- Konstantinos Filippou
- ics23044

Exercise 1

Έστω ένας μεγάλος πρώτος, ας πούμε **2^159 < p < 2^160,** και **K=M=C= {1,2,3,….,p-1}** . Η Alice και ο Bob επιλέγουν τυχαία ένα κλειδί **k∈K, (δηλ. 1<=k<p)** και ορίζουν την κρυπτογράφηση ως:ek(m) = k*m (mod p) . Η αντίστοιχη συνάρτηση αποκρυπτογράφησης είναι: **dk(c) =k^-1 * c (mod p)** , όπου **k^-1 mod p**, δηλ. ο αντίστροφος του **k modulo p** .

- Ο κρυπταλγόριθμος αυτός έχει τις ιδιότητες 1 και 2 (διαφάνειες 22 και 23, Crypto2025_02)?
- Ο κρυπταλγόριθμος είναι ασφαλής έναντι επίθεσης COA?
- Ο κρυπταλγόριθμος είναι ασφαλής έναντι επίθεσης KPA?
- Αν οι Alice και Bob ορίσουν τη συνάρτηση κρυπτογράφησης να είναι απλά , δηλαδή χωρίς αναγωγή , τότε ο κρυπταλγόριθμος εξακολουθεί να είναι ασφαλής έναντι επίθεσης COA?

Solution

- Ο κρυπταλγόριθμος έχει τις ιδιότητες 1 και 2 καθώς ο πολλαπλασιασμός k , m με modulo p και ο υπολογισμός του αντίστροφου k modulo p είναι γρήγοροι λόγω της χρήσης του mod (%) και xgcd (Extended Euclidean Algorithm).
- Ο κρυπταλγόριθμος δεν είναι ασφαλής έναντι επίθεσης COA (Ciphertext Only Attack) διότι με βάση το dk(c), αν επιλέξει δύο ciphertexts γνωρίζει τον λόγο τους c2/c1≡m2/m1(mod p), αυτό αποτελεί σε διαρροή πληροφορίας διότι αν και το p είναι μεγάλος αριθμός ο αλγόριθμος είναι σχετικά απλός. Αν υποθέσει ότι κάποιο ciphertext c αντιστοιχεί σε plaintext m, η πράξη γίνεται άμεσα μέσω των παραπάνω συναρτήσεων που αναφέραμε.
- Ο κρυπταλγόριθμος δεν είναι ασφαλής έναντι επίθεσης KPA (Known Plaintext Attack) καθώς αν γνωρίζει ένα ζεύγος (m,c) τότε μπορεί να βρει το k απευθείας ως k≡c·m^−1(mod p)
- Ο αλγόρθμος γίνεται ακόμη λιγότερο ασφαλής έναντι επίθεσης COA καθώς πλέον είναι απλά ένας πολλαπλασιασμός, και βρίσκεται από τον λόγο:
  k = c/m.

Exercise 2

Χρησιμοποιώντας τον κρυπταλγόριθμο Pohlig-Hellman με πρώτο p = 2633 και κλειδί κρυπτογράφησης e=9, κρυπτογραφήστε και στη συνέχεια αποκρυπτογραφήστε το μήνυμα "The gold is hidden in the garden!"

Solution

Στην αρχή πήγα να κρυπτογραφήσω και να αποκρυπτογραφήσω το μήνυμα χωρίς block, όταν πήγα να τρέξω το πρόγραμμα όμως το p ήταν πολύ μικρός αριθμός σε αντίθεση με το message. Όταν τρέχουμε το PH_enc(m,e,p) το αποτέλεσμα είναι m^e mod p, επομένως το ciphertext θα είναι πάντα μεταξύ 0 και p-1 (2632). Ως αποτέλεσμα χάνουμε πληροφορία και η αποκρυπτογράφηση δεν ανακτάει το αρχικό μήνυμα.

