ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

2η Σειρά Ασκήσεων

Υπολογιστική Κρυπτογραφία

Ανδρέας Στάμος

Αριθμός μητρώου: 03120***

Διεύθυνση ηλεκτρονικού ταχυδρομείου: stamos.aa@gmail.com

9ο Εξάμηνο — 2024-2025

Περιεχόμενα

1 Άσκηση 1: Κύρια τετραγωνική ρίζα αριθμού Blum

Αρχικά δείχνουμε ότι ο αριθμός x είναι μια τετραγωνική ρίζα του y, έπειτα δείχνουμε ότι ο x είναι τετραγωνικό υπόλοιπο ως προς n και τέλος δείχνουμε πως ο αριθμός που είναι τετραγωνική ρίζα του y και επίσης τετραγωνικό υπόλοιπο ως προς n είναι μοναδικός.

Λήμμα 1.1. *Έστω $p, q \in \mathbb{P}$ με $p \equiv q \equiv 3 \pmod 4$ και $n = pq$ (Blum integer). Έστω ένα τετραγωνικό υπόλοιπο $y \in QR(n)$ ως προς n. Έστω ο αριθμός:*

$$x = y^{\frac{(p-1)(q-1)+4}{8}} \pmod n \tag{1.1}$$

Ο x είναι τετραγωνική ρίζα του y.

Απόδειξη. Αρχικά, αφού $p \equiv q \equiv 3 \pmod 4$ τότε $(p-1)(q-1)+4 \equiv 0 \pmod 4$ οπότε ο x μπορεί να οριστεί.

Αφού $y \in QR(n)$ τότε υπάρχει $z \in U(\mathbb{Z}_n)$ ώστε:

$$y \equiv z^2 \pmod n \tag{1.2}$$

Αφού $p, q \in \mathbb{P}$ τότε p, q σχετικά πρώτοι, οπότε:

$$\begin{aligned} y &\equiv z^2 \pmod p \\ y &\equiv z^2 \pmod q \end{aligned} \tag{1.3}$$

(αυτό ισχύει διότι $pq \mid (y - z^2)$ οπότε αφού οι p, q δεν έχουν κοινούς παράγοντες τότε πρέπει και οι δύο να διαιρούν τον $y - z^2$).

Άρα $y \in QR(p)$ και $y \in QR(q)$. Τότε από Κριτήριο Euler είναι:

$$\begin{aligned} y^{\frac{p-1}{2}} \equiv 1 \pmod p \Rightarrow y^{\frac{(p-1)(q-1)}{4}} \equiv 1^{\frac{q-1}{2}} \equiv 1 \pmod p \\ y^{\frac{q-1}{2}} \equiv 1 \pmod q \Rightarrow y^{\frac{(p-1)(q-1)}{4}} \equiv 1^{\frac{p-1}{2}} \equiv 1 \pmod q \end{aligned} \tag{1.4}$$

Αφού $p, q \in \mathbb{P}$, τότε p, q σχετικά πρώτοι, οπότε το παραπάνω σύστημα (θεωρώντας το $y^{\frac{(p-1)(q-1)}{4}}$ ως άγνωστο) έχει μοναδική λύση στο Z_{pq}. Το 1 είναι μια λύση αυτού του συστήματος. Συνεπώς:

$$y^{\frac{(p-1)(q-1)}{4}} \equiv 1 \pmod{pq} \tag{1.5}$$

Τότε:

$$x^2 \equiv y^{\frac{(p-1)(q-1)+4}{4}} \equiv y \cdot y^{\frac{(p-1)(q-1)}{4}} \overset{1.5}{\equiv} y \cdot 1 \pmod{pq} \tag{1.6}$$

Άρα ο αριθμός x είναι τετραγωνική ρίζα του y ως προς $n = pq$. □

Προχωράμε να αποδείξουμε ότι $x \in QR(n)$.

Λήμμα 1.2. *Έστω $p, q \in \mathbb{P}$ με $p \equiv q \equiv 3 \pmod 4$ και $n = pq$ (Blum integer). Έστω ένα τετραγωνικό υπόλοιπο $y \in QR(n)$ ως προς n. Έστω ο αριθμός:*

$$x = y^{\frac{(p-1)(q-1)+4}{8}} \pmod n \tag{1.7}$$

Ο x είναι τετραγωνικό υπόλοιπο ως προς n ($x \in QR(n)$).

Απόδειξη. Αρχικά, αφού $p \equiv q \equiv 3 \pmod 4$ τότε $(p-1)(q-1)+4 \equiv 0 \pmod 4$ οπότε ο x μπορεί να οριστεί.

Ο αριθμός y είναι τετραγωνικό υπόλοιπο ως προς n.

Αφού $y \in QR(n)$ τότε υπάρχει $z \in U(\mathbb{Z}_n)$ ώστε:

$$y \equiv z^2 \pmod n \tag{1.8}$$

Τότε:

$$x \equiv y^{\frac{(p-1)(q-1)+4}{8}} \equiv x \equiv (z^2)^{\frac{(p-1)(q-1)+4}{8}} \equiv x \equiv \left(z^{\frac{(p-1)(q-1)+4}{8}} \right)^2 \pmod{n} \tag{1.9}$$

Άρα ο x είναι τετραγωνικό υπόλοιπο ως προς n ($x \in QR(n)$). □

Τέλος αποδεικνούμε ότι ο x είναι μοναδικός.

Λήμμα 1.3. *Έστω $p, q \in \mathbb{P}$ και $n = pq$. Έστω ένα τετραγωνικό υπόλοιπο $y \in QR(n)$ ως προς n. Έστω μια κύρια τετραγωνική ρίζα x, του y, η οποία δηλαδή είναι και αυτή τετραγωνικό υπόλοιπο ως προς n ($x \in QR(n)$).*

Τότε ο αριθμός x είναι μοναδικός. Ισοδύναμα αν υπάρχουν δύο τέτοιοι αριθμοί x_1, x_2 τότε $x_1 \equiv x_2 \pmod{n}$.

Απόδειξη. Έστω δύο τέτοιοι αριθμοί x_1, x_2, δηλαδή ${x_1}^2 \equiv {x_2}^2 \equiv y \pmod{n}$, $x_1, x_2 \in QR(n)$. Έστω $x_1 \neq x_2 \pmod{n}$.

Θεωρούμε $0 \leq x_1, x_2 < n$ (αν δεν ισχύει, θεωρούμε x_1, x_2 τους υπόλοιπους τους ως προς n)

Επίσης θεωρούμε $x_1 > x_2$. (αν δεν ισχύει, τους αντιμεταθέτουμε)

Είναι:

$${x_1}^2 \equiv {x_2}^2 \pmod{n} \Rightarrow (x_1 - x_2)(x_1 + x_2) \equiv 0 \pmod{pq} \Rightarrow pq \mid (x_1 - x_2)(x_1 + x_2) \tag{1.10}$$

Επειδή $0 \leq x_2 < x_1 < n$ είναι $0 \leq x_1 - x_2 < n$ οπότε αποκλείεται $n \mid (x_1 - x_2)$.

Επειδή $0 \leq x_2 < x_1 < n$ είναι $0 \leq x_1 + x_2 < 2 \cdot n$.

Έστω $n \mid (x_1 + x_2)$. Τότε είναι ακριβώς $x_1 + x_2 = n$, οπότε $x_2 \equiv -x_1 \pmod{n}$ και άρα αφού $p, q \in \mathbb{P}$ είναι $x_1 \equiv -x_1 \pmod{p}$.

Όπως έχουμε δειξεί σε απόδειξη προηγούμενου θεωρήματος αφού $x_1, x_2 \in QR(n)$ τότε $x_1, x_2 \in QR(p)$. Συνεπώς από Κριτήριο Euler:

$$x_1^{\frac{p-1}{2}} \equiv x_2^{\frac{p-1}{2}} \pmod{p} \tag{1.11}$$

Όμως:

$$x_2^{\frac{p-1}{2}} \equiv (-x_1)^{\frac{p-1}{2}} \equiv (-1)^{\frac{p-1}{2}} \cdot x_1^{\frac{p-1}{2}} \pmod{p} \tag{1.12}$$

Είναι $p - 1 \equiv 3 - 1 \equiv 2 \pmod{4}$, οπότε $\frac{p-1}{2} \equiv 1 \pmod{2}$ οπότε $(-1)^{\frac{p-1}{2}} = -1$.

Συνεπώς:

$$\begin{gathered} x_1^{\frac{p-1}{2}} \equiv x_2^{\frac{p-1}{2}} \equiv -x_1^{\frac{p-1}{2}} \pmod{p} \Rightarrow \\ 2x_1^{\frac{p-1}{2}} \equiv 0 \pmod{p} \overset{p \in \mathbb{P}}{\Rightarrow} x_1^{\frac{p-1}{2}} \equiv 0 \pmod{p} \Rightarrow {x_1}^{p-1} \equiv 0 \pmod{p} \end{gathered} \tag{1.13}$$

Το τελευταίο είναι άτοπο από Θεωρήμα Euler. Συνεπώς είναι και $n \nmid (x_1 + x_2)$.

Άρα $q \mid (x_1 - x_2)$ και $p \mid (x_1 + x_2)$ ή αντίστροφα. Εξετάζουμε αυτή την περίπτωση και όμοια ισχύει η άλλη.

Είναι $x_2 \equiv -x_1 \pmod{p}$, οπότε προκύπτει το ίδιο άτοπο ακριβώς με πριν.

Άρα $x_1 \equiv x_2 \pmod{n}$. □

Τελικά προκύπτει το ζητούμενο.

Θεώρημα 1.1. *Έστω $p, q \in \mathbb{P}$ με $p \equiv q \equiv 3 \pmod{4}$ και $n = pq$ (Blum integer). Έστω ένα τετραγωνικό υπόλοιπο $y \in QR(n)$ ως προς n. Έστω ο αριθμός:*

$$x = y^{d^{\frac{(p-1)(q-1)+4}{8}}} \pmod{n} \tag{1.14}$$

Ο x είναι ο μοναδικός αριθμός που είναι και τετραγωνική ρίζα του y και ο ίδιος τετραγωνικό υπόλοιπος ως προς $n = pq$.

Απόδειξη. Ισχύουν τα λήμματα 1.1, 1.2 και 1.3. □

2 Άσκηση 2: Ασφάλεια πειραγμένου DES-X

Θεωρούμε μια παραλλαγή του DES-X, με 2 κλειδιά $k_1, k_2 \in \{0,1\}^{64}$, όπου η κρυπτογράφηση ενός μηνύματος M ορίζεται ως:

$$Enc_{k_1,k_2}(M) = E_{k_1}(M \oplus k_2) \tag{2.1}$$

όπου E η συνάρτηση κρυπτογράφησης του DES.

Υποθέτοντας ότι το DES προσφέρει ασφάλεια ψευδοτυχαίας μετάθεσης, το σύστημα αυτό δεν προσφέρει περισσότερη ασφάλεια. Η υπόθεση καλής ασφάλειας του DES απαιτείται προκειμένου το DES να "σπάει" μόνο με brute-force επίθεση που χρειάζεται μέση τιμή 2^{56-1} δοκιμές κλειδιών.

Χρησιμοποιούμε δύο ζεύγη μηνύματος-κρυπτοκειμένου $(m_1, c_1), (m_2, c_2)$. Είναι:

$$D_{k_1}(c_1) \oplus D_{k_1}(c_2) = (m_1 \oplus k_2) \oplus (m_2 \oplus k_1) = m_1 \oplus m_2 \tag{2.2}$$

Συνεπώς προκύπτει ο εξής αλγόριθμος επίθεσης για το σύστημα κρυπτογράφησης:

Για όλα τα $k_1 \in \{0,1\}^{56}$ υπολόγισε το $D_{k_1}(c_1) \oplus D_{k_1}(c_2)$. Αν βγει ίσο με το $m_1 \oplus m_2$ το k_1 είναι *πιθανό κλειδί*. Για όλα τα *πιθανά κλειδιά* k_1 υπολόγισε $k_2 = m_1 \oplus D_{k_1}(c_1)$, και έλεγξε έλεγξε αν το (m_2, c_2) είναι έγκυρο ζεύγος. Αν διατίθεται περισσότερα ζεύγη έλεγξε και περισσότερα. Αν δεν προκύψει κάποιο άκυρο ζεύγος, ανακοίνωσε επιτυχία το κλειδί (k_1, k_2).

