Υπολογιστική Κρυπτογραφία
Δεύτερη σειρά ασκήσεων

Μέη Αρετή, A.M.:03120062

December 18, 2024

Άσκηση 1

Θέλουμε να βρούμε τη λύση της εξίσωσης $x^2 \equiv y \mod n$, όπου $n = pq$ και p, q είναι πρώτοι αριθμοί, με την επιπλέον πληροφορία ότι το x είναι τετραγωνικό υπόλοιπο modulo n.

Επειδή $n = pq$ και οι p, q είναι πρώτοι αριθμοί με $\gcd(p, q) = 1$, από την ιδιότητα του Κινεζικού Υπολοίπου, οι σχέσεις modulo n διασπώνται ως εξής:

$$x^2 \equiv y \mod n \iff x^2 \equiv y \mod p \quad \text{και} \quad x^2 \equiv y \mod q.$$

Επιπλέον, το x είναι τετραγωνικό υπόλοιπο modulo n, δηλαδή υπάρχει κάποιος k τέτοιος ώστε:

$$k^2 \equiv x \mod n.$$

Άρα και:

$$k^2 \equiv x \mod p \quad \text{και} \quad k^2 \equiv x \mod q.$$

Από το κριτήριο Euler, γνωρίζουμε ότι ένας αριθμός x είναι τετραγωνικό υπόλοιπο modulo p αν και μόνο αν:

$$x^{(p-1)/2} \equiv 1 \mod p.$$

Αντίστοιχα, επειδή x είναι τετραγωνικό υπόλοιπο modulo q, ισχύει:

$$x^{(q-1)/2} \equiv 1 \mod q.$$

Συνδυάζοντας τις παραπάνω σχέσεις, πολλαπλασιάζουμε τους εκθέτες:

$$x^{(p-1)(q-1)/4} \equiv 1 \mod p \quad \text{και} \quad x^{(p-1)(q-1)/4} \equiv 1 \mod q.$$

Επομένως, ισχύει:

$$x^{(p-1)(q-1)/4} \equiv 1 \mod n,$$

επειδή $n = pq$ και οι p, q είναι πρώτοι με $\gcd(p, q) = 1$.

Από την παραπάνω σχέση, έχουμε:

$$x^{(p-1)(q-1)/4+1} \equiv x \mod n.$$

Αναδιατάσσοντας, βρίσκουμε ότι:

$$x^{\frac{(p-1)(q-1)+4}{4}} \equiv x \mod n.$$

Από την εξίσωση $x^2 \equiv y \mod n$, μπορούμε να αντικαταστήσουμε στη θέση του x:

$$x^{\frac{(p-1)(q-1)+4}{8}} \equiv y^{\frac{(p-1)(q-1)+4}{8}} \mod n.$$

Επομένως, η κύρια τετραγωνική ρίζα x δίνεται από τον τύπο:

$$x \equiv y^{\frac{(p-1)(q-1)+4}{8}} \mod n.$$

Άσκηση 2

Θεωρούμε ότι ο αντίπαλος διαθέτει δύο ζεύγη απλού κειμένου και κρυπτοκειμένου (m_1, c_1) και (m_2, c_2). Επίσης, θεωρούμε την αποκρυπτογράφηση μέσω του κλασικού DES ως $D_{k_1}(c_i)$, όπου $i \in \{1, 2\}$. Η αποκρυπτογράφηση επιστρέφει $m_i \oplus k_2$.

Από τη σχέση αυτή προκύπτει ότι:

$$D_{k_1}(c_1) \oplus D_{k_1}(c_2) = (m_1 \oplus k_2) \oplus (m_2 \oplus k_2) = m_1 \oplus m_2,$$

όπου m_1, m_2, c_1, c_2 είναι γνωστά, ενώ $D_{k_1}(c_1)$ και $D_{k_1}(c_2)$ είναι άγνωστα. Ο αντίπαλος μπορεί να εκτελέσει τον παρακάτω αλγόριθμο για να ανακτήσει τα k_1 και k_2:

Αλγόριθμος εύρεσης k_1 και k_2

Για κάθε πιθανό k_1: Υπολόγισε $D_{k_1}(c_1)$ με γνωστά c_1 και k_1. Υπολόγισε $D_{k_1}(c_2)$ με γνωστά c_2 και k_1. Έλεγξε αν $D_{k_1}(c_1) \oplus D_{k_1}(c_2) = m_1 \oplus m_2$. Αν ισχύει, υπολόγισε $k_2 = m_1 \oplus D_{k_1}(c_1)$ και επέστρεψε τα k_1, k_2. Αν όχι, επανέλαβε με το επόμενο πιθανό k_1.

