ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Κρυπτογραφια

9ο Εξαμηνο

2η Σειρά Ασκήσεων

Ονοματεπώνυμο (Αριθμός Μητρώου):

Γεώργιος Γκοτζιάς (03119047)

20 Δεκεμβρίου 2023

Άσκηση 1

Αρχικά, δείχνουμε ότι ο αριθμός $x = y^{((p-1)(q-1)+4)/8} \mod n$ είναι τετραγωνική ρίζα του y.
Επειδή $y \in QR(n)$ υπάρχει λ τέτοιο, ώστε $\lambda^2 \equiv y \mod n$ και $\lambda^2 \equiv ymodp$ από κινέζικο θεώρημα υπολοίπων.
Ακόμη, από το κριτήριο Euler γνωρίζουμε ότι η ισοτιμία $k^2 \equiv y \mod p$ έχει λύση αν και μόνο αν $y^{(p-1)/2} \equiv 1 \mod p$, όμως γνωρίζουμε ότι υπάρχει η λύση $x = \lambda$, επομένως $y^{(p-1)/2} \equiv 1 \mod p$.
Έχουμε

$$\begin{aligned}
y^{(p-1)/2} &\equiv 1 \mod p \Rightarrow \\
(y^{(p-1)/2})^{(q-1)/2} &\equiv 1 \mod p \Rightarrow \\
y^{(p-1)(q-1)/4} &\equiv 1 \mod p \Rightarrow \\
y \cdot y^{(p-1)(q-1)/4} &\equiv y \mod p \Rightarrow \\
y^{1+(p-1)(q-1)/4} &\equiv y \mod p \Rightarrow \\
y^{(4+(p-1)(q-1))/4} &\equiv y \mod p \Rightarrow \\
(y^{(4+(p-1)(q-1))/8})^2 &\equiv y \mod p \Rightarrow \\
x^2 &\equiv y \mod p
\end{aligned}$$

Ομοίως, προκύπτει $x^2 \equiv y \mod q$, επομένως από κινέζικο θεώρημα υπολοίπων προκύπτει ότι $x^2 \equiv y \mod n$.
Μένει να αποδειχθεί ότι το x είναι και αυτό τετραγωνικό υπόλοιπο. Πράγματι:

$$\begin{aligned}
(x^{(4+(p-1)(q-1))/8})^2 &\equiv (x^2)^{(4+(p-1)(q-1))/8} && \mod n \\
&\equiv y^{(4+(p-1)(q-1))/8} && \mod n \\
&\equiv x && \mod n
\end{aligned}$$

Άσκηση 2

Το συγκεκριμένο σύστημα δεν παρέχει παραπάνω ασφάλεια σε σχέση με το κλασσικό DES. Αυτό, γιατί μία επίθεση KPA για το DES μπορεί να τροποποιηθεί ως εξής:
Αρχικά, αν θέλει να ελεγχθεί αν το k_1 είναι το κλειδί του DES, τότε υπολογίζουμε για κάποιο γνωστό ζεύγος plaintext-cipher m_0, c_0 την τιμή $k_2 = E_{k_1}^{-1}(c_0) \oplus m_0$, όπου το k_2 είναι η μοναδική πιθανή τιμή για το κλειδί k_2, αν το k_1 είναι πράγματι το κλειδί για DES, οπότε στα υπόλοιπα μηνύματα m_i που αντιστοιχούν σε κάποιο c_i γίνεται ο έλεγχος που θα γινόταν σε μία επίθεση κλασσικού DES για τα ζεύγη $m_i \oplus k_2, c_i$.
Για κάθε k_1 που δοκιμάζεται αντιστοιχεί σε διαφορετικό k_2 οπότε σε κάθε έλεγχο πιθανού κλειδιού υπολογίζεται νέα τιμή για το δεύτερο κλειδί.