Είναι απίθανο να υπάρξουν κλειδιά που οδηγούν σε άκυρο ζεύγος, καθώς δίνουν μια σύγκρουση για το $(m_1 \oplus k_1, c_1)$ στην ψευδοτυχαία συνάρτηση του DES, και αυτές οι συγκρούσεις είναι απίθανες από το παράδοξο γενεθλίων. Πιο συγκεκριμένα, έχουμε διαλέξει $n = 2^{56}$ δείγματα (μια φορά για κάθε κλειδί) από σύνολο μεγέθους $N = 2^{64} \cdot 2^{64} = 2^{128}$. Η πιθανότητα σύγκρουσης είναι, από γνωστό θεώρημα, περίπου ίση με:

$$\frac{n^2}{N} \approx 2^{56-128} = 2 \cdot 10^{-22} \tag{2.3}$$

Συνεπώς περιμένουμε ότι αν βρέθει πιθανό κλειδί, αυτό είανι με πάρα πολύ υψηλή πιθανότητα το αληθινό κλειδί.

Συνεπώς ο αλγόριθμος εξετάζει κατά μέσο όρο 2^{56-1} κλειδιά όπως ακριβώς και ο αλγόριθμος για την κρυπτανάλυση του DES. Βέβαια εδώ απαιτούνται δύο αποτιμήσεις αποκρυπτογράφησης για κάθε κλειδί, οπότε στο τέλος θα χρειαστούμε διπλό χρόνο, αλλό αυτό δεν θεωρείται ότι προσδίδει πρόσθετη ασφάλεια.

Συνεπώς το σύστημα προσφέρει ίδια ασφάλεια με το κλασικό DES.

3 Άσκηση 3: Ασθενή και ημιασθενή κλειδιά DES

3.1 Ασθενή κλειδιά

Ορισμός 3.1. *Για ένα πεπερασμένο σύνολο S, μια 1-1 και επί συνάρτηση $f : S \mapsto S$, δηλαδή μια μετάθεση, είναι ενέλιξη αν $f(f(x)) = x, \forall x in S$.*

Ορισμός 3.2. *Ασθενές κλειδί είναι ένα κλειδί για το οποίο η συνάρτηση Enc_k είναι ενέλιξη (involution).*

Σκοπός είναι να βρεθούν 4 ασθενή κλειδιά του DES.

Σε γενικές γραμμές σκοπός είναι να κάνουμε exploit το Πρόγραμμα Παραγωγής Κλειδιών (Key Schedule) ώστε να παράγει για κάθε γύρο το ίδιο κλειδί και να δείξουμε ότι τότε η συνάρτηση κρυπτογράφησης του DES γίνεται ενέλιξη.

Ξεκινάμε εξετάζοντας τι συμβαίνει αν σε όλους τους γύρους προκύψει ίδιο κλειδί.

Θεώρημα 3.1. *Έστω ότι το Πρόγραμμα Παραγωγής Κλειδιών παράγει σε όλους τους γύρους το ίδιο κλειδί γύρου $k_i = k$.*

Τότε η συνάρτηση κρυπτογράφησης του DES είναι ενέλιξη.

Απόδειξη. Κατά τα γνώστα η συνάρτηση αποκρυπτογράφησης του DES είναι ακριβώς ίδια με την συνάρτηση κρυπτογράφησης απλά με αντίστροφη παραγωγή κλειδιών (το κλειδί γύρου 1, γίνεται το κλειδί τελευταίου γύρου, κ.λπ.). Η ιδιότητα αυτή στηρίζεται στο γεγονός ότι:

- Η αρχική μετάθεση (Initial Permutation) που εφαρμόζεται στην μη κρυπτογραφημένη είσοδο είναι αντίστροφη της τελικής μετάθεσης (Final Permutation) που εφαρμόζεται στην κρυπτογραφημένη έξοδο.
- Ο i-οστός γύρος κρυπτογράφησης δίνει $(L_{i+1}, R_{i+1}) = (R_i, L_i \oplus F(R_i, k_i))$. Στον τελευταίο γύρο επιστέφουμε το (R_n, L_n) (προσέχουμε ότι είναι αντίστροφα επίτηδες, ώστε να μπορεί να χρησιμοποιείται η ίδια συνάρτηση για κρυπτογράφηση και αποκρπτογράφηση). Χρησιμοποιώντας την ίδια διαδικασία αλλά με τα κλειδιά ανάποδα λαμβάνουμε ότι (εδώ το "τρέχουμε" στο (R_n, L_n)):

$$Round(R_{i+1}, L_{i+1}) = (L_{i+1}, R_{i+1} \oplus F(L_{i+1}, k_i)) = (R_i, L_i \oplus F(R_i, k_i) \oplus F(R_i, k_i)) = (R_i, L_i) \quad (3.1)$$

Άρα ούτως ή άλλως στο DES η συνάρτηση κρυπτογράφησης γύρου είναι ίδια με την συνάρτηση αποκρυπτογράφησης, απλά στην κρυπτογράφηση τα κλειδιά πρέπει να δοθούν με αντίστροφη σειρά.

Συνεπώς αν η ακουλουθία των κλειδιών είναι ίδια με την αντίστροφη της, που ισχύει αν είναι η ακολουθία επαναλαμβάνει συνεχώς το ίδιο στοιχείο όπως εδώ, η πράξη κρυπτογράφησης είναι ίδια με την πράξη αποκρυπτογράφησης. □

Απομένει να φτιάξουμε κάποιο αρχικό κλειδί ώστε το Πρόγραμμα Παραγωγής Κλειδιών να δώσει σε όλους τους γύρους ίδιο κλειδί.

Μελέτουμε το κλειδί αφού έχει εφαρμοστεί η αρχική μετάθεση `PC1`.

Το Πρόγραμμα Παραγωγής Κλειδιών σπάει το κλειδί 56-bit στην μέση σε 2 28-bit αριθμούς. Σε κάθε γύρο εφαρμόζει αριστερό κυκλικό shift σε κάθε αριθμό και έπειτα επιλέγει 24-bit από αυτόν με βάση μια μετάθεση `PC2` (δεν είναι τυπικά μετάθεση, διότι απορρίπτει κάποια bit, ωστόσο έτσι αποκαλείται.). Το κλειδί γύρου είναι η παράθεση αυτών των δύο 24-bit αριθμών.

Για να προκύψουν όλα τα κλειδιά γύρου ίδια αρκεί να βγαίνουν ίδιοι σε όλους τους γύρους χωριστά οι δύο αριθμοί 24-bit. Για να συμβεί αυτό αρκεί σε κάθε γύρο πριν την επιλογή `PC2` να προκύπτουν ίδιοι σε όλους τους γύρους χωριστά οι δύο αριθμοί 28-bit. Οι αριθμοί 28-bit είναι κυκλικό αριστερό shift του προηγούμενου γύρου. Συνεπώς την ιδιότητα αυτή ικανοποιούν οι 0^{28} και 1^{28}. Άρα αρκεί στο αρχικό κλειδί, μετά την εφαρμογή του `PC1` να έχει τα MSB 28-bit και τα LSB 28-bit κάποιο από αυτά τα δύο (συνολικά 4 συνδυασμοί).

Με άλλα λόγια, βρήκαμε 4 ασθενή κλειδιά. Αυτά που μετά την εφαρμογή του `PC1` αποκτούν μια από τις παρακάτω τιμές:

1. $0^{28}0^{28}$
2. $0^{28}1^{28}$
3. $1^{28}0^{28}$
4. $1^{28}1^{28}$

Η μετάθεση `PC1`, αγνοώντας τα 8 bits που απορρίπτονται, είναι αντιστρεπτή, οπότε από αυτά μπορούν να βρεθούν τα αρχικά ασθενή κλειδιά.

3.2 Ημιασθενή κλειδιά

Ορισμός 3.3. *Ημιασθενές κλειδί είναι ένα κλειδί το οποίο δεν είναι ασθενές και για το οποίο υπάρχει κλειδί k' τέτοιο ώστε:*

$$Dec_k = Enc_{k'} \tag{3.2}$$

Σκοπός είναι να βρεθούν 4 ημιασθενή κλειδιά.

Η συνάρτηση αποκρυπτογράφησης, όπως αναφέρθηκε παραπάνω, είναι ίδια με την συνάρτηση κρυπτογράφησης, απλά δίνοντας τα κλειδιά γύρων με την αντίστροφη σειρά. Συνεπώς θέλουμε η αντίστροφη σειρά να μπορεί να προκύψει ως παραγωγή κλειδιών από κάποιο άλλο αρχικό κλειδί.

Για την αντίστροφη σειρά, ισοδύναμα, σε κάθε 28-bit αριθμό αντί να κάνουμε αριστερό shift, είναι σαν να κάνουμε δεξιό shift. Συνεπώς αρκεί ο 28-bit αριθμός με αριστερό και δεξιό κυκλικό shift να δίνει τον ίδιο αριθμό. Αγνοώντας τους αριθμούς που επιλέχθηκαν στα ασθενή κλειδιά, την ιδιότητα ικανοποιούν οι αριθμοί που εναλλάσσουν τα 0,1, δηλαδή ο 01010101 ... και 101010 ....

Καθένας από τους δύο αριθμούς μπορεί να είναι είτε στα MSB είτε στα LSB 28-bit.

Παρατηρούμε ότι τα κλειδιά αυτά δεν είναι ασθενή, διότι ο τελευταίος γύρος έχει διαφορετικό κλειδί από τον πρώτο γύρο, καθώς απαιτείται άρτιο πλήθος shifts για να προκύπτει ίδιος αριθμός (δηλαδή ίδια κλειδιά έχουν οι γύροι $1, 3, 5, \cdots$ και $2, 4, 6, \cdots$ όχι όμως ο 1 με τον 16)

Με άλλα λόγια, βρήκαμε 4 ημιασθενή κλειδιά. Αυτά που μετά την εφαρμογή του `PC1` αποκτούν μια από τις παρακάτω τιμές:

1. 010101 ... 010101 ...
2. 010101 ... 101010 ...
3. 101010 ... 010101 ...
4. 101010 ... 101010 ...

Η μετάθεση `PC1`, αγνοώντας τα 8 bits που απορρίπτονται, είναι αντιστρεπτή, οπότε από αυτά μπορούν να βρεθούν τα αρχικά ημιασθενή κλειδιά.

4 Άσκηση 4: Cipher Block Chaining (CBC) Mode of Operation

4.1 Εξαγωγή πληροφορίας από σύγκρουση κρυπτοκειμένων

Υποθέτουμε ότι προέκυψαν δύο blocks κρυπτοκειμένου, έστω το y_i και το y_j $(i \neq j)$, ίσα.

Ισχύει ότι:

$$y_i = y_j \iff Enc(y_{i-1} \oplus x_i) = Enc(y_{j-1} \oplus x_j) \overset{Enc:\text{μετάθεση}}{\iff} y_{i-1} \oplus x_i = y_{j-1} \oplus x_j \iff x_i \oplus x_j = y_{i-1} \oplus y_{j-1} \quad (4.1)$$

Συνεπώς μπορούμε να υπολογίσουμε το $x_i \oplus x_j$ καθώς τα y_i είναι γνωστά.

Επίσης:

$$\begin{aligned} y_{i+1} &= Enc_k(x_{i+1} \oplus y_i) \\ y_{j+1} &= Enc_k(x_{j+1} \oplus y_j) = Enc_k(x_{j+1} \oplus y_i) \end{aligned} \quad (4.2)$$

Συνεπώς: $y_{i+1} = y_{j+1} \iff x_{i+1} = x_{j+1}$, οπότε μπορούμε να ξέρουμε αν τα επόμενα μηνύματα των i, j είναι ίσα ή όχι (φαίνεται ασήμαντο αλλά μπορεί είτε να μην είναι είτε να μπορεί να γίνει exploit σε μια επίθεση.).

4.2 Πιθανότητα σύγκρουσης

Υποθέτουμε ότι το κρυπτοσύστημα συμπεριφέρεται ως ψευδοτυχαία μετάθεση (*pseudorandom permutation*). Η ψευδοτυχαία μετάθεση διακρίνεται από PPT αντίπαλο με αμελητέα πιθανότητα από την ψευδοτυχαία συνάρτηση (*pseudorandom function*). Εργαζόμαστε στο ερώτημα για την πιθανότητα που θεωρεί ότι αντιμετωπίζει ένας PPT αντίπαλος, θεωρώντας ότι το κρυπτόσυστημα είναι τυχαία συνάρτηση, αφού αυτός ο αντίπαλος ούτως ή άλλως με αμελητέα πιθανότητα μπορεί να γνωρίζει αν πράγματι λαμβάνει δεδομένα από το κρυπτοσύστημα ή μια αληθινά τυχαία συνάρτηση.