Ο παραπάνω αλγόριθμος, στη χειρότερη περίπτωση, θα διατρέξει όλα τα πιθανά k_1. Καθώς το k_1 έχει μήκος 56 bits, υπάρχουν 2^{56} πιθανά κλειδιά. Για κάθε πιθανό k_1, απαιτούνται δύο αποκρυπτογραφήσεις (μία για κάθε c_i), με συνολικό κόστος στη χειρότερη περίπτωση $2 \cdot 2^{56}$ αποκρυπτογραφήσεις.

Στον κλασικό DES, η brute force επίθεση απαιτεί τη δοκιμή όλων των 2^{56} πιθανών κλειδιών, με συνολικό κόστος 2^{56} αποκρυπτογραφήσεις. Αντίστοιχα, στη χειρότερη περίπτωση της παραλλαγής DES-X, απαιτούνται $2 \cdot 2^{56}$ αποκρυπτογραφήσεις.

Παρότι ο αριθμός των αποκρυπτογραφήσεων διπλασιάζεται στην περίπτωση του DES-X, η διαφορά αυτή είναι αμελητέα ασυμπτωτικά. Επομένως, η παραλλαγή DES-X δεν παρέχει σημαντική αύξηση ασφάλειας σε σχέση με τον κλασικό DES υπό συνθήκες brute force επίθεσης.

Άσκηση 3

Ένα κλειδί k του DES θεωρείται ασθενές αν η συνάρτηση DES_k είναι ενέλιξη, δηλαδή ικανοποιεί τη σχέση:

$$DES_k(DES_k(x)) = x, \quad \forall x \in \{0,1\}^{64}$$

Τα ασθενή κλειδιά έχουν την ιδιότητα ότι όλες οι 16 υπο-κλειδιά $(k_1, k_2, \ldots, k_{16})$ που δημιουργούνται κατά τη διαδικασία κρυπτογράφησης είναι ίσα. Τέσσερα ασθενή κλειδιά του DES είναι τα εξής:

$$\begin{aligned} k_1 &= 0000000000000000 \\ k_2 &= FFFFFFFFFFFFFFFF \\ k_3 &= 1E1E1E1E0E0E0E0E \\ k_4 &= E0E0E0E0F1F1F1F1 \end{aligned}$$

Ένα κλειδί k του DES θεωρείται ημιασθενές αν:

- Δεν είναι ασθενές.
- Υπάρχει κλειδί k' τέτοιο ώστε:

$$DES_k^{-1}(x) = DES_{k'}(x), \quad \forall x \in \{0,1\}^{64}$$

Υπάρχουν συνολικά 6 ζεύγη ημιασθενών κλειδιών στον DES. Τέσσερα παραδείγματα ζευγών ημιασθενών κλειδιών είναι:

$$\begin{aligned} (k_1, k_1') &= (0x0123456789ABCDEF, 0xFEDCBA9876543210) \\ (k_2, k_2') &= (0x1F1F1F1F0E0E0E0E, 0xE0E0E0E0F1F1F1F1) \\ (k_3, k_3') &= (0x01FE01FE01FE01FE, 0xFE01FE01FE01FE01) \\ (k_4, k_4') &= (0x1E1E1E1E0E0E0E0E, 0xE1E1E1E1F0F0F0F0) \end{aligned}$$

Άσκηση 4

Θεωρούμε ότι το κρυπτοσύστημα εφαρμόζει συνάρτηση $\mathcal{E}$, η οποία είναι ένα προς ένα (bijective). Από τη λειτουργία σε CBC έχουμε:

$$y_i = y_j \implies \mathcal{E}(y_{i-1} \oplus x_i) = \mathcal{E}(y_{j-1} \oplus x_j).$$

Λόγω του ότι η $\mathcal{E}$ είναι ένα προς ένα, έπεται:

$$y_{i-1} \oplus x_i = y_{j-1} \oplus x_j \implies x_i \oplus x_j = y_{i-1} \oplus y_{j-1}.$$

Συνεπώς, από τη σύγκρουση $y_i = y_j$, μπορούμε να εξάγουμε πληροφορία για τη διαφορά των blocks $x_i \oplus x_j$ του plaintext.

2. Πιθανότητα σύγκρουσης

Ορίζουμε ως NoColl_n το ενδεχόμενο να μην έχουμε σύγκρουση στα $\{y_1, y_2, \ldots, y_n\}$. Γνωρίζουμε ότι:

$$\Pr[\text{NoColl}_n] = \frac{(x-1)(x-2)\cdots(x-n+1)}{x^n},$$

όπου $x = 2^{64}$. Μια προσεγγιστική εκτίμηση είναι:

$$\Pr[\text{NoColl}_n] \leq e^{-\frac{n(n-1)}{2x}}.$$

Για $x = 2^{64}$, η πιθανότητα μη σύγκρουσης γίνεται:

$$\Pr[\text{NoColl}_n] \leq e^{-\frac{n(n-1)}{2\cdot 2^{64}}}.$$

Άρα η πιθανότητα σύγκρουσης είναι:

$$\Pr[\text{Coll}_n] = 1 - \Pr[\text{NoColl}_n] \geq 1 - e^{-\frac{n(n-1)}{2\cdot 2^{64}}}.$$