Άσκηση 3

1. Ασθενή κλειδιά του DES είναι τα εξής 4:

$$K_0 = 0000000\_0000000\_0000000\_0000000\_0000000\_0000000\_0000000\_0000000\_$$

$$K_1 = 1111111\_1111111\_1111111\_1111111\_1111111\_1111111\_1111111\_1111111\_$$

$$K_2 = 0001111\_0001111\_0001111\_0001111\_0000111\_0000111\_0000111\_0000111\_$$

$$K_3 = 1110000\_1110000\_1110000\_1110000\_1111000\_1111000\_1111000\_1111000\_$$

Όπου ο χαρακτήρας _ δηλώνει ότι το συγκεκριμένο bit αγνοείται στο τελικό κλειδί, οπότε μπορεί να είναι οποιοδήποτε.
Τα παραπάνω κλειδιά αντιτοιχούν μετά το αρχικό Permutation στα κλειδιά:

$$k_0 = 00000000000000000000000000000000000000000000000000000000$$

$$k_1 = 11111111111111111111111111111111111111111111111111111111$$

$$k_2 = 00000000000000000000000000001111111111111111111111111111$$

$$k_3 = 11111111111111111111111111110000000000000000000000000000$$

Επομένως, τα πρώτα 28 bit του κλειδιού ταυτίζονται, οπότε παραμένουν τα ίδια μετά από οποιονδήποτε αριθμό ολισθήσεων. Ομοίως, και τα τελευταία 28 bit.
Άρα, θα είναι σε κάθε γύρο το ίδιο κλειδί, οπότε για κάθε ένα κλειδί θα ισχύει (με τον εκθέτη δηλώνεται ο γύρος):

$$k_i^1 = k_i^{16}$$
$$k_i^2 = k_i^{15}$$
$$k_i^3 = k_i^{14}$$
$$k_i^4 = k_i^{13}$$
$$k_i^5 = k_i^{12}$$
$$k_i^6 = k_i^{11}$$
$$k_i^7 = k_i^{10}$$
$$k_i^8 = k_i^9$$

Οπότε η κρυπτογράφηση ταυτίζεται με την αποκρυπτογράφηση, άρα $DES_k(DES_k(x)) = DES_k^{-1}(DES_k(x)) = x$. 2. Ημιασθενή κλειδία του DES είναι τα εξής:

$$K_0 = 1111111\_0000000\_1111111\_0000000\_1111111\_0000000\_1111111\_0000000\_$$

$$K_1 = 0000000\_1111111\_0000000\_1111111\_0000000\_1111111\_0000000\_1111111\_$$

$$K_2 = 0000000\_1111111\_0000000\_1111111\_1111111\_0000000\_1111111\_0000000\_$$

$$K_3 = 1111111\_0000000\_1111111\_0000000\_0000000\_1111111\_0000000\_1111111\_$$

Που μετά το αρχικό permutation δίνουν τα κλειδιά:

$$k_0 = 01010101010101010101010101010101010101010101010101010101$$

$$k_1 = 10101010101010101010101010101010101010101010101010101010$$

$$k_2 = 01010101010101010101010101011010101010101010101010101010$$

$$k_3 = 10101010101010101010101010100101010101010101010101010101$$

Ουσιαστικά, για τα blocks 28 bit που θα προκύψουν είναι 2 πιθανά ενδεχόμενα, τα εξής:

$$0101010101010101010101010101$$

$$1010101010101010101010101010$$

Θεωρούμε ότι $c^1 = 0101010101010101010101010101$.
Τότε για κάθε γύρο είναι :

$$c_1^1 = 1010101010101010101010101010$$
$$c_2^1 = 0101010101010101010101010101$$
$$c_3^1 = 0101010101010101010101010101$$
$$c_4^1 = 0101010101010101010101010101$$
$$c_5^1 = 0101010101010101010101010101$$
$$c_6^1 = 0101010101010101010101010101$$
$$c_7^1 = 0101010101010101010101010101$$
$$c_8^1 = 0101010101010101010101010101$$
$$c_9^1 = 1010101010101010101010101010$$
$$c_{10}^1 = 1010101010101010101010101010$$
$$c_{11}^1 = 1010101010101010101010101010$$
$$c_{12}^1 = 1010101010101010101010101010$$

$$c_{13}^1 = 1010101010101010101010101010$$
$$c_{14}^1 = 1010101010101010101010101010$$
$$c_{15}^1 = 1010101010101010101010101010$$
$$c_{16}^1 = 0101010101010101010101010101$$

Έστω ακόμη $c^2 = 1010101010101010101010101010$.
Τότε για κάθε γύρο είναι :

$$c_1^2 = 0101010101010101010101010101$$
$$c_2^2 = 1010101010101010101010101010$$
$$c_3^2 = 1010101010101010101010101010$$
$$c_4^2 = 1010101010101010101010101010$$
$$c_5^2 = 1010101010101010101010101010$$
$$c_6^2 = 1010101010101010101010101010$$
$$c_7^2 = 1010101010101010101010101010$$
$$c_8^2 = 1010101010101010101010101010$$
$$c_9^2 = 0101010101010101010101010101$$
$$c_{10}^2 = 0101010101010101010101010101$$
$$c_{11}^2 = 0101010101010101010101010101$$
$$c_{12}^2 = 0101010101010101010101010101$$
$$c_{13}^2 = 0101010101010101010101010101$$
$$c_{14}^2 = 0101010101010101010101010101$$
$$c_{15}^2 = 0101010101010101010101010101$$
$$c_{16}^2 = 1010101010101010101010101010$$