Θεωρούμε block size λ και στο τέλος θα αντικαταστήσουμε $\lambda := 64$. Είναι:

$$y_i = Enc_k x_i \oplus y_{i-1} \quad (4.3)$$

Θα αποδείξω με ισχυρή επαγωγή ότι τα y_i ακολουθούν ομοιομόρφη κατανομή και είναι μεταξύ τους ανεξάρτητα.

Θεώρημα 4.1. *Τα $y_j,\ j \leq i$ ακολουθούν ομοιόμορφη κατανομή και είναι μεταξύ τους ανεξάρτητα.*

Απόδειξη.

- Το y_0 είναι το IV που επιλέγεται ομοιόμορφα τυχαία.
- Έστω ότι τα $y_i,\ j \leq i$ ακολουθούν ομοιόμορφη κατανομή και είναι ανεξάρτητα. Θα δείξω ότι και το y_{i+1} ακολουθεί ομοιόμορφη κατανομή και είναι ανεξάρτητο από όλα τα $y_i,\ i \leq j$.

  Η συνάρτηση $f(x) = c \oplus x$, με $c, x \in \{0,1\}^n$ και c σταθερά είναι μετάθεση, οπότε αν το x ακολουθεί ομοιομόρφη κατανομή τότε και το $f(x)$ ακολουθεί ομοιόμορφη κατανομή.

  Συνεπώς το $x_{i+1} \oplus y_i$ ακολουθεί ομοιόμορφη κατανομή.

  Τότε αφού η Enc είναι τυχαία συνάρτηση, δίνει y_{i+1} μια τυχαία μεταβλητή με ομοιόμορφη κατανομή που είναι ανεξάρτητη από κάθε άλλη τυχαία επιλογή.

□

Συνεπώς τα y_i αποτελούν n διαφορετικές τυχαίες μεταβλητές με ομοιόμορφη κατανομή στο σύνολο $\{0,1\}^B$.

Ζητάμε την πιθανότητα σύγκρουσης σε επιλογή n ανεξάρτητων δειγμάτων καθένα με ομοιόμορφη επιλογή από σύνολο μεγέθους $m = 2^B$.

Υπολογίζουμε την πιθανότητα μη σύγκρουσης.

$$\mathbb{P}\left[NoColl_n\right] = \mathbb{P}\left[NoColl_n \mid NoColl_{n-1}\right] \cdot \mathbb{P}\left[NoColl_{n-1} \mid NoColl_{n-2}\right] \cdot \ldots \cdot 1 = \prod_{i=0}^{n-1} \left(1 - \frac{i}{m}\right) \quad (4.4)$$

Όμως: $e^x \geq x + 1,\ \forall x \in \mathbb{R}$ (λόγω κυρτότητας).

Συνεπώς:

$$1 - \frac{i}{m} \leq \exp -\frac{i}{m} \quad (4.5)$$

Επίσης, λόγω κυρτότητας της $\exp -x$ η τέμνουσα ευθεία που περνά από τα $x = 0, x = 1$ είναι "πάνω" από την $\exp -x$ στο $[0, 1]$, οπότε

$$\exp -\chi \leq 1 + (\exp -1 - 1) \cdot x \leq 1 - \frac{1}{2}x \tag{4.6}$$

Τότε:

$$\mathbb{P}\left[NoColl_n\right] = \prod_{i=1}^{n-1}\left(1 - \frac{i}{m}\right) \leq \prod_{i=1}^{n-1}\exp -\frac{i}{m} = \exp -\frac{\sum_{i=1}^{n-1} i}{m} = \exp -\frac{n(n-1)}{2m} \leq 1 - \frac{n(n-1)}{4m} \tag{4.7}$$

Επίσης εφαρμόζοντας αντίστροφα τις ανισότητες λαμβάνουμε:

$$\begin{gathered}\mathbb{P}\left[NoColl_n\right] = \prod_{i=1}^{n-1}\left(1 - \frac{i}{m}\right) = \prod_{i=1}^{n-1}\left(1 - \frac{2i}{m} \cdot \frac{1}{2}\right) \\ \geq \prod_{i=1}^{n-1}\exp -\frac{2 * i}{m} = \exp -\frac{2\sum_{i=1}^{n-1} i}{m} = \exp -\frac{2n(n-1)}{2m} = \exp -\frac{n(n-1)}{m} \\ \geq 1 - \frac{n(n-1)}{m}\end{gathered} \tag{4.8}$$

Προκύπτει:

$$1 - \frac{n(n-1)}{m} \leq \exp -\frac{n(n-1)}{m} \leq \mathbb{P}\left[NoColl_n\right] \leq \exp -\frac{n(n-1)}{2m} \leq 1 - \frac{n(n-1)}{4m} \tag{4.9}$$

Επειδή $\mathbb{P}\left[Coll_n\right] = 1 - \mathbb{P}\left[NoColl_n\right]$ λαμβάνουμε:

$$\frac{n(n-1)}{4m} \leq 1 - \exp -\frac{n(n-1)}{2m} \leq \mathbb{P}\left[Coll_n\right] \leq 1 - \exp -\frac{n(n-1)}{m} \leq \frac{n(n-1)}{m} \tag{4.10}$$

TL;DR Τελικά είναι:

$$\mathbb{P}\left[Coll_n\right] = \prod_{i=1}^{n-1}\left(1 - \frac{i}{m}\right) \approx 1 - \exp -\frac{n(n-1)}{2m} \tag{4.11}$$

που αποτελεί μια αρκετά καλή προσέγγιση (αρκετά κοντά στην αληθινή τιμή όπως δείχνουν οι ανισότητες) όμως είναι υποεκτίμηση, γεγονός που αξιολογείται χρήσιμο αν θέλουμε να απαιτήσουμε η πιθανότητα να έχει ένα κάτω φράγμα (μπορούμε να απαιτήσουμε η "υπόεκτίμηση" να είναι κάτω φραγμένη με το φράγμα αυτό).

Για $\lambda := 64$ λαμβάνουμε $m = 2^{64}$ οπότε η πιθανότητα σύγκρουσης είναι: $1 - \exp -\frac{n(n-1}{2^{64}}$

4.3 Πλήθος n που καθιστά την επίθεση χρήσιμη

4.3.1 Ασυμπτωτική περίπτωση

Μελετάμε αρχικά την ασυμπτωτική περίπτωση, προκειμένου να δείξουμε ότι, όπως αναμένεται για μια τυχαία συνάρτηση, δεν θα ξεφύγουμε από το επίπεδο ασφαλείας του PPT αντιπάλου (απαιτείται εκθετικά μεγάλο n ως προς την παράμετρο ασφαλείας B), καθώς επίσης και για να δούμε πως για μικρές παραμέτρους ασφαλείας εκθετική δυσκολία μπορεί να είναι ανεκτή.

Η επίθεση είναι χρήσιμη αν επιτυγχάνει με αμελητέα πιθανότητα, δηλαδή αν υπάρχει πολυώνυμο $p(\lambda)$ ώστε η πιθανότητα επιτυχία να είναι $\geq \frac{1}{p(\lambda)}$, υπό την προϋπόθεση ότι το n είναι πολυωνυμικά φραγμένο (έστω από ένα $p'(\lambda)$ ως προς την παράμετρο ασφαλείας λ (αφού ο PPT αντίπαλος θα εξετάσει το πολύ πολυωνυμικά πολλά κρυπτοκείμενα).

Χρησιμοποιούμε το κάτω φράγμα της πιθανότητας σύγκρουσης απαιτώντας να είναι $\geq \frac{1}{p(\lambda)}$ ώστε να εξασφαλίσουμε ότι η πιθανότητα σύγκρουσης θα είναι και αυτή $\geq \frac{1}{p(\lambda)}$.

Συνεπώς απαιτούμε:

$$\begin{gathered}1 - \exp -\frac{n(n-1)}{2m} \geq \frac{1}{p(\lambda)} \Rightarrow 1 - \exp -\frac{p'(\lambda)}{2^\lambda} \geq \frac{1}{p(\lambda)} \Rightarrow \\ p(\lambda) \geq \frac{1}{1 - \exp -\frac{p'(\lambda)}{2^\lambda}} \overset{1-\exp -x \geq x,\ \forall x \in \mathbb{R}}{\geq} \frac{1}{\frac{p'(\lambda)}{2^\lambda}} = \frac{2^\lambda}{p'(\lambda)} \Rightarrow \\ p(\lambda)p'(\lambda) \geq 2^\lambda\end{gathered} \tag{4.12}$$

Η τελευταία σχέση είναι, ασυμπτωτικά ως προς το λ, αδύνατο να ισχύει. Συνεπώς η επίθεση, ασυμπτωτικά, δεν είναι χρήσιμη.

4.3.2 Συμπαγής περίπτωση 64 bits

Θεωρούμε $\lambda := 64$ bits.

Αυθαίρετα, ας θεωρήσουμε μια επίθεση χρήσιμη αν έχει πιθανότητα επιτυχίας $\geq \frac{1}{2}$. Για να ισχύει λαμβάνουμε:

$$1 - \exp -\frac{n(n-1)}{22^{64}} \geq \frac{1}{2} \Rightarrow -\frac{n(n-1)}{2^{65}} \geq -\ln 2 \Rightarrow n(n-1) \approx n^2 \geq 2^{65} \ln 2 \Rightarrow n \geq \sqrt{2^{65} \ln 2} \approx 5.05 \cdot 10^9 \quad (4.13)$$

Παρατηρούμε ότι ο αριθμός που προκύπτει είναι σχετικά ρεαλιστικός!

Κάθε block έχει μέγεθος 64 bits = 8 bytes. Οπότε τα $5 \cdot 10^9$ κρυπτοκείμενα αντιστοιχούν σε $40 \cdot 10^9$ bytes = 40 GB. Πρόκειται για ρεαλιστικό αριθμό. Αν αναλογιστούμε ότι σήμερα μπορεί κάποιος να έχει μια σύνδεση 100 Mbps στο διαδίκτυο, και να στέλνει συνεχώς κρυπτογραφημένα κείμενα, 40 GB κρυπτοκειμένου θα έχει μεταδοθεί σε μόλις 1 ώρα! Βέβαια θα πρέπει να ανιχνεύσουμε το διπλότυπο κρυπτοκείμενο, όμως αυτό μπορούμε να το κάνουμε αποδοτικά με κάποιο Hash Table, Δέντρο Αναζήτησης, φίλτρο Bloom, κ.λπ. (τα σύγχρονα συστήματα διαχείρισης βάσεων δεδομένων διαχειρίζονται σχετικά αποδοτικά τέτοιες ποσότητες δεδομένων.)

Η δύναμη της ασυμπτωτικής δυσκολίας, όμως, είναι πως μπορούμε να αυξήσουμε λίγο την παράμετρο ασφαλείας και το πρόβλημα να γίνεται στην συμπαγή έκδοση πολύ πιο δύσκολο. Για παράδειγμα, ας θέσουμε $\lambda = 128$. Πλέον, απαιτούνται $1.5 \cdot 10^{19}$ κρυπτοκείμενα, που αντιστοιχούν, ακόμα και με το παλιό block size, σε $122 \cdot 10^{18}$ bytes = $122 \cdot 10^9$ GB = 122 EB (exabytes). Βέβαια, ζούμε στην εποχή του exascale computing, όπου ο υπερυπολογιστής El Capitan, 1ος στο TOP500, δίνει περίπου 2 exaFLOPs υπολογιστικής δύναμης (βέβαια αυτό για πολλαπλασιασμό πυκνού πίνακα-διανύσματος και όχι για διαχείριση 122 EB δεδομένων). Αλλά ακόμα και έτσι, επιλέγουμε με μια μικρή αλλαγή $\lambda = 256$ και το πρόβλημα γίνεται δύσκολο.

5 Άσκηση 5: Σταθερό salting σε Block Cipher

5.1 (Αποδοτική) Ανίχνευση μεγέθους Block

Σκοπός είναι να βρούμε το μέγεθος του block.

Θα εκμετελλευτούμε το γεγονός ότι το κρυπτοκείμενο είναι μέγεθους ακέραιου πολλαπλάσιου του B. Έτσι αν βρούμε δύο μήνυματα μεγέθους $|m|$ και $|m|+1$ που δίνουν κρυπτοκείμενα c_a και c_b ξέρουμε ότι το μέγεθος block είναι $B = |c_b| - |c_a|$. Εκτελούμε δυαδική αναζήτηση στο $|m|$ ώστε να βρούμε μια τιμή $|m_*|$ που ικανοποιεί την ιδιότητα που θέλουμε.