3. Χρησιμότητα της επίθεσης

Για να είναι χρήσιμη η επίθεση, απαιτούμε:

$$\Pr[\text{Coll}_n] \geq \frac{1}{2}.$$

Δηλαδή:

$$1 - e^{-\frac{n(n-1)}{2\cdot 2^{64}}} \geq \frac{1}{2}.$$

Απλοποιώντας:

$$e^{-\frac{n(n-1)}{2\cdot 2^{64}}} \leq \frac{1}{2}.$$

Λαμβάνοντας φυσικούς λογαρίθμους:

$$-\frac{n(n-1)}{2\cdot 2^{64}} \leq \ln\left(\frac{1}{2}\right) \implies \frac{n(n-1)}{2^{65}} \geq \ln(2).$$

Λύνοντας ως προς n:

$$n(n-1) \geq 2^{65}\ln(2) \implies n \geq \sqrt{2^{65}\ln(2)} + 1.$$

Υπολογίζοντας τη σταθερά, προκύπτει:

$$n \geq 0.8325\sqrt{2^{65}} + 1.$$

Συμπέρασμα

Για να είναι η επίθεση χρήσιμη, απαιτείται n τουλάχιστον ίσο με $0.8325\sqrt{2^{65}} + 1$, γεγονός που δείχνει ότι η πιθανότητα σύγκρουσης αυξάνεται ραγδαία για μεγάλες τιμές του n.

Άσκηση 5

1. Εύρεση μεγέθους block

Η ιδέα του αλγορίθμου είναι να ξεκινήσουμε με μήνυμα m μεγέθους ενός χαρακτήρα και να αυξάνουμε κάθε φορά το μέγεθός του κατά 1 χαρακτήρα. Σε κάθε βήμα ελέγχουμε το μήκος του ciphertext. Όταν «γεμίσει» ένα block και το μήνυμα περάσει στο επόμενο, θα παρατηρήσουμε αύξηση στο μήκος του ciphertext. Αν αυτό συμβεί στο i-οστό βήμα, τότε το μέγεθος του block υπολογίζεται ως:

$$\text{block size} = \text{length}(c_{i+1}) - \text{length}(c_i).$$

Ο αλγόριθμος σε ψευδοκώδικα περιγράφεται παρακάτω:

Listing 1: Ψευδοκώδικας για εύρεση μεγέθους block

```
m = 'A'
init_length = length(AESk(m || s))
Repeat:
    m = m + 'A'
    new_length = length(AESk(m || s))
    if new_length > init_length:
        block_size = new_length - init_length
        return block_size
    init_length = new_length
```

2. Έλεγχος για ECB mode

Έστω l το *block size*, που έχει υπολογιστεί από το προηγούμενο ερώτημα. Δημιουργούμε μήνυμα της μορφής:

$$m = m_1||m_2||m_1,$$

όπου $\text{length}(m_1) = \text{length}(m_2) = l$. Εάν το oracle χρησιμοποιεί ECB mode, τότε τα δύο blocks m_1 θα δώσουν ίδιο ciphertext c_1.

Υπάρχει, ωστόσο, και η περίπτωση το oracle να μη χρησιμοποιεί ECB mode, αλλά να προκύψει ίδια έξοδος λόγω collision, π.χ., σε CBC mode. Ωστόσο, για μικρά μηνύματα, η πιθανότητα να συμβεί κάτι τέτοιο είναι πρακτικά αμελητέα, γεγονός που καθιστά τον έλεγχο αξιόπιστο.

3. Αλγόριθμος για εύρεση της μυστικής συμβολοσειράς s

Για να βρούμε τη μυστική συμβολοσειρά s, χρησιμοποιούμε την ιδιότητα ότι μπορούμε να δημιουργήσουμε μηνύματα της επιλογής μας:

1. Υπολογίζουμε το μέγεθος του block b χρησιμοποιώντας τον αλγόριθμο της Ενότητας 1 (πιθανότατα $b = 16$ bytes).

2. Δημιουργούμε ένα μήνυμα m που είναι μήκους $b-1$, π.χ.:

$$m = \text{"}AAAAAAAAAAAAAAA\text{"}$$

Έτσι, το $m||s$ γεμίζει το πρώτο block και αφήνει το υπόλοιπο της s στο δεύτερο block.

3. Υπολογίζουμε το κρυπτοκείμενο $c_1 = AES_k(m||s)$.

