ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Λύσεις $2^{ης}$ Σειράς Ασκήσεων για το μάθημα “Υπολογιστική Κρυπτογραφία”

Δημήτριος Βασιλείου
03119830
$9^{ο}$ Εξάμηνο

ΑΣΚΗΣΗ 1

Ψάχνουμε τη λύση της εξίσωσης $x^2 \equiv y \bmod pq$. Γνωρίζουμε ότι το x είναι τετραγωνικό υπόλοιπο $mod\ pq$ οπότε υπάρχει $k \in Z_{pq}^*$ τέτοιο ώστε $k^2 \equiv x \bmod pq$. Όμως αφού $\gcd(p,q) = 1$ θα είναι $k^2 \equiv x \bmod p$ και $k^2 \equiv x \bmod q$. Όμως, από κριτήριο Euler, αφού x τετραγωνικό υπόλοιπο $mod\ p$ και $mod\ q$, έχουμε ότι $x^{(p-1)/2} \equiv 1 \bmod p \Rightarrow x^{(p-1)(q-1)/4} \equiv 1 \bmod p$ και $x^{(q-1)/2} \equiv 1 \bmod q \Rightarrow x^{(p-1)(q-1)/4} \equiv 1 \bmod q$. Άρα και $x^{(p-1)(q-1)/4} \equiv 1 \bmod pq \Rightarrow x^{\frac{(p-1)(q-1)}{4}+1} \equiv x \bmod pq \Rightarrow x^{\frac{(p-1)(q-1)+4}{4}} \equiv x \bmod pq \Rightarrow x^{2\frac{(p-1)(q-1)+4}{8}} \equiv x \bmod pq\ (1)$

Είναι $x^2 \equiv y \bmod pq \Rightarrow x^{2\frac{(p-1)(q-1)+4}{8}} \equiv y^{\frac{(p-1)(q-1)+4}{8}} \bmod pq \overset{(1)}{\Rightarrow} y^{\frac{(p-1)(q-1)+4}{8}} \equiv x \bmod pq \Rightarrow$

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

ΑΣΚΗΣΗ 2

Θεωρούμε ότι ο αντίπαλος γνωρίζει ζεύγη m_i, c_i, συγκεκριμένα 2 τέτοια ζεύγη. Επίσης θεωρούμε ως $D_{k_1}(c_i)$ την αποκρυπτογράφηση με κλασσικό DES, η οποία θα δώσει $m_i \oplus k_2$. Άρα, ισχύει για τα δύο ζεύγη m_1, c_1 και m_2, c_2:

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

με γνωστά τα m_1, m_2, c_1, c_2 και άγνωστα τα $D_{k_1}(c_1), D_{k_1}(c_2)$. Ο αντίπαλος εκτελεί τον παρακάτω αλγόριθμο

Για κάθε πιθανό k_1:
- *Υπολόγισε* $D_{k_1}(c_1)$ *με γνωστά* c_1, k_1
- *Υπολόγισε* $D_{k_1}(c_2)$ *με γνωστά* c_2, k_1
- *Αν* $D_{k_1}(c_1) \oplus D_{k_1}(c_2) = m_1 \oplus m_2$ *τότε:*
  - *Υπολόγισε* $k_2 = m_1 \oplus D_{k_1}(c_1)$
  - *Επέστρεψε* k_1, k_2
- *Αλλιώς, επανάλαβε*

Ο παραπάνω αλγόριθμος εκτελεί στη χειρότερη περίπτωση 2^{56} επαναλήψεις (αφού το κλειδί είναι μήκους $56\,bits$) και βρίσκει τα κλειδιά k_1, k_2. Στη χειρότερη περίπτωση θα κάνει $2 \cdot 2^{56}$ αποκρυπτογραφήσεις. Στην περίπτωση κλασικού DES, η brute force επίθεση πάλι διατρέχει όλα τα κλειδιά και κάνει συνολικά 2^{56} αποκρυπτογραφήσεις στη χειρότερη περίπτωση. Ασυμπτωτικά μιλώντας λοιπόν, η παραλλαγή του $DES - X$ δεν προσφέρει περισσότερη ασφάλεια από τον κλασσικό DES.
(μπορεί να μην βρίσκει με την πρώτη το k1 και να θέλει να ξανατρέξει τον κλασσικό DES μετά έχοντας βρει μόνο το k2)

ΑΣΚΗΣΗ 3

1).
Θεωρούμε ότι έχουμε DES 2 γύρων. Ορίζουμε ως $f_i = F(k_i, R)$ τη συνάρτηση που ενεργεί στο δεξί μισό, με κλειδί k_i στον i –οστό γύρο. Θεωρούμε αρχικά $Feistel_{f_1 f_2}$ ένα δίκτυο Feistel δύο γύρων που χρησιμοποιεί στον πρώτο γύρο τη συνάρτηση f_1 και στο δεύτερο γύρο f_2. Θα δείξουμε ότι αν $Feistel_{f_1 f_2}(L_0, R_0) = (L_2, R_2)$ τότε

$$Feistel_{f_2 f_1}(R_2, L_2) = (R_0, L_0)$$

- Για $Feistel_{f_1 f_2}(L_0, R_0)$:
  $L_1 = R_0$
  $R_1 = L_0 \oplus f_1(R_0)$
  $L_2 = R_1 = L_0 \oplus f_1(R_0) \;\; (1)$
  $R_2 = L_1 \oplus f_2(R_1) = R_0 \oplus f_2(L_2) \;\; (2)$

- Υπολογίζουμε $Feistel_{f_2 f_1}(R_2, L_2)$ θέτοντας $L'_0 = R_2$ και $R'_0 = L_2$:
  $L_1{}' = R_0{}'$
  $R_1{}' = L_0{}' \oplus f_2(R_0{}')$
  $L_2{}' = R_1{}' = L_0{}' \oplus f_2(R_0{}') \;\; (3)$
  $R_2{}' = L_1{}' \oplus f_1(R_1) = R_0{}' \oplus f_1(L_2{}') \;\; (4)$

Άρα για $L'_0 = R_2$ και $R'_0 = L_2$ είναι από (3), (4):

$$L'_2 = R_2 \oplus f_2(L_2) \quad (5)$$
$$R'_2 = L_2 \oplus f_1(L'_2) \quad (6)$$

Από (2) είναι $R_0 = R_2 \oplus f_2(L_2) \overset{5}{\Rightarrow} L'_2 = R_0$. Άρα από (6) είναι $R'_2 = L_2 \oplus f_1(R_0)$. Όμως από (1) είναι $L_0 = L_2 \oplus f_1(R_0)$. Άρα $R'_2 = L_0$. Τελικά $L'_2 = R_0$ και $R'_2 = L_0$ και $Feistel_{f_2 f_1}(R_2, L_2) = (R_0, L_0)$.