Επομένως, δημιούργησα την συνάρτηση encrypt_message_block η οποία δέχεται τα αρχικά μας ορίσματα

της PH_enc μαζί με το μέγεθος του block. Αφού η φράση είναι μεγάλη δεν χωράει πάνω από έναν χαρακτήρα, επομένως block_size = 1. Στη συνέχεια , δημιούργησα και την decrypt_message_block που δέχεται τα προηγούμενα ορίσματα της PH_dec, η μόνη διαφορά είναι ότι το c θα είναι σε blocks.

```
def encrypt_message_block(message, e, p, block_size):
    blocks = []
    for i in range(0, len(message), block_size):
        block = message[i:i+block_size]
        m_block = str2num(block)
        if m_block >= p:
            raise ValueError(f"Block '{block}' είναι πολύ μεγάλο για p={p}")
        c_block = PH_enc(m_block, e, p)
        blocks.append(c_block)
    return blocks

def decrypt_message_block(blocks, d, p):
    message_parts = []
    for c_block in blocks:
        m_block = PH_dec(c_block, d, p)
        block_msg = num2str(m_block)
        message_parts.append(block_msg)
    return ''.join(message_parts)
```

```
p = 2633
e = 9
d = inverse_mod(e, p-1)
# Σπάσιμο σε block
message = "The gold is hidden in the garden!"
blocks = encrypt_message_block(message, e, p, block_size=1)
print(f"Μπλοκ: {blocks}")

decrypted_msg = decrypt_message_block(blocks, d, p)
print(f"Αποκρυπτογραφημένο: '{decrypted_msg}'")
```

```
Μπλοκ: [655, 2034, 592, 551, 1172, 1911, 1514, 2565, 551, 1131, 882, 551, 2034, 1131, 2565, 2565, 592, 2053, 551, 1131, 2053, 551, 1719, 2034, 592, 55
1, 1172, 356, 2388, 2565, 592, 2053, 345]
Αποκρυπτογραφημένο: 'The gold is hidden in the garden!'
```

Exercise 3

1. Με modulus p = 29 και άγνωστο κλειδί κρυπτογράφησης e ο κρυπταλγόριθμος Pohlig-Hellman παράγει το κρυπτοκείμενο 04 19 19 11 04 24 09 15 15. Κρυπταναλύστε τον κρυπταλγόριθμο αυτόν, αν είναι γνωστό ότι το τμήμα κρυπτοκειμένου **24** αντιστοιχεί στο γράμμα απλού κειμένου U (με αριθμητικό ισοδύναμο **20**). Χρησιμοποιήθηκε σχήμα κωδικοποίησης όπου A=00, ..., Z=25.

Solution

Γνωρίζουμε ότι c=m^e mod p , 24 = 20^e mod 29 , επομένως 20^e ≡ 24 (mod 29).

$20^1 = 20$

$20^2 = 400 \bmod 29 = 400 - 13\times29 = 400 - 377 = 23$

$20^3 = 20^2 \times 20 = 23 \times 20 = 460 \bmod 29 = 460 - 15\times29 = 460 - 435 = 25$

$20^4 = 20^3 \times 20 = 25 \times 20 = 500 \bmod 29 = 500 - 17\times29 = 500 - 493 = 7$

$20^5 = 20^4 \times 20 = 7 \times 20 = 140 \bmod 29 = 140 - 4\times29 = 140 - 116 = 24$

Άρα e=5

d = e^{-1} mod (p-1) ,5^-1 mod 28:

Βρίσκουμε τον αντίστροφο του 5 mod 28, 5 × 17 = 85 ≡ 1 (mod 28) (αφού 85 - 3×28 = 85 - 84 = 1). Άρα d=17

m = c^d mod p = c^17 mod 29. (Για τις παρακάτω πράξεις χρησιμοποιήθηκε η πράξη % καθώς στην προηγούμενη εργασία δείξαμε πως υπολογίζεις modulo με το χέρι)