4. Στη συνέχεια, αυξάνουμε το μήκος του m κατά 1 byte, π.χ.:

$$m' = \text{"}AAAAAAAAAAAAAAAA\text{"}$$

Υπολογίζουμε το νέο κρυπτοκείμενο $c_2 = AES_k(m'||s)$.

5. Συγκρίνουμε το πρώτο block του c_1 με το πρώτο block του c_2. Το νέο byte του m' μετακινεί το πρώτο byte της s στο δεύτερο block. Επαναλαμβάνουμε τη διαδικασία προσθέτοντας επιπλέον bytes στο m έως ότου ανακτηθεί ολόκληρη η s.

Με αυτόν τον τρόπο, μπορούμε να εξάγουμε τη μυστική συμβολοσειρά s που χρησιμοποιείται από το oracle.

Άσκηση 6

Εξετάζουμε τη φάση παραγωγής τυχαίων bytes ($PRGA$) για τη γεννήτρια RC4.

Πρώτη επανάληψη (first iteration)

$$i = 0, \quad j = 0$$

$$i = 1, \quad j = 0 + P[1] = P[1] \neq 2$$

Εκτελούμε την ενέργεια *swap*:

$$\text{swap}(P[1], P[a])$$

με:

$$P[1] \leftarrow P[a] = P[P[1]], \quad P[a] \leftarrow a$$

Ο πρώτος παραγόμενος byte είναι:

$$K_0 = P\big[(P[1] + P[a]) \mod 256\big] = (P[1] + a) \mod 256$$

Δεύτερη επανάληψη (second iteration)

$$i = 2, \quad j = a + P[1] = a$$

Εκτελούμε και πάλι την ενέργεια *swap*:

$$\text{swap}(P[2], P[a])$$

με:

$$P[2] \leftarrow P[a] = a, \quad P[a] \leftarrow P[2] = 0$$

Ο δεύτερος παραγόμενος byte είναι:

$$K_1 = P\big[(P[2] + P[a]) \mod 256\big] = P[a] = 0$$

Άρα, έχουμε:

$$\Pr[K_1 = 0] = 1, \quad \text{αν } P[2] = 0 \text{ και } P[1] \neq 2 \text{ μετά τη φάση KSA.}$$

Υπολογισμός πιθανότητας

Ορίζουμε το ενδεχόμενο $Q = (P[2] = 0 \wedge P[1] \neq 2)$. Αναζητούμε την πιθανότητα το δεύτερο byte να ισούται με μηδέν, δηλαδή την $\Pr[K_1 = 0]$. Έχουμε:

$$\Pr[K_1 = 0] = \Pr[K_1 = 0 \mid Q] \cdot \Pr[Q] + \Pr[K_1 = 0 \mid \neg Q] \cdot \Pr[\neg Q]$$

Επειδή $\Pr[K_1 = 0 \mid Q] = 1$ και $\Pr[K_1 = 0 \mid \neg Q] = \frac{1}{256}$, προκύπτει:

$$\Pr[K_1 = 0] = 1 \cdot \Pr[Q] + \frac{1}{256} \cdot (1 - \Pr[Q])$$

Θεωρούμε ότι η μετάθεση P ακολουθεί ομοιόμορφη κατανομή. Υπολογίζουμε:

$$\Pr[Q] = \Pr[P[2] = 0] \cdot \Pr[P[1] \neq 2] = \frac{1}{256} \cdot (1 - \Pr[P[1] = 2]) = \frac{1}{256} \cdot \left(1 - \frac{1}{256}\right) = \frac{1}{256} \cdot \frac{255}{256}$$

Άρα:

$$\Pr[Q] \simeq \frac{1}{256}$$

Υπολογίζουμε τη συνολική πιθανότητα:

$$\Pr[K_1 = 0] = 1 \cdot \frac{1}{256} + \frac{1}{256} \cdot \left(1 - \frac{1}{256}\right) = \frac{1}{256} + \frac{1}{256} \cdot \frac{255}{256}$$

$$\Pr[K_1 = 0] = \frac{1}{256} + \frac{255}{256^2} = \frac{256}{256^2} = \frac{1}{128} = 2^{-7}$$

Συμπεραίνουμε ότι:

$$\Pr[K_1 = 0] = 2^{-7}.$$

Άσκηση 7

1. $F_1(k,x) = F(k,x)||0$

Έστω διαχωριστής D που ρωτά το μαντείο O για $x_1, x_2, \ldots, x_n$, όπου $O(x_i) = y_i$. Ο D επιστρέφει αποτέλεσμα 1 αν όλα τα y_i τελειώνουν σε 0.

- Αν $O = F_1^k$, τότε:
$$\Pr[D^{F_1^k}(1^n) = 1] = 1$$
καθώς όλα τα y_i τελειώνουν σε 0.