Θεωρούμε τώρα το $DES_{f_1 f_2}$ το οποίο χρησιμοποιεί το $Feistel_{f_1 f_2}$ και στο τέλος αντιμεταθέτει το L_2 με το R_2. Δηλαδή, είναι

$$DES_{f_1 f_2}(L_0, R_0) = (R_2, L_2)$$

Δίνουμε στο $DES_{f_1 f_2}$ ως είσοδο το $(L'_0, R'_0) = (R_2, L_2)$. Είναι
$DES_{f_2 f_1}(L'_0, R'_0) = DES_{f_2 f_1}(R_2, L_2) = Swap\big(Feistel_{f_2 f_1}(R_2, L_2)\big) = Swap(R_0, L_0) = (L_0, R_0)$ όπου $Swap$ απλή συνάρτηση που αντιμεταθέτει το δεξί μισό με το αριστερό μιας συμβολοσειράς.
Με παρόμοιο τρόπο αποδεικνύουμε ότι το ίδιο ισχύει και όταν έχουμε DES 16 γύρων.

Δείξαμε λοιπόν με βάση τα παραπάνω το εξής:

$$DES_{f_{16} \dots f_1}\left(DES_{f_1 \dots f_{16}}(L_0, R_0)\right) = DES_{f_{16} \dots f_1}(R_2, L_2) = (L_0, R_0)$$

Όταν όμως τα κλειδιά σε κάθε γύρο είναι ίδια, σε κάθε γύρο εφαρμόζεται η ίδια συνάρτηση f. Δηλαδή, είναι

$$DES_{f \dots f}\left(DES_{f \dots f}(L_0, R_0)\right) = (L_0, R_0)$$

Συνεπώς, ασθενές είναι ένα κλειδί όταν σε κάθε γύρο, τα υποκλειδιά που προκύπτουν είναι ίδια.

Με βάση τον τρόπο παραγωγής των κλειδιών, βλέπουμε πως κάθε φορά το νέο κλειδί k_{i+1} είναι

$$P(L_{k_i})||\, P'(R_{k_i})$$

όπου P, P' μεταθέσεις (χωρίς να μας ενδιαφέρει τι κάνουν οι μεταθέσεις) L_{k_i}το αριστερό μισό του k_iκαι R_{k_i} το δεξί μισό. Έτσι, αν τα L_{k_i} και R_{k_i} αποτελούνται μόνο από 0 ή από 1, τότε $k_i = k_{i+1}$, δηλαδή το κλειδί παραμένει ίδιο σε κάθε γύρο. Τελικά, τα 4 ασθενή κλειδιά είναι τα:

$$1^{56}, \qquad 0^{28}||1^{28}, \qquad 1^{28}||0^{28}, \qquad 0^{56}$$

2).

Γνωρίζουμε ότι στο DES το κλειδί του $i - οστού$ γύρου K_i προκύπτει ως εξής:

- $K_L||K_R = P_1(K)$
- $K_i = P_2(K_L \ll n_i, K_R \ll n_i)$

όπου K_L, K_R είναι το αριστερό και δεξί μισό του $P_1(K)$, όπου P_1 και P_2 είναι μεταθέσεις που απλώς μετακινούν τα bits με fixed τρόπο (δεν μας ενδιαφέρει ακριβώς τι κάνουν) και $\ll n_i$ δηλώνει bit rotation με βάση τον εξής κανόνα:

$$(n_0, n_1, \dots, n_{15}) = (1, 2, 4, 6, 8, 10, 12, 14, 15, 17, 19, 21, 23, 25, 27, 28)$$

Θα εξηγήσουμε ότι semi-weak είναι ένα κλειδί όταν τα K_L, K_R επαναλαμβάνονται με περίοδο 2, συγκεκριμένα όταν

$$K_L, K_R \in \{000000\dots00, \quad 010101\dots01, \quad 101010\dots10, \quad 111111\dots11\}.$$

Εξαιρούμε από τους παραπάνω συνδυασμούς όσους δίνουν weak key.
Έστω πχ ένα semi-weak key (μη λαμβάνοντας υπόψιν τις μεταθέσεις P_1, P_2) $K = 0\dots0||0101\dots1$.
Το συμμετρικό του K θα είναι το $K' = 0\dots0||1010\dots0$. Τα key schedule των K, K' είναι τα εξής:

$0\dots0||1010\dots0$
$0\dots0||0101\dots1$
$0\dots0||0101\dots1$
$0\dots0||0101\dots1$
$0\dots0||0101\dots1$
$0\dots0||0101\dots1$
$0\dots0||0101\dots1$
$0\dots0||0101\dots1$
$0\dots0||1010\dots0$
$0\dots0||1010\dots0$
$0\dots0||1010\dots0$
$0\dots0||1010\dots0$
$0\dots0||1010\dots0$
$0\dots0||1010\dots0$
$0\dots0||1010\dots0$
$0\dots0||0101\dots1$

$0\dots0||0101\dots1$
$0\dots0||1010\dots0$
$0\dots0||1010\dots0$
$0\dots0||1010\dots0$
$0\dots0||1010\dots0$
$0\dots0||1010\dots0$
$0\dots0||1010\dots0$

$0 \dots 0||1010 \dots 0$
$0 \dots 0||0101 \dots 1$
$0 \dots 0||0101 \dots 1$
$0 \dots 0||0101 \dots 1$
$0 \dots 0||0101 \dots 1$
$0 \dots 0||0101 \dots 1$
$0 \dots 0||0101 \dots 1$
$0 \dots 0||0101 \dots 1$
$0 \dots 0||1010 \dots 0$

Παρατηρούμε ότι το key schedule του K' είναι ίδιο με του K αλλά αντεστραμμένο. Γνωρίζοντας ότι στο DES η αποκρυπτογράφηση είναι ίδια με την κρυπτογράφηση, αλλά με αντεστραμμένο key schedule, μπορούμε να συμπεράνουμε ότι $DES^{-1}{}'_k = DES_k$. Το ίδιο προκύπτει και για τους υπόλοιπους συνδυασμούς των K_L, K_R.
Τελικά 4 semi-weak keys του DES είναι τα:
$0 \dots 0||0101 \dots 1$, $1 \dots 1||0101 \dots 1$, $0101 \dots 1||0 \dots 0$, $0101 \dots 1||1 \dots 1$. Συμμετρικά τους, θα είναι αντίστοιχα τα
$0 \dots 0||1010 \dots 0$, $1 \dots 1||1010 \dots 0$, $1010 \dots 0||0 \dots 0$, $1010 \dots 0||1 \dots 1$.

ΑΣΚΗΣΗ 4

1).
Θεωρούμε ότι το κρυπτοσύστημα που εξετάζουμε εφαρμόζει συνάρτηση E η οποία είναι ένα προς ένα. Από τη λειτουργία σε CBC έχουμε:

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

συνεπώς μπορούμε να εξάγουμε πληροφορία για το plaintext.