Προτείνουμε τον εξής αλγόριθμο:

Αλγόριθμος 1 Εύρεση μεγέθους block B

Είσοδος: Μαντείο κρυπτογράφησης Enc
Έξοδος: Μέγεθος Block B

```
▷ Μέτρησε το μέγεθος |c_0| κρυπτοκειμένου για ένα arbitrary μήνυμα μεγέθους 1 (π.χ. το 0).
c_0 ← |Enc('0')|
▷ Βρες με εκθετική αύξηση του |m'| ένα κρυπτοκειμένου που είναι μεγαλύτερο από το |c_0|.
▷ Το |m_*| είναι σίγουρα μικρότερο ή ίσο της τιμής που θα βρούμε και ταυτόχρονα μεγαλύτερο ή ίσο του 1.
m' ← 1
c ← c_0
while c = c_0 do
    m' ← 2 · m
    c ← |Enc('0^m)|
end while
▷ Διαθέτουμε ένα διάστημα που ανήκει το |m_*|: |m_*| ∈ [1, |m'|].
▷ Εκτελούμε δυαδική αναζήτηση στο διάστημα αυτό.
L ← 1
R ← m'
while L ≠ R − 1 do
    mid ← (L+R)/2
    c ← |Enc('0^mid)|
    if c = c_0 then
        L ← mid
    else
        R ← mid
    end if
end while
B ← Enc('0^R) − c_0
```

Η ορθότητα του αλγορίθμου βασίζεται στο invariant ότι μήνυμα μήκους L δίνει πάντα κρυπτοκείμενο μήκους c_0, ενώ μήνυμα μήκους R δίνει πάντα κρυπτοκείμενο μήκους $c > c_0$. Στο τέλος της διαδικασίας, έχουμε ένα μήκος L ώστε μήκος $L+1$ δίνει κρυπτοκείμενο μήκους $c > c_0$. Συνεπώς $B = Enc('0^R) - Enc('0^L) = Enc('0^R) - c_0$.

Ο αλγοριθμός απαιτεί χρόνο $\Theta(\log B)$ για την αρχική φάση μέχρι να βρει το αρχικό m'.

Είναι $\Theta(m') = \Theta(\mathrm{B})$.

Η δυαδική αναζήτηση απαιτεί χρόνο $\Theta(\log(R-L)) = \Theta(\log(m'-1)) = \Theta(\log B)$.

Συνολικά ο αλγόριθμος απαιτεί χρόνο (και πλήθος κρυπτογραφήσεων) $\Theta(\log B)$.

Η παράμετρος ασφαλείας λ λαμβάνεται συνήθως στην 1-based μορφή του αριθμού, και όλοι οι αντίπαλοι φράσονται πολυωνυμικά ως προς αυτή την αναπαράσταση, δηλαδή με τον παρόντα συμβολισμό φράσονται πολυωνυμικά ως προς το 2^B. (θεωρούμε ως είσοδο στους αντιπάλους το 1^B).

Συνεπώς ο αλγόριθμος, αν τον θεωρήσουμε ως επίθεση θεωρείται υπερβολικά πολύ αποδοτικός, είναι στην πραγματικότητα υπογραμμικός, και μάλιστα διαβάζει μόλις λογάριθμο της εισόδου του.

Για block size 128 bits, που είναι το block size του AES (εμείς κιόλας θα έχουμε μικρότερο διότι προσθέτουμε το αλάτι), ανεξάρτητα του μεγέθους του κλειδιού, θα το βρούμε με μόλις το πολύ $\log_2 256 + \log_2(2 \cdot 256) = 17$ ερωτήσεις στο μαντείο κρυπτογράφησης.

5.2 (Αποδοτικός) Έλεγχος χρήσης Electronic Codebook (ECB) Mode of Operation

Στο ECB απλά σπάμε το μήνυμα μεγέθους $|m|$ σε μικρότερα μηνύματα μέγεθους $|B|$ το καθένα και τα κρυπτογραφούμε ανεξάρτητα. Μπορούμε εύκολα να εξετάσουμε αν όντως γίνεται αυτό, βλέποντας αν μια παράθεση του ίδιου τυχαίου μηνύματος, μεγέθους όσο το μέγεθος του block, δύο φορές προκαλεί την ίδια κρυπτογράφηση.

Φυσικά, δεν μπορούμε να είμαστε απόλυτα βέβαιοι ότι η υλοποίηση είναι ECB, αφού πάντα θα μπορούσε για περισσότερα blocks να συμβαίνει κάτι διαφορετικό. Εξάλλου, το να γνωρίζουμε αν το πρόγραμμα κρυπτογράφησης ακολουθεί το ECB, είναι μια *σημασιολογική ιδιότητα αυτού του προγράμματος*, οπότε βάσει του *Θεωρήματος Rice*, είναι undecidable να απαντήσουμε αν ακουλουθεί ECB. Ωστόσο είναι μια καλή ευριστική απάντηση να απαντήσουμε ότι χρησιμοποιεί ECB αν σε παράθεση ενός τυχαίου μηνύματος δύο φορές δώσει δύο ίδια κρυπτοκείμενα και ούτως ή άλλως αν ισχύει αυτό υπάρχει το πρόβλημα ασφάλειας που έχει το ECB.

Τελικά προτείνουμε τον εξής αλγόριθμο:

Αλγόριθμος 2 Έλεγχος χρήσης ECB

Είσοδος: Μαντείο κρυπτογράφησης Enc
Έξοδος: Απάντηση NAI/OXI χρήσης ECB.

```
▷ Αρχικά βρίσκουμε το B με χρήση του προηγούμενου αλγορίθμου
B ← (Αλγόριθμος 1)(Enc)
m ← '0'^{2B}
c ← Enc(m)
if c[0 : B] = c[B : 2B] then
    return NAI
else
    return OXI
end if
```

Η πολυπλοκότητα του αλγορίθμου είναι όπως και του αλγορίθμου 1, $\Theta(\log B)$.

5.3 (Αποδοτική) Εύρεση της salt συμβολοσειράς s

Στα παρακάτω αναφερόμαστε σε δυαδικές συμβολοσειρές, θα βρούμε δηλαδή την δυαδική αναπαράσταση της ASCII συμβολοσειράς. Συνεπώς όλα τα μεγέθη αναφέρονται σε bits, και κάθε bit έχει δύο επιλογές το 0 και το 1.

Σκοπός του αλγόριθμου, σε αδρές γραμμές, είναι να ανακαλύψει με ωμή επίθεση ένα-ένα ακολουθιακά τα bits της s από το τέλος προς την αρχή της, τοπεθετώντας τα σε χωριστό block. Πιο συγκεκριμένα με μήνυμα μεγέθους $1 + (|s| \mod B)$ (εξηγούμε σε λίγο πώς θα βρουμε το $|s| \mod B$), φροντίζουμε το τελευταίο bit να γίνει spill σε χωριστό block. Δοκιμάζουμε και τις δύο τιμές 0/1 και δεχόμαστε την τιμή που κρυπτογραφείται στο σωστό block (θα εξηγήσουμε σε λίγο πώς το κάνουμε αυτό). Προσθέτοντας ακόμα ένα bit στο μήνυμα, ένα δεύτερο bit γίνεται spill στο χωριστό αυτό block. Όμως πλέον γνωρίζουμε το τελευταίο bit. Οπότε απαιτείται να δοκιμάσουμε μόνο ένα bit. Επαναλάμβάνουμε μέχρι να έχουμε βρει όλη την συμβολοσειρά. Προκειμένου να βρουμε το μέγεθος της συμβολοσειράς στηριζόμαστε ότι το πλήθος bits απλού κειμένου είναι ίσο με το πλήθος bits κρυπτοκειμένου αν είναι πολλαπλάσιο του μεγέθους block. Έτσι κρυπτογραφούμε μήνυμα μεγέθους $|s| \mod B$, βρίσκουμε κρυπτοκείμενο μεγέθους $|c|$ και τότε $|s| = |c| - |s| \mod B$.

Για τον υπολογισμό του $|s| \mod B$, αν και θα μπορούσαμε με δυαδική αναζήτηση, επειδή ούτως ή αλλως έπειτα θα κάνουμε ανάζητηση $|s|$ βημάτων, το βρίσκουμε με γραμμική αναζήτηση ψάχνοντας τον μικρότερο αριθμό s_B που ΔΕΝ προκαλεί αύξηση του κρυπτοκειμένου (επειδή δεν μπορούμε ίσως να εξετάσουμε το μηδενικό μήνυμα, αν βρούμε $s_B = B$ κρατάμε $s_B = 0$).

Τέλος για να υπολογίσουμε την κρυπτογράφηση ενός block της επιλογής μας (με κατάλληλο zero-padding ώστε να είναι όντως block), ζητάμε την κρυπυτογράφηση του και απλά κρατάμε το πρώτο block κρυπτοκειμένου.

Συνολικά, με περισσότερη ακρίβεια, ο αλγόριθμος είναι ο εξής:

Αλγόριθμος 3 Εύρεση salt συμβολοσειράς s

Είσοδος: Μαντείο Κρυπτογράφησης Enc
Έξοδος: Salt Συμβολοσειρά s

▷ Αρχικά βρίσκουμε το B με χρήση του προηγούμενου αλγορίθμου.
$B \leftarrow$ (Αλγόριθμος 1)(Enc)
▷ Θα βρούμε το $s_B = |s| \mod B$
$c_0 \leftarrow |Enc('0'^1)|$
$s_B \leftarrow 1$
$c \leftarrow c_0)$
while $c = c_0$ **do**
  $s_B \leftarrow s_B + 1$
  $c \leftarrow |Enc('0'^{s_B})|$
end while
▷ Βρήκαμε τον μικρότερο αριθμό που προκαλεί αύξηση του κρυπτοκειμένου. Ο μικρότερος που δεν προκαλεί είναι ο προηγούμενος του.
$s_B \leftarrow s_B - 1$
▷ Αν βρήκαμε $s_B = B$ κρατάμε $s_B = 0$
if $s_B = B$ **then**
  $s_B \leftarrow 0$
end if
▷ Προχωράμε στην εύρεση της συμβολοσειράς s.
$len_s \leftarrow |Enc('0'^{s_B}))| - s_B|$
$s \leftarrow$ String of Lengthlen_s
for $i = 1, \ldots, |s|$ **do**
  $c \leftarrow Enc('0'^{s_B+i})$
  $cipher\_block \leftarrow c[: B]$ ▷ Τελευταία B bits, δηλαδή το τελευταίο block
  if $i < B - 1$ **then**
    ▷ Χρειάζεται Zero Padding στο block.
    $plain\_block\_0 \leftarrow\ '0' + s\,[|s| - i + 1 :] +' 0'^{B-i}$
    $plain\_block\_1 \leftarrow\ '1' + s\,[|s| - i + 1 :] +' 0'^{B-i}$
  else
    ▷ Το block απλού κειμένου αποτελείται μόνο από χαρακτήρες της s.
    $plain\_block\_0 \leftarrow\ '0' + s\,[|s| - i + 1 : |s| - i + B + 1]$
    $plain\_block\_1 \leftarrow\ '1' + s\,[|s| - i + 1 : |s| - i + B + 1]$
  end if
  $c_{b0} \leftarrow Enc(plain\_block\_0)$
  $c_{b1} \leftarrow Enc(plain\_block\_1)$
  if $c_{b0} = c$ **then**
    $s[i] \leftarrow 0$
  else
    ▷ Ισχυεί $c_{b1} = c$. Αν βρεθεί αλλιώς σημαίνει ότι το μαντείο κρυπτογράφησης δεν είναι συμβατό με την υπόθεση.
    $s[i] \leftarrow 1$
  end if
end for
return s

Η χρονική πολυπλοκότητα (και η πολυπλοκότητα κλήσεων στο μαντείο) είναι: $\Theta(B+|s|)$, δηλαδή γραμμικού χρόνου, γεγονός που καθιστά τον αλγόριθμο αποδοτικό.

Με άλλα λόγια, η σταθερή salt συμβολοσειρά δεν προσφέρει πρόσθετη ασφάλεια, καθώς σε γραμμικό χρόνο μπορούμε να την βρούμε και έπειτα να μπορούμε να προοσμοιώσουμε ένα μαντείο κρυπτογράφησης (ή και αποκρυπτογράφησης) του ίδιου συστήματος αλλά χωρίς την συμβολσειρά αυτή (απλά την προσθέτουμε πριν την κρυπτογράφηση, και την αφαιρούμε μετά την αποκρυπτογράφηση).

6 Άσκηση 6: Γεννήτρια Ψευδοτυχαιότητας RC4

Θα δείξουμε ότι στην γεννήτρια ψευδοτυχαιότητας RC4 το δεύτερο byte εξόδου έχει τιμή 0 με πιθανότητα 2^{-7}.

Ξεκινάμε αποδεικνύοντας το εξής λήμμα.