- Αν $O = g$, όπου g είναι τυχαία συνάρτηση ομοιόμορφα επιλεγμένη, τότε:
$$\Pr[D^g(1^n) = 1] = \frac{1}{2} \cdot \frac{1}{2} \cdots \frac{1}{2} = \frac{1}{2^n}$$

Η διαφορά:

$$\left|\Pr[D^{F_1^k}(1^n) = 1] - \Pr[D^g(1^n) = 1]\right| = 1 - \frac{1}{2^n} > \text{negl}(n)$$

όταν το n είναι αρκετά μεγάλο. Συνεπώς, η F_1^k δεν είναι ψευδοτυχαία.

2. $F_2(k,x) = F(k,x) \oplus x$

Υποθέτουμε ότι η F_2 δεν είναι ψευδοτυχαία. Τότε υπάρχει διαχωριστής D_{F_2} που διαχωρίζει με μη αμελητέα πιθανότητα την F_2 από μια τυχαία συνάρτηση f_2. Δείχνουμε ότι υπάρχει και διαχωριστής D_F που διαχωρίζει με μη αμελητέα πιθανότητα την F από μια τυχαία συνάρτηση f, οδηγώντας σε άτοπο.

Ο D_F λειτουργεί ως εξής: Όταν ο D_{F_2} ζητά ένα x, ο D_F απαντά παριστάνοντας το μαντείο ως:

$$O_F(x) \oplus x$$

Παρατηρούμε:

- Αν $O_F = F$, τότε:
$$O_F(x) \oplus x = F(k,x) \oplus x = F_2(k,x)$$
Οπότε:
$$\Pr[D_F^{F^k}(1^n) = 1] = \Pr[D_{F_2}^{F_2^k}(1^n) = 1] = 1$$

- Αν $O_F = f$, τότε:
$$\Pr[D_F^f(1^n) = 1] = \Pr[D_{F_2}^{f \oplus x}(1^n) = 1] = \frac{1}{2^n}$$

Η διαφορά:

$$\Pr[D_F^{F^k}(1^n) = 1] - \Pr[D_F^f(1^n) = 1] = 1 - \frac{1}{2^n}$$

είναι μη αμελητέα, άρα η F είναι ψευδοτυχαία, οδηγώντας σε άτοπο.

3. $F_3(k,x) = F(k, x \oplus 1^n)$

Αφού:

$$F_3(k,x) = F(k, x \oplus 1^n) = F(k, \bar{x}),$$

όπου κάθε x αντιστοιχεί σε μοναδικό $\bar{x}$ και αντίστροφα, η F είναι ψευδοτυχαία. Συνεπώς, και η F_3 είναι ψευδοτυχαία.

4. $F_4(k,x) = F(k,x)||F(k,F(k,x))$

Έστω διαχωριστής D που έχει πρόσβαση σε μαντείο O, με $O(x) = y_L||y_R$. Ο D λειτουργεί ως εξής:

1. Υπολογίζει $O(x_1) = y_L||y_R$
2. Υπολογίζει $O(y_L) = y'_L||y'_R$
3. Επιστρέφει 1 αν $y'_L = y_R$.

- Αν $O = F_4^k$, τότε:
$$O(x_1) = F_k(x_1)||F_k(F_k(x_1))$$
και:
$$O(F_k(x_1)) = F_k(F_k(x_1))||F_k(F_k(F_k(x_1))),$$
όπου το y'_L ισούται με το y_R, οπότε:
$$\Pr[D^{F_4^k}(1^n) = 1] = 1$$

- Αν $O = g$, όπου g είναι τυχαία συνάρτηση, τότε ο D επιστρέφει 1 με αμελητέα πιθανότητα ϵ.

Η διαφορά:

$$\Pr[D^{F_4^k}(1^n) = 1] - \Pr[D^g(1^n) = 1] = 1 - \epsilon > \text{negl}(n),$$

οπότε η F_4^k δεν είναι ψευδοτυχαία.

Άσκηση 8

(α) Περίοδος της γεννήτριας και συνθήκες για το $gcd(p-1, q-1)$

Η γεννήτρια ψευδοτυχαίων bit BBS βασίζεται σε ένα Blum integer $n = pq$, όπου p και q είναι πρώτοι αριθμοί με $p \equiv q \equiv 3 \pmod 4$. Η περίοδος της γεννήτριας εξαρτάται από το n και την αρχική κατάσταση s_0 ως εξής:

$$\text{Περίοδος} = \text{lcm}(\text{ord}_p(s_0), \text{ord}_q(s_0)),$$

όπου $\text{ord}_p(s_0)$ και $\text{ord}_q(s_0)$ είναι η τάξη του $s_0 \pmod p$ και $\pmod q$ αντίστοιχα.

