

Fiecarui prefix unic de 6 cifre i se atribuie aceeaşi probabilitate, astfel ca ceea ce va face diferenţa este numărul de cifre din postfix – d. Deci probabilitatea unui mesaj va fi $10^{-(d-1)}$. Pentra ca suma probabilitatilor să fie 1, se va scala aceasta probabilitate la suma ponderilor (weigth) prefixelor.

CDF – Cumulative Distribution Function:

Se calculează prima data CDF-ul pentru toate prefixele. Apoi, pentru un mesaj dat, se va extrage prefixul şi CDF-ul calculat pentru acesta şi având în vedere ca toate mesajele cu acel prefix au aceeaşi lungime, se va adauga un offset proportional cu lungimea.

Pseudocod criptare:

```
encode(m):
    start = CDF(m) * SEED_SPACE_SIZE
    end = int(start + PDF(m) * SEED_SPACE_SIZE) -1
    start = int(start)
    seed = random(range(start, end))
    return seed
```

Pseudocod decriptare:

```
decode(m, inverse_table):
    seed_prop = float(s) / SEED_SPACE_SIZE
    (prev_value, prev_msg) = binary_search(inverse_table, seed_prop)
    next_msg = next_message(prev_msg)
    next_value = CDF(next_msg)

    while seed_loc >= next_value:
        (prev_value, prev_msg) = (next_value, next_msg)
        next_msg = next_message(prev_msg)
        next_value = CDF(next_msg)
    return prev_msg
```

Funcţia next_message primeşte un numar şi returneaza urmatorul card de credit astfel: îndepărtează ultima cifra, aduna 1 şi recalculeaza checksum-ul.

Evaluarea algoritmului

HE poate fi folosită cu succes în aplicaţii în care utilizatorul trebuie sa introduca o parola deoarece nu contează foarte mult entropia acesteia pentru ca în cazul unui atac, adversarul nu va şti sa distinga intre mesajele decriptate incorect şi mesajul bun. Dar atunci când nu se foloseşte HE şi adversarul are acces la aplicaţie şi la posibilitatea de a introduce un numar mare de parole, el poate încerca un atac brute-force sau dictionary-based pentru a găsi parola.

Pentru a evalua algoritmul am făcut o comparaţie cu ce se întâmpla atunci când un mesaj este decriptat cu o cheie greşita în cazul în care se foloseşte o schema diferita de HE, şi anume PBE – Password Based Encryption. Aceasta metoda de criptare se bazează pe o cheie generata din parola aleasa de utilizator. În multă cazuri, aceasta parola nu este foarte puternica (sunt alese cuvinte din dicţionar, sau „1234”) şi de aceea cheia generata nu este tocmai una care sa garanteze securitate sporita. Pentru a creşte gradul de securitate se foloseşte o tehnica pentru a imbunatati parola numita salting. Practic se adauga câteva cifre

random la sfârşitul parolei, după care se genereaza cheia. Aceasta cheie se foloseşte la un algoritm de criptare de genul AES.

În implementarea mea am folosit o parola de 4 cifre introdusa de utilizator, un salt de 12 cifre şi criptare cu AES. La decriptare am testat brute-force toate parolele de 4 cifre şi am analizat output-ul. Din cele 10000 de mesaje, doar unul putea fi identificat cu formatul unui card de credit, restul mesajelor continand caractere diferite de cifrele 0-9. Aşadar atacatorul poate găsi foarte uşor plaintextu-ul în cazul unei parole slabe. Dar nu şi în cazul lui HE!