Λήμμα 6.1. *Έστω ότι μετά τη φάση δημιουργίας κλειδιών (KSA) ισχύει για την μετάθεση P ότι $P[2] = 0$ και $P[1] \neq 2$ (zero-indexed ο πίνακας P).*

Τότε το δεύτερο byte εξόδου είναι ίσο με 0.

Απόδειξη. Θα εκτελέσουμε "συμβολικά" την εξαγωγή των δύο πρώτων bytes της *Φάσης Παραγωγής Ψευδοτυχαίων bytes (PRGA)*. Θεωρούμε ότι η *φάση δημιουργίας κλειδιών (KSA)* έχει ολοκληρωθεί και ισχύει η υπόθεση.

Συμβολίζουμε με P' τον πίνακα μετά τον πρώτο swap και με P'' τον πίνακα μετά το δεύτερο swap.

Αρχικά $i \leftarrow 0$, $j \leftarrow 0$.

1. Στην πρώτη εκτέλεση του βρόχου, γίνεται $i' \leftarrow 1$, $j' \leftarrow P[1]$.

   (το $P[1]$ είναι byte, οπότε $P[1] \mod 256 = P[1]$.)

   Άρα $P'[1] \leftarrow P[P[1]]$
   και $P'[P[1]] \leftarrow P[1]$.

   Επειδή $P[1] \neq 2$ είναι $P'[2] = P[2] = 0$.

   Δεν μας απασχολεί το πρώτο byte εξόδου K_0.

2. Στην δεύτερη εκτέλεση του βρόχου, γίνεται $i'' \leftarrow 1 + 1 = 2$
   και $j'' \leftarrow (P[1] + P'[2]) \mod 256 = (P[1] + 0) \mod 256 = P[1]$.

   Άρα $P''[2] \leftarrow P'[P[1]] = P[1]$
   και $P''[P[1]] \leftarrow P'[2] = 0$.

   Τότε:
   $$K_1 \leftarrow P''[(P''[i''] + P''[j'']) \mod 256] = P''[(P''[2] + P''[P[1]]) \mod 256] = P''[(P[1] + 0) \mod 256] = P''[P[1]] = 0 \tag{6.1}$$

Συνεπώς το δεύτερο byte εξόδου είναι το $K_1 = 0$. □

Υπόθετουμε ότι η *φάση δημιουργίας σειράς κλειδιών (KSA)* καταλήγει σε μια τυχαία μετάθεση P.

Θα υπολογίσουμε την πιθανότητα να ισχύει η υπόθεση του Λήμματος 6.1, δηλαδή $P[2] = 0$ και $P[1] \neq 0$ (εφεξής το ονομάζω ενδεχόμενο E).

Αφού υποθέτουμε τυχαία μετάθεση P, το ζεύγος αριθμών $(P[1], P[2])$ λαμβάνει ομοιόμορφα τιμές από το σύνολο $\{0, \dots, 255\}^2$ εκτός των ίσων $P[1] = P[2]$ ώστε να είναι μετάθεση. Δηλαδή το ζεύγος λαμβάνει $256 \cdot 255$ ομοιόμορφα πιθανές τιμές. Αποδεκτές είναι 254 από αυτές. (το 0 πάει στο $P[1]$ και το $P[2]$ έχει όλες τις επιλογές εκτός του 2 και του 0).

Συνεπώς η πιθανότητα του E είναι:
$$\mathbb{P}[E] = \frac{254}{255 \cdot 256} = \frac{254}{255} \cdot 2^{-8} \tag{6.2}$$

Υποθέτουμε πως αν δεν ισχύει το ενδεχόμενο E, τότε το δεύτερο byte εξόδου είναι πράγματι ομοιόμορφα τυχαίο, όπως θα το αντιλαμβανόταν ένας PPT αλγόριθμος αν η RC4 είναι ψευδοτυχαία συνάρτηση, οπότε έχει πιθανότητα να είναι 0:
$$\mathbb{P}[K_1 = 0 \mid \neg E] = \frac{1}{256} = 2^{-8} \tag{6.3}$$

Τότε η πιθανότητα το δεύτερο byte εξόδου να είναι 0 είναι:
$$\begin{gathered} \mathbb{P}[K_1 = 0] = \mathbb{P}[K_1 = 0 \mid E] \cdot \mathbb{P}[E] + \mathbb{P}[K_1 = 0 \mid \neg E] \cdot \mathbb{P}[\neg E] \overset{6.1}{=} \\ 1 \cdot \frac{254}{255} \cdot 2^{-8} + 2^{-8} \cdot (1 - 2^{-8}) = \left(\frac{254}{255} + \frac{255}{256}\right) \cdot 2^{-8} \approx 2 \cdot 2^{-8} = 2^{-7} \end{gathered} \tag{6.4}$$

Συνεπώς η πιθανότητα το δεύτερο byte εξόδου να είναι 0 είναι ίση (προσεγγιστικά) με 2^{-7} και όχι με 2^{-8} που θα έπρεπε αν είχε όντως μια ομοιόμορφα τυχαία τιμή.

7 Άσκηση 7: Ψευδοτυχαίες Συναρτήσεις

Στα παρακάτω υποθέτουμε ότι η $F : \{0,1\}^n \times \{0,1\}^n \mapsto \{0,1\}^n$ είναι μια ψευδοτυχαία συνάρτηση (εφεξής συμβολίζω F_k την μερική εφαρμογή της F στο πρώτο της όρισμα με το k), δηλαδή για κάθε PPT (Probabilistic Polynomial Time) αλγόριθμο D (συμβολίζω $D^{\mathcal{O}}$ θεωρώντας ότι ο D έχει πρόσβαση στο μαντείο τυχαιότητας $\mathcal{O}$) υπάρχει μια αμελητέα συνάρτηση $negl(n)$ τέτοια ώστε:

$$\left| \mathbb{P}_{k \xleftarrow{R} \{0,1\}^n} \left[D^{F_k()}(1^n) = 1 \right] - \mathbb{P}_{f \xleftarrow{R} Func_n} \left[D^{f()}(1^n) = 1 \right] \right| \leq negl(n) \tag{7.1}$$

7.1 Συνάρτηση $F_1(k,x) = F(k,x) \mid\mid 0$

Θα δείξουμε ότι η $F_1(k,x) = F(k,x) \mid\mid 0$ δεν είναι ψευδοτυχαία συνάρτηση.

Ένας PPT διαχωριστής D_1 είναι ο εξής:

Επιλέγει μια arbitrary τιμή, π.χ. $x = 0$. Δίνει το x στο μαντείο τυχαιότητας $\mathcal{O}$, και λαμβάνει μια απάντηση y. Αν το τελευταίο bit του y είναι το 0, επιστρέφει ΝΑΙ, αλλιώς επιστρέφει ΟΧΙ.

Ο D_1 είναι ένας PPT αλγόριθμος.

Με μαντείο $\mathcal{O}$ την ψευδοτυχαία συνάρτηση F_k ο αλγοριθμος επιστρέφει ΝΑΙ με πιθανότητα 1.

Με μαντείο $\mathcal{O}$ την τυχαία συνάρτηση f ο αλγόριθμος επιστρέφει ΝΑΙ με πιθανότητα $\frac{1}{2}$. (αυτό ισχύει διότι αφού η f είναι τυχαία συνάρτηση, τότε το τελευταίο bit είναι ανεξάρτητο και έχει τιμή 0 ή 1 με πιθανότητα 1/2.)

Άρα:

$$\left| \mathbb{P}_{k \xleftarrow{R} \{0,1\}^n} \left[D_1^{F_k()}(1^n) = 1 \right] - \mathbb{P}_{f \xleftarrow{R} Func_n} \left[D_1^{f()}(1^n) = 1 \right] \right| = \left| 1 - \frac{1}{2} \right| = \frac{1}{2} \tag{7.2}$$

Ο αριθμός $\frac{1}{2}$ είναι μεγαλύτερος από κάθε αμελητέα συνάρτηση $negl(n)$. Συνεπώς η $F_1(k,x)$ δεν είναι ψευδοτυχαία συνάρτηση.

7.2 Συνάρτηση $F_2(k,x) = F(k,x)$ `xor` x

Θα δείξουμε ότι η $F_2(k,x) = F(k,x) \oplus x$ είναι ψευδοτυχαία συνάρτηση.

(συμβολίζω όπως πριν F_{2k} την μερική εφαρμογή της F_2 στο πρώτο όρισμα με το k.)

Έστω ένας PPT διαχωριστής D_2 της F_2. Κατασκευάζω τον εξής PPT διαχωριστή D για την F (με είσοδο ένα μαντείο $\mathcal{O}$):

Κατασκευάζω ένα μαντείο $\mathcal{O}'$ που με είσοδο x, δίνει το x στο μαντείο $\mathcal{O}$ λαμβάνει μια απάντηση y και επιστρέφει $y \oplus x$, δηλαδή:

$$\mathcal{O}'(x) = \mathcal{O}(x) \oplus x \tag{7.3}$$

Εκτελώ τον διαχωριστή D_2 με είσοδο το μαντείο $\mathcal{O}'$ και επιστρέφω την απάντησή του.

Αν στον D δοθεί ως μαντείο $\mathcal{O}$ η F_k τότε το μαντείο $\mathcal{O}'$ είναι:

$$\mathcal{O}'(x) = \mathcal{O}(x) \oplus x = F(k,x) \oplus x = F_{2k}(x) \tag{7.4}$$

Συνεπώς στον D_2 δίνεται ως μαντείο η F_{2k}.

Ο D επιτυγχάνει ανν επιτυγχάνει ο D_2 που συμβαίνει συνεπώς με πιθανότητα: $\mathbb{P}_{k \xleftarrow{R} \{0,1\}^n} \left[D_2^{F_{2k}()}(1^n) = 1 \right]$.

Αν στον D δοθεί ως μαντείο $\mathcal{O}$ η τυχαία συνάρτηση f τότε το μαντείο $\mathcal{O}'$ είναι:

$$\mathcal{O}'(x) = \mathcal{O}(x) \oplus x = f(x) \oplus x \tag{7.5}$$

Το $\mathcal{O}'(x) = f(x) \oplus x$ είναι 1-1 και επί συνάρτηση, δηλαδή το $\mathcal{O}'(x)$ είναι μια μετάθεση του $\{0,1\}^n$. Αφού η f είναι η τυχαία συνάρτηση, το $f(x)$ λαμβάνει ομοιόμορφα τυχαία οποιαδήποτε τιμή του $\{0,1\}^n$ ανεξάρτητη του x. Συνεπώς και η μετάθεση $\mathcal{O}'(x) = f(x) \oplus x$ λαμβάνει ομοιόμορφα τυχαία τιμή από το $\{0,1\}^n$, ανεξάρτητη του x. Άρα το μαντείο $\mathcal{O}'(x)$ είναι μια τυχαία συνάρτηση. Συνεπώς στον D_2 δίνεται ως μαντείο μια τυχαία συνάρτηση.

Ο D επιτυγχάνει ανν επιτυγχάνει ο D_2 που συμβαίνει συνεπώς με πιθανότητα: $\mathbb{P}_{f \xleftarrow{R} Func_n} \left[D_2^{f()}(1^n) = 1 \right]$.

Όμως η f είναι ψευδοτυχαία συνάρτηση. Συνεπώς υπάρχει μια αμελητέα συνάρτηση $negl(n)$ τέτοια ώστε:

$$\begin{aligned}\left|\mathbb{P}_{k\overset{R}{\leftarrow}\{0,1\}^n}\left[D^{F_k()}(1^n)=1\right]-\mathbb{P}_{f\overset{R}{\leftarrow}Func_n}\left[D^{f()}(1^n)=1\right]\right|\leq negl(n)\Rightarrow\\ \left|\mathbb{P}_{k\overset{R}{\leftarrow}\{0,1\}^n}\left[D_2^{F_{2k}()}(1^n)=1\right]-\mathbb{P}_{f\overset{R}{\leftarrow}Func_n}\left[D_2^{f()}(1^n)=1\right]\right|\leq negl(n)\end{aligned} \tag{7.6}$$

Η τελευταία ισχύει για κάθε PPT διαχωριστή D_2 της F_2.

Συνεπώς η F_2 είναι ψευδοτυχαία συνάρτηση.

7.3 Συνάρτηση $F_3(k,x)=F(k,x \texttt{ xor } 1^n)$

Θα δείξουμε ότι η $F_3(k,x)=F(k,x\oplus 1^n)$ είναι ψευδοτυχαία συνάρτηση.

(συμβολίζω όπως πριν F_{3k} την μερική εφαρμογή της F_3 στο πρώτο όρισμα με το k.)