Η μέγιστη περίοδος της γεννήτριας επιτυγχάνεται όταν:

$$\text{lcm}(\text{ord}_p(s_0), \text{ord}_q(s_0)) = \frac{(p-1)(q-1)}{2},$$

εφόσον $p \equiv q \equiv 3 \pmod 4$.

Για να είναι η γεννήτρια αποτελεσματική, πρέπει να είναι μικρό το $gcd(p-1, q-1)$. Αν το $gcd(p-1, q-1)$ είναι μεγάλος αριθμός, τότε μειώνεται η πιθανότητα το $\text{lcm}(\text{ord}_p(s_0), \text{ord}_q(s_0))$ να επιτύχει τη μέγιστη δυνατή τιμή.

Λόγος για το μικρό $gcd(p-1, q-1)$**:** Το μικρό $gcd(p-1, q-1)$ εξασφαλίζει ότι τα $p-1$ και $q-1$ είναι σχεδόν «ασύμβατα», αυξάνοντας τις πιθανότητες για μέγιστη περίοδο. Αυτό είναι κρίσιμο για τη διατήρηση της ασφάλειας και της ομοιόμορφης κατανομής των εξόδων της γεννήτριας.

(β) Περίοδος της γεννήτριας με SafeSafe primes p και q

Ένας safe prime είναι της μορφής $p = 2p' + 1$, όπου p' είναι πρώτος. Ένας SafeSafe prime είναι ένας safe prime για τον οποίο:

$$p' = 2p'' + 1, \quad \text{με } p'' \equiv 1 \pmod 4.$$

Η μέγιστη περίοδος της γεννήτριας BBS, όταν τόσο ο p όσο και ο q είναι SafeSafe primes, δίνεται από:

$$\text{Μέγιστη περίοδος} = \frac{(p-1)(q-1)}{2}.$$

Απόδειξη: 1. Η περίοδος καθορίζεται από το $\text{lcm}(\text{ord}_p(s_0), \text{ord}_q(s_0))$. 2. Για SafeSafe primes, τα $p-1$ και $q-1$ είναι μεγιστοποιημένα ως γινόμενα πρώτων αριθμών (λόγω της μορφής $p = 2p' + 1$ και $p' = 2p'' + 1$). 3. Επομένως, οι τιμές $p-1$ και $q-1$ είναι της μορφής:

$$p - 1 = 2^2 p'' \quad \text{και} \quad q - 1 = 2^2 q'',$$

όπου p'' και q'' είναι πρώτοι και $p'' \equiv 1 \pmod 4$. Αυτό οδηγεί στο μέγιστο δυνατό $\text{lcm}(p-1, q-1)$ και κατά συνέπεια στη μέγιστη περίοδο της γεννήτριας.

Άσκηση 9:

(α) Κατασκευή της γεννήτριας με κατάλληλο s_0

Για την υλοποίηση της γεννήτριας: 1. Επιλέγουμε δύο SafeSafe primes p και q 20 δυαδικών ψηφίων. 2. Υπολογίζουμε το $n = pq$. 3. Βρίσκουμε αρχική κατάσταση s_0, τέτοια ώστε s_0 να είναι τετράγωνο υπόλοιπο $\pmod n$ (δηλαδή $gcd(s_0, n) = 1$ και $s_0 \neq 0$).

(β) Προσομοίωση της γεννήτριας και επαλήθευση της περιόδου

Η γεννήτρια λειτουργεί ως εξής: 1. Υπολογίζουμε διαδοχικά $s_{i+1} = s_i^2 \pmod{n}$. 2. Παράγουμε το bit $b_i = s_i \pmod 2$ για κάθε i. 3. Επαληθεύουμε πειραματικά την περίοδο επαναλαμβάνοντας τη διαδικασία μέχρι το s_i να επιστρέψει στην αρχική του τιμή s_0.

Προγραμματισμός: Το ακόλουθο πρόγραμμα σε Python υλοποιεί τη γεννήτρια και ελέγχει την περίοδο της.

```
import sympy
from math import gcd

# E    SafeSafe primes
def generate_safe_safe_prime(bits):
    while True:
        p_double_prime = sympy.randprime(2**(bits-4), 2**(bits-3))
        if p_double_prime % 4 == 1:
            p_prime = 2 * p_double_prime + 1
            if sympy.isprime(p_prime):
                p = 2 * p_prime + 1
                if sympy.isprime(p):
                    return p

# K     Blum integer
p = generate_safe_safe_prime(20)
q = generate_safe_safe_prime(20)
n = p * q

# K           s0
s0 = 2
while gcd(s0, n) != 1:
    s0 += 1

# Π
def bbs_generate(n, s0, iterations):
    state = s0
    bits = []
    for _ in range(iterations):
        state = (state**2) % n
        bits.append(state % 2)
    return bits, state

# E
def verify_period(n, s0):
```

```
    seen_states = {}
    state = s0
    period = 0
    while state not in seen_states:
        seen_states[state] = period
        state = (state**2) % n
        period += 1
    return period

# E
bits, _ = bbs_generate(n, s0, 100)
period = verify_period(n, s0)
print(f"Π    : {period}")
```

Συμπέρασμα: Η πειραματική περίοδος της γεννήτριας ταυτίζεται με τη θεωρητική μέγιστη τιμή $\frac{(p-1)(q-1)}{2}$, επαληθεύοντας την ορθότητα του ισχυρισμού.