Παρατηρείται ότι:

$$c_1^1 = c_{16}^2$$
$$c_2^1 = c_{15}^2$$
$$c_3^1 = c_{14}^2$$
$$c_4^1 = c_{13}^2$$
$$c_5^1 = c_{12}^2$$
$$c_6^1 = c_{11}^2$$
$$c_7^1 = c_{10}^2$$
$$c_8^1 = c_9^2$$
$$c_9^1 = c_8^2$$
$$c_{10}^1 = c_7^2$$
$$c_{11}^1 = c_6^2$$
$$c_{12}^1 = c_5^2$$
$$c_{13}^1 = c_4^2$$
$$c_{14}^1 = c_3^2$$
$$c_{15}^1 = c_2^2$$
$$c_{16}^1 = c_1^2$$

Παρατηρείται ακόμη, ότι :

$$k_0 = c^1 || c^1$$
$$k_1 = c^2 || c^2$$
$$k_2 = c^1 || c^2$$
$$k_3 = c^2 || c^1$$

Με βάση τα προηγούμενα, αν έχουμε το κλειδί k_0 κατά την αποκρυπτογράφηση θα χρησιμοποιούταν η αλληλουχία κλειδιών του k_1, επομένως το k_0 είναι ημιασθενές κλειδί με $k' = k_1$. Αντίστοιχα, k_1 ημιασθενές με $k' = k_0$, k_2 ημιασθενές με $k' = k_3$ και k_3 ημιασθενές με k_2.

Άσκηση 4

1. Έστω $y_i = y_j$ για $i \neq j$. Έχουμε ότι:

$$\begin{aligned} y_i &= y_j \Rightarrow \\ Enc(k, x_i \oplus y_{i-1}) &= Enc(k, x_j \oplus y_{j-1}) \Rightarrow \ (Enc\ is\ 1-1) \\ x_i \oplus y_{i-1} &= x_j \oplus y_{j-1} \Rightarrow \\ x_i \oplus x_j &= y_{i-1} \oplus y_{j-1} \end{aligned}$$

Επομένως, παρατηρώντας εισόδους που συμπίπτουν μπορεί να εξαχθεί πληροφορία για το xor των εισόδως τις ίδιες στιγμές.
2. Συνολικά, υπάρχουν $(2^{64})^n$ πιθανές έξοδοι. Οι έξοδοι χωρίς συγκρούσεις είναι $2^{64} \cdot (2^{64}-1) \cdot (2^{64}-2) \cdot \ldots \cdot (2^{64} - n+1)$. Επομένως, η πιθανότητα να μην υπάρχει σύγκρουση είναι $p_{NoColl} = \frac{2^{64} \cdot (2^{64}-1) \cdot (2^{64}-2) \cdot \ldots \cdot (2^{64}-n+1)}{(2^{64})^n}$. Το αποτέλεσμα δεν επηρεάζεται από την εξάρτηση από τις προηγούμενες εξόδους, καθώς για οποιαδήποτε προηγούμενη έξοδο θα υπάρχει μοναδική είσοδος που να δίνει την επιθυμητή έξοδο.
Τελικά, η πιθανότητα να υπάρχει σύγκρουση είναι:

$$\begin{aligned} p_{Coll} &= 1 - p_{NoColl} \\ &= 1 - \frac{2^{64} \cdot (2^{64}-1) \cdot (2^{64}-2) \cdot \ldots \cdot (2^{64}-n+1)}{(2^{64})^n} \\ &= 1 - \left(1 - \frac{1}{2^{64}}\right) \cdot \left(1 - \frac{2}{2^{64}}\right) \cdot \left(1 - \frac{3}{2^{64}}\right) \cdot \ldots \cdot \left(1 - \frac{n-1}{2^{64}}\right) \end{aligned}$$