2).
Ορίζουμε ως $NoColl_n$ το ενδεχόμενο να μην έχουμε σύγκρουση στα $\{y_1, y_2, \dots, y_n\}$. Γνωρίζουμε ότι

$$\Pr[NoColl_n] = \frac{(x-1)\dots(x-n+1)}{x^n} \leq e^{\frac{-n(n-1)}{2x}}$$

Για $x = 2^{64}$ παίρνουμε

$$\Pr[NoColl_n] \leq e^{\frac{-n(n-1)}{2 \cdot 2^{64}}}$$

Άρα $\Pr[Coll_n] = 1 - \Pr[NoColl_n] \geq 1 - e^{\frac{-n(n-1)}{2^{65}}}$

3).
Για να είναι χρήσιμη η επίθεση θεωρούμε ότι θέλουμε $\Pr[Coll_n] \geq {}^1/_2$

Άρα

$$1 - e^{\frac{-n(n-1)}{2^{65}}} \geq \frac{1}{2}$$

Λύνοντας την ανίσωση ως προς n ότι $n \geq 0.8325\sqrt{2^{65}} + 1$.

ΑΣΚΗΣΗ 5

1).

Η ιδέα του αλγορίθμου είναι να ξεκινήσουμε με m μεγέθους ενός χαρακτήρα και να αυξάνουμε κάθε φορά κατά 1 το μέγεθος του m. Κάθε φορά ελέγχουμε το μέγεθος του ciphertext. Σε κάποιο σημείο, όταν «γεμίσει» ένα block και πάμε στο επόμενο, θα παρατηρήσουμε αλλαγή στο μέγεθος του ciphertext. Αν αυτό συμβεί στο i –οστό iteration, παίρνουμε