Έστω ένας PPT διαχωριστής D_3 της F_3. Κατασκευάζω τον εξής PPT διαχωριστή D της F (με είσοδο ένα μαντείο $\mathcal{O}$).

Κατασκευάζω το μαντείο $\mathcal{O}'$:

$$\mathcal{O}'(x)=\mathcal{O}(x\oplus 1^n) \tag{7.7}$$

Εκτελώ τον διαχωριστή D_3 με είσοδο το μαντείο $\mathcal{O}'$ και επιστρέφω την απάντησή του.

Αν στον D δοθεί ως μαντείο $\mathcal{O}$ η συνάρτηση F τότε:

$$\mathcal{O}'(x)=\mathcal{O}(x\oplus 1^n)=F_k(x\oplus 1^n)=F_{3k}(x) \tag{7.8}$$

Ο D επιτυγχάνει ανν επιτυγχάνει ο D_3 που συμβαίνει με πιθανότητα: $\mathbb{P}_{k\overset{R}{\leftarrow}\{0,1\}^n}\left[D_3^{F_{3k}()}(1^n)=1\right]$.

Αν στον D δοθεί ως μαντείο $\mathcal{O}$ η τυχαία συνάρτηση f τότε:

$$\mathcal{O}'(x)=\mathcal{O}(x\oplus 1^n)=f(x\oplus 1^n) \tag{7.9}$$

Η $h(x)=x\oplus 1^n=\sim x$ είναι μια 1-1 συνάρτηση. Συνεπώς αφού η f είναι τυχαία συνάρτηση αντιστοιχεί σε όλες τις τιμές του πεδιού ορισμού της ανεξάρτητες ομοιόμορφα τυχαίες τιμές. Αφού η h είναι 1-1, διαφορετικά x του πεδίου ορισμού του $\mathcal{O}'$ καταλήγουν σε διαφορετικά ορίσματα της f, οπότε τελικά το $\mathcal{O}'$ αντιστοιχεί σε όλο το πεδίο ορισμού ανεξάρτητες ομοιόμορφα τυχαίες τιμές, οπότε το $\mathcal{O}'$ είναι μια τυχαία συνάρτηση.

Ο D επιτυγχάνει ανν επιτυγχάνει ο D_3 που συμβαίνει συνεπώς με πιθανότητα: $\mathbb{P}_{f\overset{R}{\leftarrow}Func_n}\left[D_3^{f()}(1^n)=1\right]$.

Η f είναι ψευδοτυχαία συνάρτηση. Συνεπώς υπάρχει μια αμελητέα συνάρτηση $negl(n)$ τέτοια ώστε:

$$\begin{aligned}\left|\mathbb{P}_{k\overset{R}{\leftarrow}\{0,1\}^n}\left[D^{F_k()}(1^n)=1\right]-\mathbb{P}_{f\overset{R}{\leftarrow}Func_n}\left[D^{f()}(1^n)=1\right]\right|\leq negl(n)\Rightarrow\\ \left|\mathbb{P}_{k\overset{R}{\leftarrow}\{0,1\}^n}\left[D_3^{F_{3k}()}(1^n)=1\right]-\mathbb{P}_{f\overset{R}{\leftarrow}Func_n}\left[D_3^{f()}(1^n)=1\right]\right|\leq negl(n)\end{aligned} \tag{7.10}$$

Η τελευταία ισχύει για κάθε PPT διαχωριστή της F_3.

Συνεπώς η F_3 είναι ψευδοτυχαία συνάρτηση.

7.4 Συνάρτηση $F_4(k,x)=F(k,x)\ ||\ F\left(k,F(k,x)\right)$

Αρχικά παρατηρούμε ότι $F_4:\{0,1\}^n\times\{0,1\}^n\mapsto\{0,1\}^{\mathbf{2n}}$, δηλαδή η έξοδος είνα διπλάσιου μήκους του ορίσματος x.

Θα δείξουμε ότι η $F_4(k,x)=F(k,x)\ ||\ F\left(k,F(k,x)\right)$ είναι ψευδοτυχαία συνάρτηση.

(συμβολίζω όπως πριν F_{4k} την μερική εφαρμογή της F_4 στο πρώτο όρισμα με το k.)

Έστω ένας PPT διαχωριστής D_4 της F_4 (με είσοδο ένα μαντείο $\mathcal{O}$). Κατασκευάζω τον εξής διαχωριστή της F.

Κατασκευάζω το μαντείο $\mathcal{O}'$:

$$\mathcal{O}'(x)=\mathcal{O}(x)\ ||\ \mathcal{O}\left(\mathcal{O}(x)\right) \tag{7.11}$$

Εκτελώ τον διαχωριστή D_4 με είσοδο το μαντείο $\mathcal{O}'$ και επιστρέφω την απάντησή του.

Αν στον D δοθεί ως μαντείο $\mathcal{O}$ η συνάρτηση F τότε:

$$\mathcal{O}'(x) = F_k(x) \,||\, F_k\left(F_k(x)\right) = F_{4k}(x) \tag{7.12}$$

Ο D επιτυγχάνει ανν επιτυγχάνει ο D_4, που συμβαίνει με πιθανότητα: $\mathbb{P}_{k \xleftarrow{R} \{0,1\}^n}\left[D_4^{F_{4k}()}(1^n) = 1\right]$

Αν στον D δοθεί ως μαντείο $\mathcal{O}$ η τυχαία συνάρτηση f τότε:

$$\mathcal{O}'(x) = f(x) \,||\, f\left(f(x)\right) \tag{7.13}$$

Αποδεικνύω ότι το μαντείο $\mathcal{O}'$ διακρίνεται από PPT αλγόριθμο $\tilde{D}$ από μια τυχαία συνάρτηση το πολύ με αμελητέα πιθανότητα.

Θεώρημα 7.1. *Έστω μια ομοιόμορφα τυχαία συνάρτηση $f : \{0,1\}^n \mapsto \{0,1\}^n$.*

Τότε η συνάρτηση $G : \{0,1\}^n \mapsto \{0,1\}^{2n}$:

$$G(x) = f(x) \,||\, f\left(f(x)\right) \tag{7.14}$$

διακρίνεται από PPT αλγόριθμο $\tilde{D}$ από μια τυχαία συνάρτηση με αμελητέα πιθανότητα.

Ακριβέστερα, για κάθε PPT αλγόριθμο $\tilde{D}$ με είσοδο ένα μαντείο $\tilde{\mathcal{O}}$ ισχύει ότι:

$$\left|\mathbb{P}_{f \xleftarrow{R} Func_n}\left[\tilde{D}^{G_f()}(1^n) = 1\right] - \mathbb{P}_{f^* \xleftarrow{R} Func_{2n}}\left[\tilde{D}^{f^*()}(1^n) = 1\right]\right| \leq negl(n) \tag{7.15}$$

Απόδειξη. Διακρίνουμε περιπτώσεις:

1. Έστω ότι κανένα ερώτημα δεν δίνει απάντηση $\tilde{\mathcal{O}}(x) = x \,||\, y$. (το πρώτο μισό ίδιο με το όρισμα.)

   Αν το $\tilde{\mathcal{O}}$ είναι η τυχαία συνάρτηση f^* τότε όλες οι απαντήσεις του μαντείου είναι ομοιόμορφα τυχαίες μεταβλητές ανεξάρτητες από κάθε άλλη τιμή (συμπεριλαμβανομένου του ορίσματος που ερωτήθηκε στο μαντείο)

   Έστω ότι το $\tilde{\mathcal{P}}$ είναι η συνάρτηση $G_f(x) = f(x) \,||\, f\left(f(x)\right)$. Λόγω της υπόθεσης $f(x) \neq x$ και αφού η f τυχαία συνάρτηση, τα $f(x)$ και $f\left(f(x)\right)$ είναι μεταξύ τους ανεξάρτητα και επίσης ομοιόμορφα τυχαία. Συνεπώς το $f(x) \,||\, f\left(f(x)\right)$ είναι και αυτό μια ομοιόμορφα τυχαία μεταβλητή ανεξάρτητη από κάθε άλλη τιμή (συμπεριλαμβανομένου του ορίσματος που ερωτήθηκε στο μαντείο.)

   Άρα ο αλγόριθμος D και για τα δύο μαντεία, βλέπει ως απαντήσεις του μαντείου ομοιόμορφα τυχαίες μεταβλητές ανεξάρτητες μεταξύ τους και ανεξάρτητες από το όρισμα που ζητείται. Άρα και για τα δύο μαντεία η πιθανότητα επιτυχίας είναι ίδια.

2. Έστω ότι κάποιο ερώτημα δίνει απάντηση $\tilde{\mathcal{O}}(x) = x \,||\, y$.

   Θα δείξουμε ότι αυτή η περίπτωση, εφεξής ενδεχόμενο E, έχει αμελητέα πιθανότητα.

   Υπολογίζω την πιθανότητα του $\neg E$, δηλαδή κανένα ερώτημα να μην καταλήξει στο ζητούμενο.

   Έστω $q(n)$ το πολυωνυμικό πλήθος διαφορετικών ορισμάτων που ζητά ο $\tilde{D}$ από το μαντείο του.

   Έστω ότι το $\tilde{\mathcal{O}}$ είναι η τυχαία συνάρτηση f^*. Η τιμή επιστροφής είναι ανεξάρτητη του ορίσματος και ομοιόμορφα τυχαία. Τότε:

   $$\mathbb{P}\left[\neg E\right] = \mathbb{P}\left[f^*(x) \neq x \,||\, y\right]^{q(n)} = \left(\frac{(2^n - 1) \cdot 2^n}{2^{2n}}\right)^{q(n)} = (1 - 2^{-n})^{q(n)} \tag{7.16}$$

   Άρα:

   $$\mathbb{P}\left[E\right] = 1 - \mathbb{P}\left[\neg E\right] = 1 - (1 - 2^{-n})^{q(n)} \tag{7.17}$$

   Από Ανισότητα Bernoulli λαμβάνουμε:

   $$\mathbb{P}\left[E\right] = 1 - (1 - 2^{-n})^{q(n)} \leq 1 - (1 - q(n) \cdot 2^{-n}) = q(n) \cdot 2^{-n} \tag{7.18}$$

   Η $q(n) \cdot 2^{-n}$ είναι αμελητέα ως γινόμενο πολυωνύμου $q(n)$ και αμελητέας συνάρτησης n.

   Άρα και η $\mathbb{P}\left[E\right](n)$ είναι αμελητέα.

Γενικά για δύο ενδεχομενα A, B ισχύει ότι:

$$\begin{aligned}
&|\mathbb{P}[A] - \mathbb{P}[B]| \\
&= |\mathbb{P}[A \mid E] \cdot \mathbb{P}[E] + \mathbb{P}[A \mid \neg E] \cdot \mathbb{P}[\neg E] - \mathbb{P}[B \mid E] \cdot \mathbb{P}[E] - \mathbb{P}[B \mid \neg E] \cdot \mathbb{P}[\neg E]| \\
&= |(\mathbb{P}[A \mid \neg E] - \mathbb{P}[B \mid \neg E]) \cdot (1 - negl(n)) + (\mathbb{P}[A \mid E] - \mathbb{P}[B \mid E]) \cdot negl(n)| \\
&\leq |(\mathbb{P}[A \mid \neg E] - \mathbb{P}[B \mid \neg E]) \cdot (1 - negl(n))| + |(\mathbb{P}[A \mid E] - \mathbb{P}[B \mid E]) \cdot negl(n)| \\
&\leq |(\mathbb{P}[A \mid \neg E] - \mathbb{P}[B \mid \neg E]) \cdot (1 - negl(n))| + |(\mathbb{P}[A \mid E] - \mathbb{P}[B \mid E])| \cdot negl(n) \\
&\leq |(\mathbb{P}[A \mid \neg E] - \mathbb{P}[B \mid \neg E]) \cdot (1 - negl(n))| + 1 \cdot negl(n) \\
&\leq |\mathbb{P}[A \mid \neg E] - \mathbb{P}[B \mid \neg E]| + negl(n)
\end{aligned} \tag{7.19}$$

Από αυτή την σχέση προκύπτει ότι ισχύει:

$$\begin{aligned}
&\left|\mathbb{P}_{f \xleftarrow{R} Func_n}\left[\tilde{D}^{G_f()}(1^n) = 1\right] - \mathbb{P}_{f^* \xleftarrow{R} Func_{2n}}\left[\tilde{D}^{f^*()}(1^n) = 1\right]\right| \\
&\leq \left|\mathbb{P}_{f \xleftarrow{R} Func_n}\left[\tilde{D}^{G_f()}(1^n) = 1 \mid \neg E\right] - \mathbb{P}_{f^* \xleftarrow{R} Func_{2n}}\left[\tilde{D}^{f^*()}(1^n) = 1 \mid \neg E\right]\right| + negl(n) \\
&= 0 + negl(n) = negl(n)
\end{aligned} \tag{7.20}$$