Για 04: έχουμε 04^17 mod 29: 6: -------> G

Για 19: έχουμε 19^17 mod 29: 14 -------> O

Για 19: έχουμε 19^17 mod 29: 14 -------> O

Για 11: έχουμε 11^17 mod 29: 3 -------> D

Για 04: έχουμε 04^17 mod 29: 6 -------> G

Για 24: έχουμε 24^17 mod 29: 20 -------> U

Για 09: έχουμε 09^17 mod 29: 4 -------> E

Για 15: έχουμε 15^17 mod 29: 18 -------> S

Για 15: έχουμε 15^17 mod 29: 18 -------> S

Επομένως το αρχικό κείμενο είναι: GOOD GUESS

Exercise 4

Ένα βασικό μειονέκτημα στον κρυπταλγόριθμο P-H είναι ότι **το μήνυμα απλού κειμένου m μπορεί να μην είναι ένας γεννήτορας modulo p** και στην πραγματικότητα μπορεί να έχει χαμηλή τάξη με αποτέλεσμα να είναι εύκολη η επίλυση του DLP σε μια επίθεση τύπου KPA. Δημιουργήστε ένα προγραμματιστικό παράδειγμα στο Z5003

Solution

Μπορούμε να πάρουμε το πιο απλό παράδειγμα, δηλαδή m=1 καθώς μας δίνει την μικρότερη δυνατή τάξη, άρα είναι εύκολη η επίλυση του DLP.

```
p = 5003 #Z5003
e = 17  #Σχετικά πρώτο με 5002
d = inverse_mod(e, p-1)

# Αν m έχει τάξη k, τότε m^k ≡ 1 mod p
# Άρα c ≡ m^e ≡ m^(e mod k) mod p

# Παράδειγμα: m = 1 (τάξη 1)
m1 = 1
c1 = pow(m1, e, p)
print("m = 1 (τάξη 1)")
print("c = 1^e mod p = {c1}")
print("Για κάθε e, c = 1 → Εύκολο DLP!")
```

```
m = 1 (τάξη 1)
c = 1^e mod p = {c1}
Για κάθε e, c = 1 → Εύκολο DLP!
```

Exercise 5

Ολοκληρώστε και εκτελέστε τα 6 βήματα της παραπάνω δραστηριότητας με μήνυμα (plaintext) της επιλογής σας. Παραθέστε το στιγμιότυπο της εκτέλεσής σας.

Solution

Αρχικά ορίσαμε τις συναρτήσεις **PH(bits)**, **PH_enc(m,e,p)**, **PH_dec(c,d,p)**, **str2num(s)** και **num2str(n)**:

```
#Ορισμοί συναρτήσεων
def PH(bits):

    p=next_prime(ZZ.random_element(2^bits))
    while True:
        e = ZZ.random_element(1, p-1)
        if gcd(e, p-1) == 1:
            d=inverse_mod(e,p-1)
            return (e,d,p)

def PH_enc(m,e,p):
    return power_mod(m,e,p)


def PH_dec(c,d,p):
    return power_mod(c,d,p)

def str2num(s):
    map2ascii=[ord(char) for char in s]
    largeInt=ZZ(map2ascii,128)
    return (largeInt)

def num2str(n):
    digitsOfInt=n.digits(128)
    charOfmsg=[chr(dg) for dg in digitsOfInt]
    msg=''.join(charOfmsg)
    return (msg)
```

Στην συνέχεια με δικό μας κείμενο: "Hello there!", δείξαμε τις λειτουργίες κρυπτογράφησης και αποκρυπτογράφησης:

```
#Κλήσεις
# Convert a text string into a large integer μέσω str2num
message="Hello there!"
m = str2num(message)
print("Large Int:", m)
```

```
Large Int: 5107117560890101249815240
```

```
#Κλήσεις
# Convert a large integer into a text string μέσω num2str
msg = num2str(m)
print ("Text String:", msg)
```

```
Text String: Hello there!
```

```
#Κλήσεις
(e,d,p) = PH(128)
print ('The Alice secret:', e)
print ('The Bob secret:', d)
print ('Modulo is:', p, 'which is', len(p.binary()), 'bit length')
m = "Hello there!"
msg = str2num(m)
enc_msg = PH_enc(msg,e,p)
print ("The encrypted message is:", enc_msg)
c = PH_dec(enc_msg,d,p)
dec_msg = num2str(c)
print("The decrypted message is:", dec_msg)
```

```
The Alice secret: 907151402619316170653112310966752130 9
The Bob secret: 157582522889518679731718392972260271 29
Modulo is: 201111931181087796137837675396578851 83 which is 124 bit length
The encrypted message is: 116422493898295219002794372464347401 79
The decrypted message is: Hello there!
```