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

Ο αλγόριθμος σε ψευδοκώδικα περιγράφεται παρακάτω:

```
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).

Έστω l το $block\ size$, που είναι πλέον γνωστό από το προηγούμενο ερώτημα. Φτιάχνουμε μήνυμα

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

όπου $length(m_1) = length(m_2) = l$. Αν το oracle χρησιμοποιεί ECB mode τότε τα δύο block m_1 θα δώσουν ίδιο ciphertext c_1.Υπάρχει βέβαια και η περίπτωση, το oracle να μη χρησιμοποιεί ECB mode, και το m_1 να δώσει ίδιο ciphertext και τις δύο φορές. Αυτό θα μπορούσε να συμβεί αν για παράδειγμα το oracle χρησιμοποιεί CBC mode και υπάρξει collision. Ωστόσο, για μικρά μηνύματα, η πιθανότητα να συμβεί κάτι τέτοιο είναι αμελητέα.

3).

ΑΣΚΗΣΗ 6

Εξετάζουμε τη φάση της παραγωγής τυχαίων bytes (PRGA)

➔ <u>first iteration</u>
$i = 0, j = 0$
$i = 1, j = 0 + P[1] = P[1] \neq 2$
$swap\ (P[1], P[a])$
$P[1] \leftarrow P[a] = P[P[1]]$
$P[a] \leftarrow a$
$K_0 = P[(P[1] + P[a])\ mod\ 256] = (P[1] + a)\ mod\ 256$

➔ <u>second iteration</u>
$i = 2, j = a + P[1] = a$
$swap(P[2], P[a])$
$P[2] \leftarrow P[a] = a$
$P[a] \leftarrow P[2] = 0$
$K_1 = P[(P[2] + P[a])\ mod\ 256] = P[a] = 0$

Άρα $Pr[K_1 = 0] = 1$, αν $P[2] = 0\ και\ P[1] \neq 2\ μετά\ από\ KSA$

Ορίζουμε το ενδεχόμενο $Q = P[2] = 0\ και\ P[1] \neq 2$. Ψάχνουμε την πιθανότητα το δεύτερο byte να ισούται με μηδέν, δηλαδή την $Pr\ [K_1 = 0]$. Είναι

$$Pr[k_1 = 0] = Pr[K_1 = 0 \mid Q] \cdot Pr\ [Q] + Pr[K_1 = 0 \mid \neg Q] \cdot Pr\ [\neg Q] =$$
$$1 \cdot P\,r[Q] + \frac{1}{256} \cdot (1 - P\,r[Q])\ (1)$$

Θεωρούμε ότι η μετάθεση P ακολουθεί ομοιόμορφη κατανομή. Είναι:

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

Άρα από την (1) έχουμε

$$Pr[K_1 = 0] = \frac{1}{256} + \frac{1}{256}\left(1 - \frac{1}{256}\right) \simeq \frac{1}{256} + \frac{1}{256} = \frac{2}{256} = 2^{-7}$$

ΑΣΚΗΣΗ 7

1). $F_1(k, x) = F(k, x) || 0$
Έστω διαχωριστής D που ρωτά μαντείο O για $x_1, x_2, \ldots, x_n$ με $O(x_i) = y_i$ και βγάζει αποτέλεσμα 1 όταν το y_i τελειώνει σε 0 για όλα τα $y_1, y_2, \ldots, y_n$.

- Αν $O = F_{1_k}$ τότε ο D επιστρέφει 1 με πιθανότητα 1. Είναι δηλαδή
$$\Pr\left[D^{\ F_{1k}(\cdot)}(1^n) = 1\right] = 1$$
- Αν $O = g$ όπου g τυχαία συνάρτηση ομοιόμορφα επιλεγμένη, τότε ο D επιστρέφει 1 με πιθανότητα
$$\frac{1}{2}\cdot\frac{1}{2}\cdot \ldots \cdot\frac{1}{2} = \frac{1}{2^n}$$
Είναι δηλαδή
$$\Pr\left[D^{g(\cdot)}(1^n) = 1\right] = \frac{1}{2^n}$$

Είναι
$$\left|1 - \frac{1}{2^n}\right| > negl(n)$$
όταν το n είναι αρκετά μεγάλο, άρα η F_{1_k} δεν είναι ψευδοτυχαία.

2).
$F_2(k,x) = F(k,x) \oplus x$
Έστω ότι η F_2 δεν είναι ψευδοτυχαία. Τότε υπάρχει διαχωριστής D_{f2} που «διαχωρίζει» με μη αμελητέα πιθανότητα την F_2 από μια τυχαία συνάρτηση f_2. Θα δείξουμε ότι υπάρχει τότε και διαχωριστής D_f που διαχωρίζει με μη αμελητέα πιθανότητα την F από μια τυχαία συνάρτηση f και θα καταλήξουμε σε άτοπο.
Ο D_f έχει πρόσβαση σε μαντείο O_f που είναι είτε η F είτε μια τυχαία συνάρτηση f. Κατασκευάζουμε τον D_f ως εξής: Όταν τρέχει ο D_{f2} και ζητά ένα x, ο D_f «απαντάει» παριστάνοντας το μαντείο, με $O_f(x) \oplus x$. Παρατηρούμε ότι όταν $O_f = F$ είναι $O_f(x) \oplus x = F(k,x) \oplus x = F_2(k,x)$. Ο D_{f2} επιστρέφει 1 όταν $O_f(x) \oplus x = F_2(k,x)$ και ο D_f επιστρέφει 1 όταν ο D_{f2} επιστρέψει 1. Άρα είναι:
$\Pr\left[D^{F_k(\cdot)}(1^n) = 1\right] = \Pr\left[D^{F_{2k}(\cdot)}(1^n) = 1\right] = 1$ και $\Pr\left[D^{f(\cdot)}(1^n) = 1\right] = \Pr\left[D^{f \oplus x(\cdot)}(1^n) = 1\right] = 1/2^n$
Η παραπάνω διαφορά είναι μη αμελητέα, οπότε F ψευδοτυχαία, άτοπο.

3). $F_3(k,x) = F(k, x \oplus 1^n)$
Είναι $F_3(k,x) = F(k, x \oplus 1^n) = F(k, \bar{x})$. Σε κάθε x αντιστοιχεί μοναδικό $\bar{x}$ και αντίστροφα, άρα αφού F ψευδοτυχαία, το ίδιο και η F_{3_k}.

4). $F_4(k,x) = F(k,x) \mid\mid F\big(k, F(k,x)\big)$
Έστω διαχωριστής D που έχει πρόσβαση σε μαντείο O με $O(x) = y_L || y_R$. Ο D δουλεύει ως εξής: Υπολογίζει $O(x_1) = y_L || y_R$ και μετά υπολογίζει $O(y_L) = y'_L || y'_R$. Επιστρέφει 1 όταν $y'_L = y_R$.
- Αν $O = F_{4_k}$ είναι:
$$O(x_1) = F_k(x_1) || \, F_k\big(F_k(x_1)\big)$$
$$O\big(F_k(x_1)\big) = F_k\big(F_k(x_1)\big) || \, F_k\Big(F_k\big(F_k(x_1)\big)\Big)$$
Άρα, αφού το δεξί μισό του πρώτου ερωτήματος ισούται με αριστερό μισό του δεύτερου, ο D επιστρέφει 1 με πιθανότητα 1. Είναι δηλαδή $\Pr\left[D^{F_{4k}(\cdot)}(1^n) = 1\right] = 1$

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

Τελικά, είναι $\Pr\left[D^{F_{4k}(\cdot)}(1^n) = 1\right] - \Pr\left[D^{g(\cdot)}(1^n) = 1\right] = 1 - \varepsilon > negl(n)$ άρα F_{4_k} όχι ψευδοτυχαία.

ΑΣΚΗΣΗ 8

Σημειώνουμε ότι η λύση σε αυτή την άσκηση βασίστηκε στο ακόλουθο paper:

https://shub.ccny.cuny.edu/articles/1986-A_simple_unpredictable_pseudo-random_number_generator.pdf
(Το official paper της Blum Blum Shub)

(α).

Θεωρούμε $x_0 = s_0$ και ορίζουμε ως $\pi(s_0)$ την περίοδο της ακολουθίας $x_0, x_1, \ldots$ όπου $x_i = x_0^{2^i} \; mod \; N$, και $N = pq$. Εύκολα προκύπτει ότι για κάθε στοιχείο και κάθε επόμενό του είναι $x_i = x_{i+1}^2 mod \; N$. Θα χρησιμοποιήσουμε στην πορεία την επέκταση του Carmichael στο θεώρημα του Euler, η οποία λέει πως $a^{\lambda(M)} \equiv 1 \; mod \; M$ αν $\gcd(a, M) = 1$ όπου λ η συνάρτηση Carmichael. Θα δείξουμε τώρα ότι $\pi \mid \lambda(\lambda(N))$.

Έστω $ord_N x$ η τάξη ενός στοιχείου x στην ομάδα Z_N^*. Τότε $ord_N x$ περιττός, διότι:

(1). $ord_N x_i = ord_N x_{i+1}$. Για να το δείξουμε αυτό, αρχικά θα δείξουμε ότι $ord_N x_{i+1} | ord_N x_i$. Έστω $k = ord_N x_i$ οπότε $x_i^k \equiv 1 \; mod \; N$ και $(x_i^2)^k \equiv 1 \; mod \; N$. Επίσης $x_{i+1}^k \equiv x_i^{2k} mod \; N$ άρα $x_{i+1}^k \equiv 1 \; mod \; N \; (*)$.

Έστω τώρα $l = ord_N x_{i+1}$ με $x_{i+1}^l \equiv 1 \; mod \; N$. Προφανώς, αφού l η τάξη του x_{i+1}, θα είναι $l < k$ και $l \mid k$ (από $(*)$).

(2). Για κάθε θετικό ακέραιο u, είναι $2^u \parallel ord_N x_i \Rightarrow 2^{u-1} \parallel ord_N x_{i+1}$. (Εδώ $2^u \parallel ord_N x_i$ σημαίνει ότι $2^u \mid ord_N x_i$ και 2^{u+1} δε διαιρεί το $ord_N x_i$)

Δείξαμε λοιπόν ότι $ord_N x$ περιττός που σημαίνει ότι $\gcd(2, ord_N x) = 1$. Άρα, από επέκταση Carmichael παίρνουμε ότι