□

Από το Θεώρημα 7.1 προκύπτει ότι ο PPT διαχωριστής D_4 της F_4 διακρίνει με αμελητέα πιθανότητα το μαντείο $\mathcal{O}'$ από την τυχαία συνάρτηση, οπότε:

$$\mathbb{P}_{f \xleftarrow{R} Func_n}\left[D_4^{\mathcal{O}'}(1^n) = 1\right] \leq \mathbb{P}_{f \xleftarrow{R} Func_{2n}}\left[D_4^{f()}(1^n) = 1\right] + negl(n) \tag{7.21}$$

Ο D επιτυγχάνει ανν επιτυγχάνει ο D_4 δηλαδή με πιθανότητα:

$$\mathbb{P}_{f \xleftarrow{R} Func_n}\left[D_4^{\mathcal{O}'}(1^n) = 1\right] \leq \mathbb{P}_{f \xleftarrow{R} Func_{2n}}\left[D_4^{f()}(1^n) = 1\right] + negl(n) \tag{7.22}$$

Τελικά, ο D είναι ένας PPT διαχωριστής της F. Άρα:

$$\begin{aligned}
&\left|\mathbb{P}_{k \xleftarrow{R} \{0,1\}^n}\left[D^{F_k()}(1^n) = 1\right] - \mathbb{P}_{f \xleftarrow{R} Func_n}\left[D^{f()}(1^n) = 1\right]\right| \leq negl(n) \Rightarrow \\
&\left|\mathbb{P}_{k \xleftarrow{R} \{0,1\}^n}\left[D_4^{F_{4k}()}(1^n) = 1\right] - \mathbb{P}_{f^* \xleftarrow{R} Func_{2n}}\left[D_4^{f^*()}(1^n) = 1\right]\right| + negl'(n) \leq negl(n) \Rightarrow \\
&\left|\mathbb{P}_{k \xleftarrow{R} \{0,1\}^n}\left[D_4^{F_{4k}()}(1^n) = 1\right] - \mathbb{P}_{f^* \xleftarrow{R} Func_{2n}}\left[D_4^{f^*()}(1^n) = 1\right]\right| \leq negl(n)
\end{aligned} \tag{7.23}$$

Η τελευταία ισχύει για κάθε PPT διαχωριστή της F_4.

Άρα η F_4 είναι ψευδοτυχαία συνάρτηση.

8 Άσκηση 8: Περίοδος Γεννήτριας Ψευδοτυχαιότητας Blum-Blum-Shub

8.1 Ερώτημα α: Υπολογισμός περιόδου

8.1.1 Υπολογισμός περιόδου

Έστω δύο πρώτοι αριθμούς $p, q \in \mathbb{P}$ με $p \equiv q \equiv 3 \pmod 4$ και έστω $n = pq$.

Έστω ένας αριθμός s_0 σχετικά πρώτος με τον n.

Το i-οστό bit εξόδου της γεννήτριας, $i \geq 1$ (το $i \neq 0$ έχει σημασία όπως θα δούμε) είναι:

$$z_i = \left(s_0^{2^i} \mod n\right) \mod 2 \tag{8.1}$$

Αν $s_0^{2^i}$ είναι περιοδική με περιόδο T τότε το ίδιο ισχύει και για την z_i. Ωστόσο, λόγω του $\mod 2$ θα μπορούσε να προκύπτει και μικρότερη περίοδος για το z_i από το T αν το μοτίβο άρτιου/περιττού εντός της ίδιας περιόδου T επαναληφθεί πολλές φορές. Παρακάτω το αγνόουμε αυτό.

Έτσι σκοπός είναι να αποδείξουμε ότι η συνάρτηση $s_0^{2^i} \mod n$ είναι περιοδική με ελάχιστη περίοδο $T > 0$. Κρίνεται σκόπιμο να αποδείξουμε πως έχουμε βρει την ελάχιστη περίοδο, ειδικά για κρυπτογραφική εφαρμογή, διότι διαφορετικά μπορεί να επαναλαμβάνονται τιμές πιο συχνά από ότι πιστεύουμε, υποβαθμίζοντας την ασφάλεια.

Η συνάρτηση $z_i = s_0^{2^i} \mod n$ είναι περιοδική με περίοδο T αν και μόνο αν για κάθε $i \in \mathbb{N}^*$ ισχύει ότι:

$$s_0^{2^{i+T}} \equiv s_0^{2^i} \pmod n \overset{i \geq 1}{\iff} \left(s_0^2\right)^{2^{i-1+T}} \equiv \left(s_0^2\right)^{2^{i-1}} \pmod n \tag{8.2}$$

Έστω $s_1 = s_0^2 \mod n$. Επειδή ο s_0 είναι σχετικά πρώτος με τον n είναι $s_0 \in U(\mathbb{Z}_n)$, οπότε αφού η $(U(\mathbb{Z}_n), \cdot)$ είναι ομάδα, τότε και $s_1 = s_0^2 \mod n \in U(\mathbb{Z}_n)$.

Τότε η συνάρτηση z_i είναι περιοδική αν και μόνο αν για κάθε $iin\mathbb{N}$ (προσέχουμε ότι σε σχέση με πριν έχουμε θέσει $i - 1 \leftarrow i$, οπότε λαμβάνουμε και την τιμή 0):

$$\begin{aligned} & s_1^{2^{i+T}} \equiv s_1^{2^i} \pmod n \overset{s_1 \in U(\mathbb{Z}_n)}{\iff} \\ & s_1^{2^{i+T} - 2^i} \equiv 1 \pmod n \iff \\ & \operatorname{ord}_n(s_1) \mid (2^{i+T} - 2^i) \iff \\ & 2^{i+T} \equiv 2^i \pmod{\operatorname{ord}_n(s_1)} \end{aligned} \tag{8.3}$$

Είναι $\operatorname{ord}_n(s_0) \mid \lambda(n)$, όπου $\lambda(n)$ η συνάρτηση Carmichael.

Επίσης:

$$\lambda(n) = \lambda(pq) = \operatorname{lcm}(\lambda(p), \lambda(q)) = \operatorname{lcm}(\phi(p), \phi(q)) = \operatorname{lcm}(p-1, q-1) \tag{8.4}$$

Όμως $p - 1 \equiv q - 1 \equiv 3 - 1 \equiv 2 \pmod 4$.

Συνεπώς $4 \nmid p - 1$, $4 \nmid q - 1$ οπότε $4 \nmid \operatorname{lcm}(p-1, q-1) = \lambda(n)$.

Άρα αφού $\operatorname{ord}_n(s_0) \mid \lambda(n)$, τότε $4 \nmid \operatorname{ord}_n(s_0)$.

Συνεπώς είτε ο $\operatorname{ord}_n(s_0)$ είναι περιττός είτε $\operatorname{ord}_n(s_0) = 2\mu$, όπου μ περιττός.

Η κυκλική υποομάδα $\langle s_0 \rangle$ του $U(\mathbb{Z}_n)$ έχει τάξη $\operatorname{ord}_n(s_0)$.

Η κυκλική υποομάδα $\langle s_0^2 \rangle$ της κυκλικής ομάδας $\langle s_0 \rangle$ έχει τότε τάξη:

$$\operatorname{ord}\left(\langle s_0^2 \rangle\right) = \frac{\operatorname{ord}\left(\langle s_0 \rangle\right)}{\gcd\left(\operatorname{ord}\left(\langle s_0 \rangle\right), 2\right)} \tag{8.5}$$

Διακρίνω τις 2 πιθανές περιπτώσεις για το $\operatorname{ord}_n(s_0)$ που αναφέρθηκαν πριν:

1. Αν $\operatorname{ord}_n(s_0)$ περιττός, τότε και ο $\operatorname{ord}\left(\langle s_0^2 \rangle\right)$ περιττός (διότι το κλάσμα έχει στον αριθμητή περιττό αριθμό).

2. Αν $\text{ord}_n(s_0) = 2\mu$, όπου μ περιττός, τότε $\gcd(\text{ord}(\langle s_0\rangle), 2) = 2$, οπότε:

$$\text{ord}\left(\langle s_0^2\rangle\right) = \frac{\text{ord}\left(\langle s_0\rangle\right)}{\gcd\left(\text{ord}\left(\langle s_0\rangle\right), 2\right)} = \frac{2\mu}{2} = \mu \tag{8.6}$$

Όμως ο μ περιττός, οπότε και ο $\text{ord}\left(\langle s_0^2\rangle\right)$ περιττός.

Άρα ο $\text{ord}_n(s_1) = \text{ord}\left(\langle s_0^2\rangle\right)$ είναι περιττός.

Συνεπώς για κάθε $\kappa \in \mathbb{N}$ είναι $\gcd\left(2^\kappa, \text{ord}_n(s_1)\right) = 1$.

Τότε:

$$\begin{gathered} 8.3 \iff 2^{i+T} \equiv 2^i \pmod{\text{ord}_n(s_1)} \iff \\ 2^T \equiv 1 \pmod{\text{ord}_n(s_1)} \iff \\ \text{ord}_{\text{ord}_n(s_1)}(2) \mid T \end{gathered} \tag{8.7}$$

Ο μικρότερος αριθμός $T > 0$ ώστε να ισχύει η τελευταία, δηλαδή ισοδύναμα ώστε να ισχύει η αρχική, δηλαδή ισοδύναμα ώστε η $s_0^{2^i} \mod n$ να είναι περιοδική με ελάχιστη περίοδο T είναι:

$$T = \text{ord}_{\text{ord}_n(s_1)}(2) = \text{ord}_{\text{ord}_n(s_0^2 \mod n)}(2) \tag{8.8}$$

8.1.2 Άνω φράξιμο περιόδου

Από τον ορισμό της συνάρτησης Carmichael, για κάθε $\tilde{n}$ και για κάθε $a \in U(\mathbb{Z}^{\tilde{n}})$ ισχύει ότι:

$$a^{\lambda(\tilde{n})} \equiv 1 \pmod{\tilde{n}} \tag{8.9}$$

οπότε $\text{ord}_{\tilde{n}}(a) \mid \lambda(\tilde{n})$.

Το $\text{ord}_n(s_1)$ είναι περιττό οπότε είναι σχετικά πρώτο με το 2. Συνεπώς $\text{ord}_{\text{ord}_n(s_1)}(2) \mid \lambda\left(\text{ord}_n(s_1)\right)$.

Επίσης είναι και $\text{ord}_n(s_1) \mid \lambda(n)$.

Αποδεικνούμε το παρακάτω λήμμα.

Λήμμα 8.1. *Έστω αριθμός n και έστω αριθμοί a, b σχετικά πρώτοι με το n.*

Αν $a \mid b$ τότε $\lambda(a) \mid \lambda(b)$.

Απόδειξη. Είναι:

$$\lambda(n) = \begin{cases} \phi(n) & \text{αν } n \text{ είναι 1, 2, 4, ή περιττή δύναμη πρώτου} \\ \frac{1}{2}\phi(n) & \text{αν } n = 2^r,\ r \geq 3 \\ \text{lcm}\left((\lambda(n_1), \lambda(n_2), \ldots, \lambda(n_k))\right) & \text{αν } n = n_1 n_2 \ldots n_k \text{ όπου } n_1, n_2, \ldots, n_k \text{ είναι δυνάμεις διακριτών πρώτων} \end{cases} \tag{8.10}$$

Αφού $a|b$ τότε η ανάλυση του a σε πρώτους παράγοντες έχει τους ίδιους πρώτους παράγοντες με του b, ενδεχεμένως σε χαμηλότερη δύναμη (αποδεκτή και η μηδενική δύναμη).

Για p πρώτο και $i < j$ είναι:

Αν $i \neq 0$: $\phi(p^i) = p^{i-1} \cdot (p-1) \mid p^{j-1} \cdot (p-1) = \phi(p^j)$.

Αν $i = 0$ $\phi(p^0) = \phi(1) = 1 \mid \phi(p^j)$.

Άρα $\phi(p^i) \mid \phi(p^j)$.

Επίσης αν $a_1 \mid a_2$, $b_1 \mid b_2$, τότε $\text{lcm}(a_1, b_1) \mid \text{lcm}(a_2, b_2)$, διότι το Ελάχιστο Κοινό Πολλαπλάσιο υπολογίζεται ως το γινόμενο του μέγιστου βαθμού ως προς τους δύο αριθμούς, κάθε πρώτου παράγοντα και των δύο αριθμών.