Άσκηση 10

Ανάλυση Ανθεκτικότητας Συνάρτησης Κατακερματισμού

1. Ανθεκτικότητα σε Συγκρούσεις (Collision Resistance)

Έστω ότι η συνάρτηση κατακερματισμού H είναι ανθεκτική σε συγκρούσεις (collision resistant). Αυτό σημαίνει ότι είναι δύσκολο να βρεθούν διαφορετικά m, m' ώστε:

$$H(m) = H(m').$$

Η ανθεκτικότητα σε συγκρούσεις συνεπάγεται επίσης ανθεκτικότητα στο πρώτο όρισμα (preimage resistance). Δηλαδή, για κάθε $y \in Y$, είναι δύσκολο να βρεθεί $m \in X$ ώστε:

$$H(m) = y.$$

Θεωρούμε την περίπτωση $y = 0$ και υποθέτουμε ότι $m = x \oplus x$, όπου $\oplus$ δηλώνει τη bitwise XOR πράξη. Τότε, έχουμε:

$$H(m) = H(x) \oplus H(x) = 0.$$

Αυτό σημαίνει ότι η H δεν είναι ανθεκτική στο πρώτο όρισμα, το οποίο έρχεται σε αντίθεση με την υπόθεση ότι η H είναι collision resistant. Επομένως, καταλήγουμε ότι η H δεν είναι ανθεκτική σε συγκρούσεις.

2. Συνδυασμένη Συνάρτηση Κατακερματισμού

Θεωρούμε τη συνάρτηση κατακερματισμού:

$$H(x) = H_1(x)||H_2(x)||H_3(x),$$

όπου $||$ δηλώνει τη συνένωση των εξόδων. Αν x, x' προκαλούν σύγκρουση στη H, τότε ισχύει:

$$H(x) = H(x') \implies H_1(x)||H_2(x)||H_3(x) = H_1(x')||H_2(x')||H_3(x').$$

Δεδομένου ότι οι H_1, H_2, H_3 παράγουν εξόδους ίσου μήκους, πρέπει να ισχύει:

$$H_1(x) = H_1(x'), \quad H_2(x) = H_2(x'), \quad H_3(x) = H_3(x').$$

Αυτό σημαίνει ότι τα x, x' προκαλούν συγκρούσεις και στις επιμέρους συναρτήσεις H_1, H_2, H_3. Συνεπώς, αν οποιαδήποτε από τις H_1, H_2, H_3 είναι ανθεκτική σε συγκρούσεις, τότε και η H είναι ανθεκτική σε συγκρούσεις.

Άσκηση 11

Θεωρούμε την περίπτωση όπου το ύψος του Merkle tree είναι $h = 3$. Υποθέτουμε ότι η συνάρτηση κατακερματισμού H δεν είναι ανθεκτική σε συγκρούσεις (collision resistant). Τότε, υπάρχει ένας αντίπαλος A που μπορεί, σε πολυωνυμικό χρόνο (PPT), να βρει δύο ακολουθίες $x_0, x_1, \ldots, x_8$ και $x'_0, x'_1, \ldots, x'_8$, με:

$$x_0, x_1, \ldots, x_8 \neq x'_0, x'_1, \ldots, x'_8,$$

ώστε:

$$H(x_0, x_1, \ldots, x_8) = H(x'_0, x'_1, \ldots, x'_8).$$

Σύμφωνα με τον ορισμό του Merkle tree, η συνάρτηση κατακερματισμού H υπολογίζεται ως εξής:

$$H(x_0, x_1, \ldots, x_8) = H_1(H_1(H_1(x_0, x_1), H_1(x_2, x_3)), H_1(H_1(x_4, x_5), H_1(x_6, x_7))).$$

Επομένως, η παραπάνω σχέση γράφεται:

$$H_1(H_1(H_1(x_0, x_1), H_1(x_2, x_3)), H_1(H_1(x_4, x_5), H_1(x_6, x_7))) = H_1(H_1(H_1(x'_0, x'_1), H_1(x'_2, x'_3)), H_1(H_1(x'_4, x'_5), H$$