3. Έχουμε ότι :

$$\begin{aligned} p_{coll} &= 1 - \left(1 - \frac{1}{2^{64}}\right) \cdot \left(1 - \frac{2}{2^{64}}\right) \cdot \left(1 - \frac{3}{2^{64}}\right) \cdot \ldots \cdot \left(1 - \frac{n-1}{2^{64}}\right) \\ &\geq 1 - e^{-\frac{1}{2^{64}}} \cdot e^{-\frac{2}{2^{64}}} \cdot \ldots \cdot e^{-\frac{n-1}{2^{64}}} \\ &= 1 - e^{-\frac{\sum_{i=1}^{n-1} i}{2^{64}}} \\ &= 1 - e^{-\frac{n(n-1)}{2^{65}}} \end{aligned}$$

Κάνοντας υπολογισμούς για το κάτω φράγμα της p_{coll} διαπιστώνουμε ότι το n πρέπει να παίρνει μεγάλες τιμές για να έχει νόημα η επίθεση. Ενδεικτικά για $n = 10^9$ παίρνουμε κάτω φράγμα για p_{coll} την τιμή 0.0267, δηλαδή χρειαζόμαστε περίπου 40 μηνύματα κατά μέσο όρο, για να πετύχουμε σύγκρουση, ενώ για $n = 5 \cdot 10^9$ η πιθανότητα είναι σχεδόν $\frac{1}{2}$.

Άσκηση 5

1. Το μέγεθος της εξόδου πρέπει να είναι ακέραιο πολλαπλάσιο του μεγέθους του block. Επομένως, μπορούμε να βρούμε το μέγεθος block με τον εξής αλγόριθμο:
A. Δίνουμε το μήνυμα m μήκους l bits ως είσοδο στο oracle. Έστω n_1 το μέγεθος της εξόδου.
B. Δίνουμε ως είσοδο μήνυμα μεγέθους $l+1$ bits. Αν η έξοδος έχει μέγεθος $n_2 > n_1$, τότε μέγεθος block $= n_2 - n_1$. Αλλιώς θέσε $l \leftarrow l+1$ και επανάλαβε το βήμα B.
2. Έστω n το μέγεθος block. Επιλέγουμε ένα τυχαίο μήνυμα m με μήκος n bits.
Δίνουμε ως είσοδο στο oracle το $m||m$. Αν τα 2 πρώτα block του cipher διαφέρουν, τότε δεν χρησιμοποιείται ECB mode.
Αν είναι ίδια, τότε ή έχουμε ECB mode ή το IV ταυτίζεται με την έξοδο για το m, οπότε επαναλαμβάνουμε για κάποιο $m' \neq m$. Αν ταυτίζονται τα 2 πρώτα block της εξόδου ταυτίζονται, τότε χρησιμοποιείται ECB mode, αλλιώς δεν χρησιμοποιείται.
3. Υποθέτουμε ότι το block έχει μέγεθος n, το s έχει μέγεθος πολλαπλάσιο του n και χρησιμοποιείται ECB mode.
Αρχικά, περιγράφεται μια απλή μορφή του αλγορίθμου:
Ζητάμε την κρυπτογράφηση του $m = 0^n$ και βρίσκουμε την κρυπτογράφηση του, έστω c.
Αν κάποιο από τα blocks εκτός του πρώτου έχουν έξοδο c, τότε θεωρούμε ότι το αντίστοιχο μέρος του s είναι m.
Στη συνέχεια, αυξάνουμε το m κατά 1 και επαναλαμβάνουμε μέχρι να βρεθεί ολόκληρο το s.
Βελτιώσεις:
Μπορούμε να μειώσουμε το πλήθος των αποκρυπτογραφήσεων που ζητάμε, αν δίνουμε ως είσοδο μεγαλύτερα

μηνύματα (π.χ. $m = 0^n|0^{n-1}1|0^{n-2}10|0^{n-2}11$ αρχικά, και στη συνέχεια αυξάνουμε το καθένα κατά 4). Χωρίζουμε την έξοδο σε τμήματα μεγέθους n και ελέγχουμε αν κάποιο block εξόδου ταυτίζεται με κάποιο από αυτά.
Μπορούμε να αφαιρέσουμε τον περιορισμό το μέγεθος του s να είναι πολλαπλάσιο του n, αφού βρούμε το ακριβές μήκος του. Για να γίνει αυτό χρησιμοποιούμε τον αλγόριθμο του ερωτήματος 1. Το μέγεθος του block θα είναι $n_1 - l$.