$$2^{\lambda(ord_N x)} \equiv 1 \; mod \; ord_N x \; (1)$$

Επίσης, αφού π περίοδος, θα είναι με βάση τον κλειστό τύπο ότι $x_0 = x_0^{2^\pi} \; mod \; N$. Δηλαδή, τα x_i, «ολοκλήρωσαν» έναν κύκλο και ξεκινάνε πάλι από το x_0. Είναι λοιπόν $2^\pi = \alpha \cdot ord_N x_0 + 1$, για α θετικό ακέραιο αφού $x_0^{ord_N x_0} \equiv 1 \; mod \; N$, ή αλλιώς $2^\pi \equiv 1 \; mod \; ord_N x_0$. Από (1) και αφού π ο ελάχιστος ακέραιος τέτοιος ώστε $2^\pi \equiv 1 \; mod \; ord_N x_0$ μιας και είναι περίοδος, θα είναι $\pi \mid \lambda(ord_N x_0)$ (2). Όμως είναι $ord_N x_0 \mid \lambda(N)$ από ορισμό $\lambda(N)$ και τάξης, οπότε και $\lambda(ord_N x_0) \mid \lambda(\lambda(N))$. Από την (2) συμπεραίνουμε ότι $\pi \mid \lambda(\lambda(N))$. Χρησιμοποιήσαμε εδώ και την ιδιότητα της Carmichael ότι αν $a \mid b$ τότε $\lambda(a) | \lambda(b)$. (πηγή : https://en.wikipedia.org/wiki/Carmichael_function)

■

Θα εξετάσουμε τώρα τις προϋποθέσεις που πρέπει να ισχύουν ώστε η περίοδος να είναι μέγιστη και συγκεκριμένα $\pi = \lambda(\lambda(N))$. Συγκεκριμένα κάνουμε τα εξής:

(1). Διαλέγουμε N τέτοιο ώστε $ord_{\frac{\lambda(N)}{2}}(2) = \lambda(\lambda(N))$

(2). Διαλέγουμε x_0 τέτοιο ώστε $ord_N x_0 = \frac{\lambda(N)}{2}$

Είναι $x_i = x_0^{2^i}\, mod\ N$ και π περίοδος, δηλαδή ο ελάχιστος θετικό ακέραιος τέτοιος ώστε $x_0 = x_\pi = x_0^{2^\pi} mod\ N \Rightarrow x_0^{2^\pi - 1} mod\ N = 1$. Από (2) $\lambda(N)/2$ είναι ο ελάχιστος θετικός ακέραιος τέτοιος ώστε $x_0^{\frac{\lambda(N)}{2}}\ mod\ N = 1$ άρα $\lambda(N)/2 \mid 2^\pi - 1$ οπότε $2^\pi mod\ (\lambda(N)/2) = 1$ $(*)$
Δείχνουμε τώρα ότι $\lambda\big(\lambda(N)\big) \mid \pi$. Από (1), $\lambda\big(\lambda(N)\big)$ είναι ο ελάχιστος θετικός ακέραιος ώστε

$$2^{\lambda(\lambda(N))} mod\ \frac{\lambda(N)}{2} = 1$$

άρα από την $(*)$ προκύπτει $\lambda\big(\lambda(N)\big) \mid \pi$.
Τελικά, όταν ισχύουν οι (1), (2) παραπάνω, η περίοδος είναι $\pi = \lambda\big(\lambda(N)\big)$.

Εξετάζουμε τώρα τη σχέση της περιόδου με το $\gcd\,(p-1, q-1)$. Είναι

$$\lambda(N) = \lambda(pq) = lcm\big(\lambda(p), \lambda(q)\big) = lcm\ \big(\varphi(p), \varphi(q)\big) = lcm(p-1, q-1)$$

αφού p, q περιττές δυνάμεις πρώτων (βλέπε κλειστό τύπο συνάρτησης Carmichael https://en.wikipedia.org/wiki/Carmichael_function)

Επίσης, γνωρίζουμε ότι

$$\gcd(p-1, q-1) = \frac{(p-1)(q-1)}{lcm(p-1, q-1)} = \frac{(p-1)(q-1)}{\lambda(N)}$$

Όταν μικραίνει το $\gcd(p-1, q-1)$ αυξάνεται το $\lambda(N)$ και γενικά αυξάνεται και το $\lambda\big(\lambda(N)\big)$ οπότε μεγαλώνει η περίοδος της BBS και γίνεται πιο ανθεκτική σε επιθέσεις.

(β).
Η μέγιστη περίοδος ισούται όπως δείξαμε παραπάνω με $\lambda\big(\lambda(N)\big)$. Θα αποδείξουμε ότι όταν p, q είναι Safe Safe primes, ισχύει η προϋπόθεση (1).
Για $N = pq$ με p, q Safe Safe primes, είναι $\lambda(N) = \lambda(pq) = lcm\big(\lambda(p), \lambda(q)\big) = lcm\big(\varphi(p), \varphi(q)\big) = lcm(p-1, q-1) = lcm(2p', 2q') = 2p'q'$ οπότε και $\lambda\big(\lambda(N)\big) = \lambda(2p'q') = lcm\big(\lambda(2), \lambda(p'), \lambda(q')\big) = lcm\big(1, \lambda(p'), \lambda(q')\big) = lcm\big(\lambda(p'), \lambda(q')\big) = 2p''q''$ όπως και προηγουμένως αφού p', q' είναι Safe Primes. Είναι επίσης $\lambda(\lambda(N)/2))\ = \lambda(p'q') = lcm\big(\lambda(p'), \lambda(q')\big) = 2p''q'' = \lambda(\lambda(N))$. Θα εξηγήσουμε ότι $ord_{\lambda(N)/2} 2 \mid \lambda\big(\lambda(N)\big)$. Από ορισμό τάξης και συνάρτησης Carmichael θα είναι $2^{ord_{\frac{\lambda(N)}{2}} 2} \equiv 1\ mod\ (\lambda(N)/2)$ και $2^{\lambda\left(\frac{\lambda(N)}{2}\right)} \equiv 1\ mod\ (\lambda(N)/2)$ (προφανώς 2 και $\lambda(N)/2$ είναι πρώτοι μεταξύ τους, καθώς $\lambda(N)/2 = 2p'q'/2 = p'q' = (2p''+1)(2q''+1)$ που είναι περιττός). Όμως $ord_{\lambda(N)/2} 2$ είναι ο ελάχιστος ακέραιος με αυτή την ιδιότητα άρα $ord_{\lambda(N)/2} 2 \mid \lambda(\lambda(N)/2\,)) = \lambda(\lambda(N))$. Δείξαμε λοιπόν ότι $ord_{\lambda(N)/2} 2\ | 2p''q''$.
Έστω τώρα ότι $ord_{\lambda(N)/2} 2 \neq 2p''q''$. Τότε, είτε $ord_{\lambda(N)/2} 2 = p''q''$ είτε $ord_{\lambda(N)/2} 2 | 2p''$ είτε $ord_{\lambda(N)/2} 2 | 2q''$. Χωρίς βλάβη της γενικότητας θεωρούμε ότι είτε $ord_{\lambda(N)/2} 2 | 2p''$ είτε $ord_{\lambda(N)/2} 2 = p''q''$. Έχουμε τις εξής περιπτώσεις:

- <u>1η Περίπτωση</u>: $ord_{\lambda(N)/2} 2 | 2p''$. Θα είναι τότε $2^{2p''} \equiv 1\ mod\ (\lambda(N)/2)) \equiv 1\ (mod\ p'q')$. Αφού $\gcd(p', q') = 1$ θα είναι και $2^{2p''} \equiv 1\ mod\ q'$. Επίσης, είναι $2^{2q''} \equiv 2^{q'-1} \equiv 1\ mod\ q'$ από μικρό Θεώρημα του Fermat. Με βάση τα παραπάνω, αν ορίσουμε ως ord την τάξη του 2 στην ομάδα $Z_{q'}^*$, τότε, είτε $ord = 2p''$, είτε $ord = 2q''$, είτε $ord | 2p''$ και $ord | 2q''$. Αν $ord = 2p''$ τότε πρέπει $2p'' | 2q''$ που δεν ισχύει, ομοίως αν $ord = 2q''$. Άρα $ord | 2p''$ και $ord | 2q''$ και αφού p'', q'' πρώτοι, τότε $ord = 2$. Άρα $2^2 \equiv 1\ (mod\ q')$ άτοπο, διότι $q'' \geq 3$ οπότε $q' \geq 7$.

- 2η Περίπτωση: : $ord_{\lambda(N)/2}2|p''q''$. Τότε $2^{p''q''} \equiv 1 \; mod \; p'q'$ άρα αφού $\gcd(p', q') = 1$ είναι και $2^{p''q''} \equiv 1 \; mod \; q'$ οπότε $2^{q''} \not\equiv -1 \; mod \; q'$ αφού p'' περιττός (αν ήταν $2^{q''} \equiv -1 \; mod \; q'$ τότε θα είχαμε $2^{p''q''} \equiv -1 \; mod \; q'$ άτοπο). Άρα $2^{(q'-1)/2} \not\equiv -1 \; mod \; q'$ οπότε το 2 είναι τετραγωνικό υπόλοιπο $mod \; q'$. Όμοια δείχνουμε ότι το 2 είναι τετραγωνικό υπόλοιπο $mod \; p'$ άτοπο.

■

Με βάση το paper πάντα υπάρχει x_0 που ικανοποιεί την προϋπόθεση (2). Τελικά, στην περίπτωση όπου p, q είναι Safe Safe primes η μέγιστη περίοδος είναι $\lambda(\lambda(N)) = 2p''q''$.

ΑΣΚΗΣΗ 9

(α).

Θεωρούμε ότι το πρόγραμμα θα λαμβάνει δύο Safe Safe primes p, q ομοίως και τα p', q', p'', q''.
Διαλέγουμε τέτοιους αριθμούς από το ακόλουθο link:
[Cunningham Chains (tripod.com)]
Εδώ, η λίστα αποτελείται από πρώτους όπου $p_{i+1} = 2p_i + 1$. Άρα σε μια λίστα με τρία στοιχεία, το πρώτο είναι το p'', το δεύτερο το p' και το τρίτο το p. Χρειαζόμαστε δύο τέτοιες λίστες, μία για το p και μία για το q. Επίσης, αφού θέλουμε οι p, q να είναι μήκους 20 bits, σημαίνει ότι διαλέγουμε p, q στο εύρος $524288, 1048575$.
Παρατίθεται ο κώδικας για την επίλυση του ερωτήματος σε γλώσσα Python:

```
import sympy
import random

def generate_s0(p, q, p_tonos, q_tonos, p_tonos_tonos, q_tonos_tonos):
    n = p * q
    carmichael_n_dia_2 = p_tonos * q_tonos

    def find_s0_quadratic_residue(p, q, n):
        # with probability 1 / 4 s0 is quadratic residue mod pq
        s0 = random.randint(1, n - 1)

        # check if s0 is indeed a quadratic residue. if not choose another s0
        while not (pow(s0, (p - 1) // 2, p) == 1 and pow(s0, (q - 1) // 2, q) == 1 and
sympy.gcd(s0, n) == 1):
            s0 = random.randint(1, n - 1)
        return s0

    while True:
        s0 = find_s0_quadratic_residue(p, q, n)

        # order of s0 mod N must be λ(N) / 2
        if pow(s0, carmichael_n_dia_2, n) == 1:
            if not pow(s0, 2, n) == 1 and not pow(s0, p_tonos, n) == 1 and not pow(s0,
q_tonos, n) == 1:
                return s0

def program(p, q, p_tonos, q_tonos, p_tonos_tonos, q_tonos_tonos):
    n = p * q
    s0 = generate_s0(p, q, p_tonos, q_tonos, p_tonos_tonos, q_tonos_tonos)
    print("Safe Safe p : " + str(p))
    print("Safe Safe q : " + str(q))
    print("Special n : " + str(n))
    print("Seed s0 : " + str(s0))

program(571199, 786959, 285599, 393479, 142799, 196739)
```

Η συνάρτηση generate_s0 δέχεται τα p, q, p', q', p'', q'', υπολογίζει το n και το $\lambda(n)/2 = p'q'$. Στη συνέχεια ορίζεται η συνάρτηση find_s0_quadratic_residue που επιλέγει τυχαία ένα s_0 από την ομάδα Z_n^*, το οποίο βέβαια πρέπει να είναι πρώτο με το n. Με πιθανότητα $1/4$ το s_0 θα είναι τετραγωνικό υπόλοιπο $mod\ n$. Ελέγχει αν το s_0 είναι τετραγωνικό υπόλοιπο ως εξής:

- ➔ Αν το s_0 είναι τετραγωνικό υπόλοιπο $mod\ n$ θα είναι και τετραγωνικό υπόλοιπο $mod\ p$ και $mod\ q$. Αφού p, q πρώτοι από θεώρημα Euler το s_0 είναι τετραγωνικό υπόλοιπο $mod\ p$ αν και μόνο αν $s_0^{(p-1)/2} \equiv 1\ mod\ p$ (αντίστοιχα για $mod\ q$)

Αν το s_0 είναι τετραγωνικό υπόλοιπο επιστρέφεται, αλλιώς επιλέγεται ξανά τυχαίο s_0, πρώτο με το n και επαναλαμβάνεται η ίδια διαδικασία.
Έχοντας βρει ένα s_0 που είναι τετραγωνικό υπόλοιπο και πρώτο με το n πρέπει να ελεγχθεί αν η τάξη του $mod\ n$ είναι $\lambda(N)/2$. Ελέγχουμε αρχικά αν $2^{\lambda(n)/2} \equiv 1\ mod\ n$. Αν αυτό ισχύει τότε υπάρχουν δύο περιπτώσεις:

- ➔ Είτε το $\lambda(N)/2$ είναι η τάξη οπότε την έχουμε βρει.
- ➔ Είτε το $\lambda(N)/2 = p'q'$ διαιρείται από την τάξη, δηλαδή η τάξη είναι είτε p' είτε q'. Συνεπώς, ελέγχουμε αν ή $2^{p'} \equiv 1\ mod\ n$ ή $2^{q'} \equiv 1\ mod\ n$. Αν καμία ισοδυναμία δεν ισχύει, τότε πρέπει να διαλέξουμε πάλι νέο s_0 οπότε επαναλαμβάνουμε από την αρχή.

Για $p = 571199, q = 786959$ έχουμε το ακόλουθο output:

```
Safe Safe p : 571199
Safe Safe q : 786959
Special n : 449510193841
Seed s0 : 171168566799
```

(β)

Παρατίθεται ο κώδικας για την επίλυση του ερωτήματος σε γλώσσα Python:

```
import sympy
import random
import time

def generate_s0(p, q, p_tonos, q_tonos, p_tonos_tonos, q_tonos_tonos):
    n = p * q
    carmichael_n_dia_2 = p_tonos * q_tonos

    while True:
        def find_s0_quadratic_residue(p, q, n):
            # with probability 1 / 4 s0 is quadratic residue mod pq
            s0 = random.randint(1, n - 1)
            while not (pow(s0, (p - 1) // 2, p) == 1 and pow(s0, (q - 1) // 2, q) == 1
and sympy.gcd(s0, n) == 1):
                s0 = random.randint(1, n - 1)
            return s0

        # order of s0 mod N must be λ(N) / 2
        s0 = find_s0_quadratic_residue(p, q, n)

        if pow(s0, carmichael_n_dia_2, n) == 1:
            if not pow(s0, 2, n) == 1 and not pow(s0, p_tonos, n) == 1 and not pow(s0,
q_tonos, n) == 1:
                print("Safe Safe p : " + str(p))
                print("Safe Safe q : " + str(q))
                print("Special n : " + str(n))
                print("Seed s0 : " + str(s0))
```