Ορίζουμε τους ενδιάμεσους υπολογισμούς:

$$\alpha = H_1(x_0, x_1), \quad \beta = H_1(x_2, x_3), \quad \gamma = H_1(x_4, x_5), \quad \delta = H_1(x_6, x_7),$$

$$\epsilon = H_1(\alpha, \beta), \quad \zeta = H_1(\gamma, \delta),$$

$$\alpha' = H_1(x'_0, x'_1), \quad \beta' = H_1(x'_2, x'_3), \quad \gamma' = H_1(x'_4, x'_5), \quad \delta' = H_1(x'_6, x'_7),$$

$$\epsilon' = H_1(\alpha', \beta'), \quad \zeta' = H_1(\gamma', \delta').$$

Έτσι, η σχέση γίνεται:

$$H_1(H_1(\epsilon, \zeta)) = H_1(H_1(\epsilon', \zeta')).$$

Από τη σχέση αυτή, ο αντίπαλος A μπορεί να βρει ζεύγη (ϵ, ζ) και (ϵ', ζ'), με:

$$(\epsilon, \zeta) \neq (\epsilon', \zeta'),$$

τέτοια ώστε:

$$H_1(\epsilon, \zeta) = H_1(\epsilon', \zeta').$$

Αυτό σημαίνει ότι η συνάρτηση H_1 δεν είναι ανθεκτική σε συγκρούσεις, που είναι άτοπο. Επομένως, η H είναι ανθεκτική σε συγκρούσεις. Η απόδειξη μπορεί να γενικευτεί για οποιοδήποτε h.

Άσκηση 12

Περίπτωση 1

Θεωρούμε έναν αντίπαλο A και έναν πρόκληση (challenger) CS. Το παιχνίδι $IND{-}CPA$ εκτελείται ως εξής:

- Ο A στέλνει στον CS ένα μήνυμα m. Ο CS του επιστρέφει το κρυπτογράφημα και ο A υπολογίζει ένα κλειδί k'. Με μη αμελητέα πιθανότητα p, ισχύει $k = k'$, όπου k είναι το πραγματικό κλειδί που χρησιμοποιεί ο CS.
- Ο A στέλνει στον CS δύο μηνύματα m_0, m_1.
- Ο CS απαντά με το κρυπτογράφημα c, το οποίο είναι είτε το m_0 είτε το m_1, επιλέγοντας τυχαία ένα bit b.
- Ο A υπολογίζει τα c_0 και c_1, δηλαδή τις κρυπτογραφήσεις των m_0 και m_1 με το k'. Αν $c_0 = c$, τότε $b' = 0$. Αν $c_1 = c$, τότε $b' = 1$. Αν διαφέρει και από τα δύο, ο A επιλέγει τυχαία b'.

Η πιθανότητα επιτυχίας του A, δηλαδή $\Pr[b = b']$, είναι:

$$\Pr[IND{-}CPA(A) = 1] = \Pr[b = b'] = \frac{p}{2} + \frac{p}{2} + \frac{1-p}{2} = \frac{1}{2} + \frac{p}{2}.$$

Η πιθανότητα αυτή είναι μη αμελητέα, συνεπώς το CS δεν είναι ασφαλές υπό CPA.

Περίπτωση 2

Θεωρούμε έναν αντίπαλο A και έναν challenger C. Ορίζουμε τον A ως εξής:

- Ο A ζητάει από τον C την κρυπτογράφηση του μηνύματος $m = 0^{n-1}1$. Ο C επιστρέφει το κρυπτογράφημα c και το IV.

- Αν το IV τελειώνει σε 0, τότε ο A επιστρέφει τυχαία b'.
- Αν το IV τελειώνει σε 1, τότε ο A στέλνει δύο μηνύματα $m_0 = 0^n$ και m_1 τυχαίο. Ο C επιστρέφει το challenge c' και το $IV + 1$. Αν $c = c'$, τότε $b' = 0$. Διαφορετικά, $b' = 1$.

Ανάλυση πιθανότητας επιτυχίας του A:

$$\Pr[b = b'] = \frac{1}{2} + \frac{1}{4} = \frac{3}{4}.$$

Η πιθανότητα επιτυχίας είναι μη αμελητέα, συνεπώς το CS δεν είναι ασφαλές υπό CPA.

Περίπτωση 3

Θεωρούμε έναν αντίπαλο A και έναν challenger C. Ορίζουμε τον A ως εξής:

- Ο A ζητάει από τον C δύο μηνύματα $m_0 = 0^n$ και $m_1 = 1^n$ και λαμβάνει το (IV, c).
- Ο A ζητάει αποκρυπτογράφηση του $c' = 0^n$ με IV. Ο C επιστρέφει το m'.
- Ο A υπολογίζει $m_b = c \oplus m'$. Αν $m_b = m_0$, τότε $b' = 0$, διαφορετικά $b' = 1$.