Άσκηση 6

Έστω $P[2] = 0$ και $P[1] \neq 2$. Θεωρούμε ότι η μνήμη μετά τη δημιουργία σειράς κλειδιών και ακριβώς πριν την έναρξη του βρόχου έχει τις παρακάτω τιμές για μεταβλητές ή τιμές πινάκων που μας ενδιαφέρουν:

i	j	P[0]	P[1]	P[2]	...	P[b]
0	0	a	b	0	...	c

Εκτελούνται οι εντολές $i = (i+1) \mod 256$, $j = (j + P[i]) \mod 256$, οπότε η μνήμη γίνεται:

i	j	P[0]	P[1]	P[2]	...	P[b]
1	b	a	b	0	...	c

Επόμενη εντολή είναι η $swap(P[i], P[j])$ οπότε έχουμε:

i	j	P[0]	P[1]	P[2]	...	P[b]
1	b	a	c	0	...	b

Στη συνέχεια, υπολογίζεται το πρώτο byte, το οποίο δεν επηρεάζει την κατάσταση του προγράμματος. Έπειτα εκτελούνται ξανά οι εντολές $i = (i+1) \mod 256$, $j = (j + P[i]) \mod 256$, οπότε η μνήμη γίνεται:

i	j	P[0]	P[1]	P[2]	...	P[b]
2	b	a	c	0	...	b

Επόμενη εντολή είναι η $swap(P[i], P[j])$ οπότε έχουμε:

i	j	P[0]	P[1]	P[2]	...	P[b]
2	b	a	c	b	...	0

Επομένως, το δεύτερο byte εξόδου είναι:

$$\begin{aligned} P[(P[i] + P[j]) \mod 256] &= P[P[2] + P[b]] \\ &= P[b + 0] \\ &= 0 \end{aligned}$$

Η υπόθεση $P[1] \neq 2$ χρησιμοποιήθηκε στο ότι $P[B]$ και $P[2]$ δεν ταυτίζονται στη μνήμη και γι' αυτό δεν έγινε νωρίτερα κάποιο swap που να αφορά τη θέση 2. Επομένως, αν $P[2] = 0$ και $P[1] \neq 2$, τότε το δεύτερο byte εξόδου είναι 0.
Επομένως, αν θεωρήσουμε ότι $Pr\{P[2] = 0\} = 2^{-8}$ και για τις υπόλοιπες τιμές είναι ομοιόμορφη η πιθανότητα για το δεύτερο byte εξόδου, λόγω της ψευδοτυχαιότητας έχουμε για την πιθανότητα p το δεύτερο byte εξόδου να είναι 0:

$$\begin{aligned} p &= Pr\{P[2] = 0\} \cdot Pr\{K_2 = 0|P[2] = 0\} + Pr\{P[2] \neq 0\} \cdot Pr\{K_2 = 0|P[2] \neq 0\} \\ &= 2^{-8}\frac{2^8 - 1}{2^8} + \frac{2^8 - 1}{2^8} \cdot 2^{-8} \\ &= 2^{-7}\frac{2^8 - 1}{2^8} \\ &= 2^{-7} \end{aligned}$$

*Στους παραπάνω υπολογισμούς έγιναν προσεγγίσεις.

Άσκηση 7

1. Δεν είναι ψευδοτυχαία, γιατί για τις μισές συμβολοσειρές μήκους $n+1$, δηλαδή αυτές που τελειώνουν σε 1, προβλέπεται με ακρίβεια ότι δεν προέρχονται από τη συνάρτηση F_1, άρα υπάρχει μη αμελητέο πλεονέκτημα.
2. Είναι ψευδοτυχαία, γιατί κάθε bit επιλέγεται τυχαία, και στη συνέχεια με βάση το x επιλέγεται το αν θα αντιστραφεί ή όχι, οπότε παραμένει τυχαίο το αποτέλεσμα.
3. Είναι ψευδοτυχαία, καθώς ακολουθεί ακριβώς την ίδια κατανομή με την F με μοναδική αλλαγή ότι το συμπλήρωμα της εισόδου θα έδινε την αντίστοιχη τιμή.
4. Δεν είναι ψευδοτυχαία, καθώς δίνει ως έξοδο μόνο 2^n διαφορετικές συμβολοσειρές από τις 2^{2n} πιθανές συμβολοσειρές που δίνει η ομοιόμορφη κατανομή.