```
                return s0
        k += 1

def blum_blum_shub(s0, p, q, p_tonos, q_tonos, p_tonos_tonos, q_tonos_tonos, 
iterations):
    print("Theoretically calculated period is π(s0) = λ(λ(n)) = 2p''q'' = " + str(2 * 
p_tonos_tonos * q_tonos_tonos))
    n = p * q

    first = s0
    second = pow(s0, 2, n)
    third = pow(second, 2, n)
    fourth = pow(third, 2, n)

    first_four_elements = [first, second, third, fourth]
    current = s0
    window = []
    period = 0
    for i in range(1, iterations):
        current = pow(current, 2, n)
        if i > 4:
            window.append(current)
        if len(window) == 4:
            if window == first_four_elements:
                period = i - 3
                break
            else:
                window = window[1:5]

    print("Programmatically calculated period is " + str(period))


def program(p, q, p_tonos, q_tonos, p_tonos_tonos, q_tonos_tonos, iterations):
    s0 = generate_s0(p, q, p_tonos, q_tonos, p_tonos_tonos, q_tonos_tonos)
    blum_blum_shub(s0, p, q, p_tonos, q_tonos, p_tonos_tonos, q_tonos_tonos, 
iterations)


program(2879, 359, 1439, 179, 719, 89, 25000000000)
```

Επεκτείνουμε τον προηγούμενο κώδικα προσθέτοντας τη συνάρτηση blum_blum_shub η οποία παίρνει ως παράμετρο το s0 που έχει επιστρέψει η generate_s0, έναν ενδεικτικό αριθμό από iterations που θα γίνουν και όσες παραμέτρους λαμβάνει και η generate_s0. Υπολογίζει την περίοδος ως εξής: Ξεκινάει με πρώτο στοιχείο το s_0 και για να βρει το επόμενο στοιχείο εφαρμόζει τον τύπο $x_{i+1} \equiv x_i^2 \; mod \; n$. Όταν τα 4 πρώτα στοιχεία είναι ίσα με τα 4 τρέχοντα (βλέπε μεταβλητή window) τότε έχουμε σίγουρα ολοκληρώσει έναν κύκλο. Θα μπορούσαμε να εξετάσουμε περισσότερα από 4, αλλά εξαιτίας της «τυχαιότητας» της bbs θεωρούμε αρκετά απίθανο να παρουσιαστούν διαδοχικά 4 ίδιες τιμές εντός του ίδιου κύκλου. Σημειώνουμε ότι η εκτέλεση του παραπάνω κώδικα, καθυστερεί πολύ για p, q μήκους 20 bits, οπότε δοκιμάσαμε με μικρότερους πρώτους, συγκεκριμένα για $p = 2879, q = 359$ παίρνουμε:

```
Safe Safe p : 2879
Safe Safe q : 359
Special n : 1033561
Seed s0 : 111231
Theoretically calculated period is π(s0) = λ(λ(n)) = 2p''q'' = 127982
Programmatically calculated period is 127982
```

οπότε η θεωρητικά υπολογιζόμενη περίοδος επαληθεύεται και πειραματικά.

ΑΣΚΗΣΗ 10

1).
Έστω ότι η H είναι collision resistant. Τότε η H έχει και αντίσταση πρώτου ορίσματος. Δηλαδή για $y \in Y$ είναι δύσκολο να βρεθεί $m \in X$ τέτοιο ώστε $H(m) = y$. Θεωρούμε $y = 0$. Αν $m = x \oplus x$ τότε $H(m) = H(x) \oplus H(x) = 0$ άρα η H δεν έχει αντίσταση πρώτου ορίσματος, <u>άτοπο</u>. Τελικά η H δεν είναι collision resistant.

2).
$H(x) = H_1(x)||H_2(x)||H_3(x)$
Θεωρούμε ότι τα x, x' προκαλούν σύγκρουση στην H. Τότε είναι

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

Αφού οι H_1, H_2, H_3 δημιουργούν συμβολοσειρές ίσου μήκους πρέπει αναγκαστικά να είναι

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

Άρα, τα x, x' προκαλούν σύγκρουση και στις H_1, H_2, H_3. Συνεπώς αν μία εκ των H_1, H_2, H_3 έχει δυσκολία εύρεσης συγκρούσεων, το ίδιο ισχύει και για την H.

ΑΣΚΗΣΗ 11

Θεωρούμε την περίπτωση όπου $h = 3$. Έστω ότι η H δεν είναι collision resistant. Τότε, υπάρχει αντίπαλος A που μέσω PPT βρίσκει $x_0x_1 \dots x_8$ και $x_0'x_1' \dots x_8'$ με $x_0x_1 \dots x_8 \neq x_0'x_1' \dots x_8'$ και $H(x_0x_1 \dots x_8) = H(x_0'x_1' \dots x_8')$. Με βάση τον ορισμό του Merkle tree είναι:

$$H(x_0x_1 \dots x_8) = H(x_0'x_1' \dots x_8') \Rightarrow$$
$$H_1\Big(H_1\big(H_1(x_0x_1)H_1(x_2x_3)\big)H_1\big(H_1(x_4x_5)H_1(x_6x_7)\big)\Big)$$
$$= H_1\Big(H_1\big(H_1(x_0'x_1')H_1(x_2'x_3')\big)H_1\big(H_1(x_4'x_5')H_1(x_6'x_7')\big)\Big) \Rightarrow$$
$$H_1\big(H_1(\alpha\beta)H_1(\gamma\delta)\big) = H_1\big(H_1(\alpha'\beta')H_1(\gamma'\delta')\big) \Rightarrow H_1(\varepsilon\zeta) = H_1(\varepsilon'\zeta')$$

με $\alpha = H_1(x_0x_1),\ \beta = H_1(x_2x_3),\ \gamma = H_1(x_4x_5),\ \delta = H_1(x_6x_7),\ \varepsilon = H_1(\alpha\beta),\ \zeta = H_1(\gamma\delta)$ και
$\alpha' = H_1(x_0'x_1'),\quad \beta' = H_1(x_2'x_3'),\quad \gamma' = H_1(x_4'x_5'),\quad \delta' = H_1(x_6'x_7'),\quad \varepsilon' = H_1(\alpha'\beta'),$
$\zeta' = H_1(\gamma'\delta')$
Άρα, ο A βρίσκει $\varepsilon\zeta, \varepsilon'\zeta'$ με $\varepsilon\zeta \neq \varepsilon'\zeta'$ τέτοια ώστε $H_1(\varepsilon\zeta) = H_1(\varepsilon'\zeta')$ οπότε η H_1 δεν είναι collision resistant, **άτοπο.** Τελικά, η H είναι collision resistant. Με παρόμοιο τρόπο η απόδειξη γενικεύεται για οποιοδήποτε h.

ΑΣΚΗΣΗ 12

1).
Θεωρούμε τον αντίπαλο A και τον challenger CS. Το παιχνίδι $IND - CPA$ λειτουργεί ως εξής:

- Ο A στέλνει στον CS ένα μήνυμα m. Λαμβάνει το κρυπτογράφημα από τον CS και υπολογίζει ένα κλειδί k'. Με μη αμελητέα πιθανότητα p είναι $k = k'$ όπου k, το πραγματικό κλειδί που χρησιμοποιεί ο CS.
- Ο A στέλνει στον CS δύο μηνύματα m_0, m_1.
- Ο CS απαντάει στον A με το c, δηλαδή την κρυπτογράφηση είτε του m_0, είτε του m_1. Η επιλογή γίνεται τυχαία επιλέγοντας ομοιόμορφα bit b από το $\{0, 1\}$.

- Ο A υπολογίζει c_0 την κρυπτογράφηση του m_0 με το κλειδί k' που έχει υπολογίσει και c_1 την αντίστοιχη κρυπτογράφηση του m_1. Αν $c_0 = c$ θέτει $b' = 0$, αλλιώς αν $c_1 = c$ θέτει $b' = 1$, αλλιώς αν το c διαφέρει και από το c_0 και από το c_1 επιλέγει ομοιόμορφα τυχαία b' και το επιστρέφει.

Ας αναλύσουμε την πιθανότητα επιτυχίας του A, δηλαδή την $\Pr[b = b']$. Είναι:

$$\begin{aligned}\Pr[IND - CPA(A) = 1] &= \Pr[b = b'] \\ &= \Pr[b = b'|c_0 = c]\Pr[c_0 = c] + \Pr[b = b'|c_1 = c]\Pr[c_1 = c] \\ &+ \Pr[b = b'|c \neq c_1, c_2]\Pr[c \neq c_1, c_2] = \\ &1 \cdot \frac{1}{2} \cdot p + 1 \cdot \frac{1}{2} \cdot p + \frac{1}{2}(1 - p) = \frac{p + 1}{2}\end{aligned}$$

που είναι μη αμελητέα, συνεπώς το CS δεν έχει ασφάλεια CPA.

2).

Θεωρούμε αντίπαλο A και challenger C. Κατασκευάζουμε τον A ως εξής:

- Ο A ρωτάει τον C για το μήνυμα $m = 0^{n-1}1$. Ο C, του επιστρέφει το κρυπτοκείμενο c για το οποίο χρησιμοποιήθηκε το initial vector IV. Ας συμβολίσουμε το ζεύγος ως (c, IV).
- Αν το IV τελειώνει σε 0 τότε ο A επιστρέφει b' τυχαίο (0 ή 1).
- Αν το IV τελειώνει σε 1 τότε ο A στέλνει μηνύματα m_0, m_1 όπου $m_0 = 0^n$ και m_1 τυχαίο μήνυμα. Ο C επιστρέφει το challenge $(c', IV + 1)$, όπου c' η κρυπτογράφηση είτε του m_0 είτε του m_1. Το ποιο μήνυμα θα κρυπτογραφηθεί επιλέγεται διαλέγοντας τυχαία, b από το $\{0, 1\}$.
- Αν $c = c'$ ο A επιστρέφει $b' = 0$, αλλιώς $b' = 1$.

Ας αναλύσουμε την πιθανότητα επιτυχίας του A. Ο A κερδίζει αν «μαντέψει» το b, δηλαδή αν $b = b'$. Είναι

$$\begin{gathered}\Pr[b = b'] = \Pr[b = b' \kappa\alpha\iota\ IV\ \tau\varepsilon\lambda\varepsilon\iota\acute{\omega}\nu\varepsilon\iota\ \sigma\varepsilon\ 1] + \Pr[b = b' \kappa\alpha\iota\ IV\ \tau\varepsilon\lambda\varepsilon\iota\acute{\omega}\nu\varepsilon\iota\ \sigma\varepsilon\ 0] = \\ \Pr[b = b' |\ IV\ \tau\varepsilon\lambda\varepsilon\iota\acute{\omega}\nu\varepsilon\iota\ \sigma\varepsilon\ 1] \cdot \Pr[IV\ \tau\varepsilon\lambda\varepsilon\iota\acute{\omega}\nu\varepsilon\iota\ \sigma\varepsilon\ 1] + \Pr[b = b' |\ IV\ \tau\varepsilon\lambda\varepsilon\iota\acute{\omega}\nu\varepsilon\iota\ \sigma\varepsilon\ 0] \\ \cdot \Pr[IV\ \tau\varepsilon\lambda\varepsilon\iota\acute{\omega}\nu\varepsilon\iota\ \sigma\varepsilon\ 0] = \\ \frac{1}{2} \cdot \frac{1}{2} + \Pr[b = b' |\ IV\ \tau\varepsilon\lambda\varepsilon\iota\acute{\omega}\nu\varepsilon\iota\ \sigma\varepsilon\ 0] \cdot \frac{1}{2} \quad (1)\end{gathered}$$

Αν IV τελειώνει σε 0 είναι $IV + 1 = IV \oplus 0^{n-1}1$. Άρα,Α

$$c = Enc(IV \oplus m) = Enc(IV \oplus 0^{n-1}1 \oplus m \oplus 0^{n-1}1) = Enc\big((IV + 1) \oplus m_0\big)$$

Οπότε αν ο C κρυπτογράφησε το m_0, είναι $c = c'$ και ο A επιστρέφει 0, αλλιώς ο C κρυπτογράφησε το m_1 οπότε ο A επιστρέφει 1. Συνεπώς,

$$\Pr[b = b' |\ IV\ \tau\varepsilon\lambda\varepsilon\iota\acute{\omega}\nu\varepsilon\iota\ \sigma\varepsilon\ 0] = 1$$

Από την (1) λοιπόν παίρνουμε

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

οπότε ο A κερδίζει με πιθανότητα $3/4$ η οποία δεν είναι αμελητέα, συνεπώς το CS δεν είναι CPA ασφαλές.

3).

Θεωρούμε αντίπαλο A και challenger C. Κατασκευάζουμε τον A ως εξής:

- Ο A ρωτάει τον C για δύο μηνύματα $m_0 = 0^n, m_1 = 1^n$ και λαμβάνει το challenge (IV, c) όπου c το κρυπτοκείμενο και IV το initial vector. Γνωρίζουμε ότι στο OFB mode ισχύει $c = Encrypt(IV) \oplus m_b$ όπου m_b είναι είτε το m_0, είτε το m_1 ανάλογα ποιο μήνυμα επέλεξε ο C να κρυπτογραφήσει. Όπως και στα προηγούμενα, η επιλογή καθορίζεται επιλέγοντας τυχαία bit b από το $\{0,1\}$.

- Ο A ζητάει την αποκρυπτογράφηση του $c' = 0^n$ με initial vector IV. Ο C επιστρέφει το $m' = Encrypt(IV) \oplus 0^n$.
- Ο A υπολογίζει το $c \oplus m' = Encrypt(IV) \oplus m_b \oplus Encrypt(IV) \oplus 0^n = m_b$. Αν $m_b = m_0$ θέτει $b' = 0$, αλλιώς αν $m_b = m_1$ θέτει $b' = 1$. Επιστρέφει το b'.

Με βάση τα παραπάνω, η πιθανότητα επιτυχίας του A είναι 1, συνεπώς το OFB mode δεν έχει ασφάλεια CCA.