Άσκηση 8

(α) Ισχύει ότι $\pi(s_0, n) \mid \lambda(\lambda(n))$, όπου $\lambda()$ η συνάρτηση Carmichael. Αυτό ισχύει, γιατί :
Αν $ord_n s_0$ η τάξη του s_0 στη $\mathbb{Z}_n$, τότε για $x \in \mathbb{Z}_n(+1)$ είναι $ord_n s_0$ περιττός, αφού :
-$ord_n s_i = ord_n s_{i+1}$, γιατί $ord_n s_i | ord_n s_{i+1}$, αφού είναι $s_{i+1} = s_i^2$ και $s_0, s_1, ...$ κυκλική
-$\forall u > 0 (u \in \mathbb{Z}$, αν $2^u | ord_n s_i \wedge 2^{u+1} \not| ord_n s_i$, τότε $2^{u+1} | ord_n s_{i+1} \wedge 2^{u+1} \not| ord_n s_{i+1}$ Επομένως, από επεκτεταμένο θεώρημα του Euler ισχύει ότι $2^{\lambda(ord_n s_0} \equiv 1 \mod ord_n s_0$. Όμως, $\pi(s_0, n)$ ο μικρότερος θετικός ακέραιος τέτοιος, ώστε $2^{\pi(s_0,n)} \equiv 1 \mod (ord_n s_0)$, αφού $\pi(s_0, n)$ ο μικρότερος θετικός ακέραιος για τον οποίο ισχύει $s_0 \equiv s_0^{2^{\pi}(s_0,n)}$ mod n, άρα προκύπτει ότι $\pi(s_0, n) | \lambda(ord_n s_0)$. Όμως, ισχύει ότι $\lambda(ord_n s_0) | \lambda(\lambda(n))$, αφού $ord_n s_0 | \lambda(n) \forall s_0 \in \mathbb{Z}_n$ από θεώρημα Lagrange.
Επομένως, $\pi(s_0, n) | \lambda(\lambda(n))$.
Για n Blum integer ισχύει ότι $\lambda(n) = lcm(p-1, q-1) = \frac{(p-1)(q-1)}{gcd(p-1,q-1)}$, για αυτό και θέλουμε το $gcd(p-1, q-1)$ να είναι μικρό, ώστε να έχει μεγάλη τιμή το $\lambda(n)$ για να μπορεί να πάρει μεγάλες τιμές η περίοδος.
Χρησιμοποιήθηκε η πηγή A Simple Unpredictable Pseudo-Random Number Generator (Blum, Blum, Shub)
(β) Η μέγιστη περίοδος για p, q SafeSafe primes είναι :

$$\begin{aligned} max(\pi(s_0, n)) &= \lambda(\lambda(n)) \\ &= \lambda(lcm(p-1, q-1)) \\ &= \lambda(lcm(2p', 2q')) \\ &= \lambda(2p'q') \\ &= lcm(p', q') \\ &= lcm(2p'', 2q'') \\ &= 2p''q'' \end{aligned}$$

Άσκηση 9

Από το paper των Blum, Blum, Shub προκύπτει ότι η περίοδος ισούται με το $\frac{\lambda(\lambda(n))}{d}$ για το μέγιστο d που διαιρεί το $\lambda(\lambda(n))$ και ισχύει $s_0^{2^{\frac{\lambda(\lambda(n))}{d} mod \lambda(n)}} \mod n = s_0$, όπου για p, q SafeSafe primes είναι $\lambda(n) = 2p'q'$ και $\lambda(\lambda(n)) = 2p''q''$. Επομένως, αρκεί να εξετάσουμε μόνο τους μη τετριμμένους διαίρετες του $\lambda(\lambda(n))$ και αν δεν ισχύει η παραπάνω σχέση για κανέναν από αυτούς θα είναι $d = 1$, οπότε θα έχουμε τη μέγιστη περίοδο.
Για την επιλογή του s_0 μπορούμε να εξετάσουμε με τη σειρά τις τιμές του s_0, στην υλοποίηση επιλέγεται τυχαία, για να φανεί ότι τα αποτελέσματα ισχύουν γενικά και όχι μόνο για μικρές τιμές του s_0. Στη συνέχεια, ξεκινάμε από το s_0^2 και υψώνουμε συνεχώς στο τετράγωνο μέχρι να φτάσουμε ξανά στο αρχικό, για να υπολογίσουμε πειραματικά την περίοδο.
Τα αποτελέσματα για κάποιες τυχαίες εκτελέσεις, όπως φαίνεται παρακάτω δείχνουν πως η θεωρητική περίδος συμπίπτει με την πειραματική:

Ο κώδικας σε python που χρησιμοποιήθηκε:

```python
import random
from math import gcd

def fastmodpower (a , n , p ) :
    result = 1
    while n > 0:
        if n % 2 == 1:
            result = result * a % p
        n = n // 2
        a = a * a % p
    return result

def isPrime(n) :
    if n <=1:
        return False
    if n <=3:
        return True
    if n %2==0:
        return False
    q = n -1
    while q % 2 == 0:
        q //=2
    for i in range (30):
        if not RabinMillerTest(n, q):
            return False
    return True

def RabinMillerTest(n , q) :
    a = random.randint(2 , n - 2)
    x = fastmodpower (a ,q , n )
    if x ==1 or x == n -1:
        return True
    while q != n -1:
        x = ( x * x ) % n
        q = 2 * q
        if x == 1:
            return False
        if x == n -1:
            return True
    return False


max_n = 2**20

while True:
    p = random.randint(0, max_n)
    ptonos = (p-1) // 2
    pdistono = (ptonos - 1) // 2
    if pdistono % 4 == 1 and isPrime(pdistono) and isPrime(ptonos) and isPrime(p):
        break

while True:
    q = random.randint(0, max_n)
    qtonos = (q-1) // 2
    qdistono = (qtonos - 1) // 2
    if qdistono % 4 == 1 and isPrime(qdistono) and isPrime(qtonos) and isPrime(q):
        break

print("p =", p, ", q = ", q)

```

```
    period = 2 * pdistono * qdistono
    n = p*q


    #find s0
    isGood = False
    while not isGood:
        isGood = True
        s = random.randint(2, n)

        s = s * s % n
        if gcd(s, n) > 1:
            continue
        factor = [2, pdistono, qdistono, 2 * pdistono, 2 * qdistono,  pdistono * qdistono]
        for d in factor:
            exp = fastmodpower(2, 2 * pdistono * qdistono / d, 2 * ptonos * qtonos)
            if fastmodpower(s, exp, n) == s:
                    isGood = False
                    break

    print("Theoretical period =", period)
    cnt = 1
    out = s
    out = out * out % n

    while out != s:
        out = out * out % n
        cnt += 1

    print("Experimental period =", cnt)
```

Άσκηση 10

1. Έστω $k \in \mathbb{Z}$ και m ένα οποιοδήποτε μήνυμα. Τότε έχουμε:

$$\begin{aligned} H(m \oplus 0^k) = H(m) \oplus H(0^k) &\Leftrightarrow \\ H(m) = H(m) \oplus H(0^k) &\Leftrightarrow \\ H(0^k) = H(m) \oplus H(m) &\Leftrightarrow \\ H(0^k) = 0^n & \end{aligned}$$

Επειδή το k που επιλέχθηκε είναι τυχαίο, ισχύει για κάθε k, οπότε για να βρούμε σύγκρουση μπορούμε να επιλέξουμε οποιεσδήποτε συμβολοσειρές διαφορετικού μήκους που αποτελούνται μόνο από 0.
2. Έστω ότι η H δεν έχει δυσκολία εύρεσης συγκρούσεων, τότε μπορούν να βρεθούν x_1, x_2 τέτοια, ώστε $x_1 \neq x_2$ και $H(x_1) = H(x_2)$. Έχουμε:

$$\begin{gathered} H(x_1) = H(x_2) \Leftrightarrow \\ H_1(x_1)||H_2(x_1)||H_3(x_1) = H_1(x_2)||H_2(x_2)||H_3(x_2) \Leftrightarrow \\ H_1(x_1) = H_1(x_2) \wedge H_2(x_1) = H_2(x_2) \wedge H_3(x_1) = H_3(x_2) \end{gathered}$$

Επομένως, αν μπορούμε να βρούμε σύγκρουση στην H βρίσκουμε σύγκρουση και για τις 3 συναρτήσεις, άρα καμία από αυτές δεν έχει δυσκολία εύρεσης συγκρούσεων, το οποίο είναι άτοπο, καθώς μία από αυτές έχει.
Άρα, η H έχει δυσκολία εύρεσης συγκρούσεων.

Άσκηση 11

Ορίζουμε τη συνάρτηση f ως εξής:

$$f(x_1, x_2) = H_1(x_1, x_2)$$

$$f(x_1, x_2, ..., x_{2^h}) = H_1(f(x_1, ..., x_{2^{h-1}}), f(x_{2^{h-1}+1}, ..., x_{2^h}))$$

Από τον ορισμό της f ισχύει ότι $H(x_1, ..., x_{2^h}) = f(x_1, ..., x_{2^h})$.
Έστω ότι η H έχει ευκολία εύρεσης συγκρούσεων, δηλαδή υπάρχουν ακολουθίες $(x_1, ..., x_{2^h})$ και $(x_1^{'}, ..., x_{2^h}^{'})$

διαφορετικές που δίνουν την ίδια τιμή μέσω της H. Επομένως, έχουμε:

$$H(x_1, ..., x_{2^h}) = H(x_1^{'}, ..., x_{2^h}^{'}) \Rightarrow$$
$$f(x_1, ..., x_{2^h}) = f(x_1^{'}, ..., x_{2^h}^{'}) \Rightarrow$$
$$H_1(f(x_1, ..., x_{2^{h-1}}), f(x_{2^{h-1}+1}, ..., x_{2^h})) = H_1(f(x_1^{'}, ..., x_{2^{h-1}}^{'}), f(x_{2^{h-1}+1}^{'}, ..., x_{2^h}^{'}))$$

Επομένως, οι είσοδοι $f(x_1, ..., x_{2^{h-1}}), f(x_{2^{h-1}+1}, ..., x_{2^h})$ και $f(x_1^{'}, ..., x_{2^{h-1}}^{'}), f(x_{2^{h-1}+1}^{'}, ..., x_{2^h}^{'})$ δίνουν σύγκρουση για την H_1, οπότε ούτε η H_1 είναι δύσκολη συγκρούσεων, το οποίο είναι άτοπο. Άρα και η H θα έχει δυσκολία συγκρούσεων.

Άσκηση 12

1. Ο A στέλνει μήνυμα m και λαμβάνει κρυπτοκείμενο c. Από αυτό ανακτά ένα κλειδί k με πιθανότητα p.
Στη συνέχεια, στέλνει μηνύματα m_0, m_1.
Ο C επιλέγει $b \in \{0, 1\}$ και στέλνει την κρυπτογράφηση $c = m_b$.
Ο A μαντεύει $b' = 0$, αν $c = Enc_k(m_0)$, αλλιώς $b' = 1$.
Η πιθανότητα να κερδίσει ο A είναι $p \cdot 1 + (1 - p) \cdot \frac{1}{2} = \frac{1}{2} + \frac{p}{2}$, αφού αν βρει το κλειδί θα απαντήσει σωστά με πιθανότητα 1, ενώ απαντά στην τύχη σε διαφορετική πιθανότητα.
Επειδή η πιθανότητα p είναι μη αμελητέα το πλεονέκτημα του A είναι $\frac{1}{2} + \frac{p}{2} - \frac{1}{2} = \frac{p}{2}$ είναι επίσης μη αμελητέο, άρα δεν παρέχεται ασφάλεια CPA.
2. Ο A στέλνει μήνυμα m και λαμβάνει κρυπτοκείμενο c.
Στη συνέχεια, στέλνει μηνύματα m_0, m_1, όπου το m_0 διαφέρει από το m μόνο στο τελευταίο bit του πρώτου block.
Ο C επιλέγει $b \in \{0, 1\}$ και στέλνει την κρυπτογράφηση $c' = m_b$.
Ο A μαντεύει $b' = 0$, αν $c = c'$, αλλιώς $b' = 1$.
Στην περίπτωση που το τελευταίο bit του IV ήταν 0, ο A βρίσκει πάντα τη σωστή λύση, αφού το xor του IV με το πρώτο block θα είναι το ίδιο τόσο στο αρχικό μήνυμα όσο και στην κρυπτογράφηση του challenger, οπότε η έξοδος θα είναι ίδια για όλα τα block.
Αντίθετα, αν το το τελευταίο bit του IV ήταν 1, τότε το αν θα βρει το σωστό bit ο A είναι τυχαίο, άρα η συνολική πιθανότητα να κερδίσει ο A είναι $\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot \frac{1}{2} = \frac{3}{4}$, άρα έχει μη αμελητέο πλεονέκτημα σε σχέση με το να μάντευαι στην τύχη.
3. Έστω ότι το OFB είναι IND-CCA2, τότε το OFB είναι non-malleable. Όμως, αν γνωρίζουμε κρυπτογράφηση (m, IV, c), τότε η (m', IV, c') είναι έγκυρη κρυπτογράφηση για το ίδιο κλειδί, άρα το OFB είναι malleable, όπου το σύμβολο $'$ δηλώνει το συμπλήρωμα. Επομένως, το OFB δεν είναι IND-CCA2.