Συνεπώς είναι $\lambda(a) \mid \lambda(b)$. □

Άρα, αφού $\text{ord}_n(s_1) \mid \lambda(n)$ τότε $\lambda\left(\text{ord}_n(s_1)\right) \mid \lambda\left(\lambda(n)\right)$.

Αφού $\text{ord}_{\text{ord}_n(s_1)}(2) \mid \lambda\left(\text{ord}_n(s_1)\right)$, τελικά:

$$T = \text{ord}_{\text{ord}_n(s_1)}(2) \mid \lambda\left(\lambda(n)\right) \tag{8.11}$$

Όμως:

$$\lambda(n) = \lambda(pq) = \operatorname{lcm}(\lambda(p), \lambda(q)) = \operatorname{lcm}(\phi(p), \phi(q)) = \operatorname{lcm}(p-1, q-1) \tag{8.12}$$

Ισχύει από τον αναδρομικό ορισμό της συνάρτησης Carmichael ότι: $\lambda(\operatorname{lcm}(a,b)) = \operatorname{lcm}(\lambda(a), \lambda(b))$.

$$\lambda(\lambda(n)) = \lambda(\operatorname{lcm}(p-1, q-1)) = \operatorname{lcm}(\lambda(p-1), \lambda(q-1)) = \frac{\lambda(p-1)\cdot\lambda(q-1)}{\gcd(\lambda(p-1), \lambda(q-1))} \tag{8.13}$$

Το $\gcd(\lambda(p-1), \lambda(q-1))$ προκύπτει από το γινόμενο του ελάχιστου βαθμού ως προς τους δύο αριθμούς, των κοινών πρώτων παραγόντων των $\lambda(p-1), \lambda(q-1)$.

Ο αναδρομικός ορισμός του $\lambda(n)$ τελικά καταλήγει σε γινόμενα όρων $\phi(p^e) = p^{e-1}\cdot(p-1) = O(p^e)$, οπότε στην ανάλυση σε πρώτους του $\lambda(n)$, οι περισσότεροι όροι υπάρχουν και στο n. Συνεπώς το $\gcd(\lambda(p-1), \lambda(q-1))$ αναμένεται να μεταβάλλεται περίπου όπως και το $\gcd(p-1, q-1)$.

Άρα για μεγάλη τιμή του $\gcd(p-1, q-1)$ αναμένεται μεγάλη τιμή του $\gcd(\lambda(p-1), \lambda(q-1))$ που θα οδηγήσει σε μικρή τιμή το $\lambda(\lambda(n))$. Επειδή $T \mid \lambda(\lambda(n))$, είναι $T \leq \lambda(\lambda(n))$, οπότε τελικά η περίοδος T φράσσεται άνω από μια μικρή τιμή και αυτό είναι ανεπιθύμητο.

8.2 Ερωτήμα β: SafeSafe πρώτοι

Οι "safe primes" είναι πρώτοι αριθμοί της μορφής $p = 2p' + 1$ όπου p' είναι επίσης πρώτος.

Οι "SafeSafe primes" είναι safe primes $p = 2p' + 1$ για τους οποίο ισχύει ότι και ο p' είναι επίσης safe prime $p' = 2p'' + 1$ με $p'' \equiv 1 \pmod 4$.

Μελετάμε την μέγιστη περίοδο της γεννήτριας Blum-Blum-Shub, δηλαδή το άνω φράγμα $\lambda(\lambda(n))$, όταν επιλεγούν πρώτοι p, q που είναι SafeSafe primes.

Είναι:

$$\begin{gathered}\lambda(n) = \operatorname{lcm}(p-1, q-1) = \operatorname{lcm}(2p'+1-1, 2q'+1-1) = \\ \operatorname{lcm}(2p', 2q') = 2p'q'\end{gathered} \tag{8.14}$$

Τότε:

$$\begin{gathered}\lambda(\lambda(n)) = \\ \lambda(2p'q') = \\ \operatorname{lcm}(\lambda(2), \lambda(p'), \lambda(q')) = \\ \operatorname{lcm}(1, \phi(p'), \phi(q')) = \\ \operatorname{lcm}(p'-1, q'-1) = \\ \operatorname{lcm}(2p''+1-1, 2q''+1-1) = \\ \operatorname{lcm}(2p'', 2q'') = \\ 2p''q''\end{gathered} \tag{8.15}$$

Άρα ένα άνω φράγμα της περιόδου T είναι το $\lambda(\lambda(n)) = 2p''q'' = \Theta(2\cdot\frac{pq}{4^2}) = \Theta(n)$.

Απομένει να δείξουμε ότι η τιμή αυτή δεν είναι απλά άνω φράγμα αλλά είναι και εφικτή, για κάποιο αρχικό s_0.

Έστω αριθμοί s_{0p} και s_{0q} γεννήτορες των κυκλικών ομάδων $\mathbb{Z}_p^*$ και $\mathbb{Z}_q^*$ αντίστοιχα. Από το Κινζεικό Θεώρημα Υπολοίπων ορίζεται μοναδικός αριθμός $s_0 \pmod{pq}$ ώστε $s_0 \equiv s_{0p} \pmod p$ και $s_0 \equiv s_{0q} \pmod q$.

Με βάση το Κινέζικο Θεώρημα Υπολοίπων, δηλαδή τον ισομορφισμό των $U(\mathbb{Z}_{pq})$ και $\mathbb{Z}_p^* \times \mathbb{Z}_q^*$ προκύπτει, επειδή το τελευταίο ευθύ γινόμενο, είναι ευθύ γινόμενο κυκλικών ομάδων ότι:

$$\operatorname{ord}_n(s_0) = \operatorname{ord}_{pq}(s_0) = \operatorname{lcm}\left(\operatorname{ord}_p(s_{0p}), \operatorname{ord}_q(s_{0q})\right) = \operatorname{lcm}(p-1, q-1) = 2p'q' (= \lambda(n)) \tag{8.16}$$

Όπως σχολιάστηκε αρκετά παραπάνω:

$$\operatorname{ord}_n(s_1) = \operatorname{ord}_n(s_0^2) = \frac{\operatorname{ord}_n(s_0)}{\gcd(\operatorname{ord}_n(s_0), 2)} = \frac{2p'q'}{\gcd(2p'q', 2)} = \frac{2p'q'}{2} = p'q' \left(= \frac{\lambda(n)}{2}\right) \tag{8.17}$$

Τότε:

$$T = \operatorname{ord}_{\operatorname{ord}_n(s_1)}(2) = \operatorname{ord}_{p'q'}(2) = \operatorname{lcm}\left(\operatorname{ord}_{p'}(2), \operatorname{ord}_{q'}(2)\right) \mid 2p''q'' (= \lambda(\lambda(n))) \tag{8.18}$$

Γενικά από το Θέωρημα Lagrange $\mathrm{ord}_{p'}(2) \mid \mathrm{ord}\left(\mathbb{Z}_{p'}^*\right) = p' - 1 = 2p''$.

Αν $\mathrm{ord}_{p'}(2) = 2$ τότε $2^2 = 4 \equiv 1 \pmod{p'}$ που είναι άτοπο διότι $p' > 4$.

Συνεπώς $\mathrm{ord}_{p'}(2) = p''$ ή $\mathrm{ord}_{p'}(2) = 2p''$. Όμοια ισχύει ως προς q'.

Άρα, αφού $p'', q'' \in \mathbb{P}$ για να ισχύει $T = \mathrm{lcm}\left(\mathrm{ord}_{p'}(2), \mathrm{ord}_{q'}(2)\right) = 2p''q''$ αρκεί και πρέπει $2 \mid \mathrm{ord}_{p'}(2)$ ή $2 \mid \mathrm{ord}_{q'}(2)$, ισοδύναμα $\mathrm{ord}_{p'}(2) \neq p'' = \frac{p-1}{2}$ ή $\mathrm{ord}_{p'}(2) \neq 2q'' = \frac{q-1}{2}$, ισοδύναμα $2^{\frac{p-1}{2}} \not\equiv 1 \pmod{p'}$ ή $2^{\frac{q-1}{2}} \not\equiv 1 \pmod{q'}$, ισοδύναμα από το Κριτήριο Euler το 2 να είναι τετραγωνικό υπόλοιπο το πολύ ως προς έναν από τους p', q'.

Από το σύμβολο Legendre το 2 είναι τετραγωνικό κατάλοιπο ως προς τον $p' \in \mathbb{P}$ ανν $p' \equiv 1 \pmod 8$ ή $p' \equiv 7 \pmod 8$, δηλαδή ανν τα 3 LSB δυαδικά ψηφία του p' είναι $0b001$ ή $0b111$. Επειδή $p'' = \frac{p'-1}{2}$ (δεξιό shift του p'), αρκεί και πρέπει τα λήγοντα ψηφία του p'' να είναι $0b00$ (που απορρίπτεται διότι $p'' \in \mathbb{P}$) ή $0b11$, δηλαδή $p'' \equiv 3 \pmod 4$. Όμως οι "SafeSafe" primes δίνουν $p'' \equiv 1 \pmod 4$, συνεπώς το 2 δεν είναι τετραγωνικό κατάλοιπο ως προς αυτούς.

(θα μπορούσαμε συνεπώς να χαλαρώσουμε τον περορισμό, και να βάλουμε μόνον έναν από τους p,q, να δώσει $p'' \equiv 1 \pmod 4$ αντί και τους δύο)

Συνεπώς, αν αρχικά διαλέξουμε έναν αριθμό που έχει υπόλοιπα ως προς p, q γεννήτορες των αντίστοιχων κυκλικών ομάδων $\mathbb{Z}_p^*$ και $\mathbb{Z}_q^*$ αντίστοιχα, προκύπτει πράγματι η μέγιστη περίοδος $T = \lambda(\lambda(n)) = 2p''q'' = \Theta(n)$.

Μάλιστα οι γεννήτορες αυτοί είναι αρκετοί. Πιο συγκεκριμένα η κυκλική ομάδα $\mathbb{Z}_p^*$ έχει πλήθος γεννητόρων $\phi(p-1) = \phi(2p') = 1' \cdot (p'-1) = p' - 1$.

Συνεπώς το ποσοστό αριθμών s_0 που θα δώσουν μέγιστη περίοδο, από όλους τους αριθμούς s_0 που είναι σχετικά πρώτοι με το n, και κατ'επέκταση η πιθανότητα πράγματι η περίοδος είναι να είναι μέγιστη, είναι (τουλάχιστον, διότι μπορεί να υπάρχουν και άλλες περιπτώσεις πέραν αυτής που εξετάσαμε που δίνουν την μέγιστη περίοδο):

$$\begin{aligned}\Pi\% = \frac{(p'-1)(q'-1)}{\phi(n)} = \frac{(p'-1)(q'-1)}{\phi(p)\phi(q)} = \frac{(p'-1)(q'-1)}{(p-1)(q-1)} = \frac{(p'-1)(q'-1)}{(2p'+1-1)(2q'+1-1)} = \\ \frac{(p'-1)(q'-1)}{4p'q'} \approx \frac{p'q'}{4p'q'} = \frac{1}{4} = 25\%\end{aligned} \tag{8.19}$$

Συνεπώς, αν οι p,q επιλεγούν να είναι "SafeSafe primes" είναι σχετικά πιθανό (πιθανότητα 25% αν το s_0 επιλεγεί ομοιόμορφα τυχαία) πράγματι η περίοδος τελικά να είναι η μέγιστη δυνατή $T = 2p''q'' \approx 2\frac{p}{4}\frac{q}{4} = \frac{n}{8}$.

9 Άσκηση 9: Υλοποίηση Γεννήτριας Blum-Blum-Shub με Μέγιστη Περίοδο

9.1 Εύρεση SafeSafe primes

Αρχικά, πρέπει να βρεθούν αριθμοί p,q, μεγέθους λ bits, που να είναι "SafeSafe primes" και $p \equiv q \equiv 3 \pmod 4$.

Υπενθυμίζεται ότι ένας αριθμός p είναι "SafeSafe prime" αν:

1. $p \in \mathbb{P}$
2. $p = 2p' + 1$, όπου $p' \in \mathbb{P}$
3. $p' = 2p'' + 1$, όπου $p'' \in \mathbb{P}$
4. $p'' \equiv 1 \pmod 4$

Ευτυχώς, φαίνεται ότι το κλάσμα των λ-bit αριθμών που είναι "SafeSafe primes" είναι $\frac{1}{\lambda^3 \ln^3 2}$. Έτσι με εκτιμώμενο πλήθος $\lambda^3 \ln^3 2$ δοκιμών ομοιόμορφα τυχαίων αριθμών θα βρούμε έναν SafeSafe prime.

Για να είναι $p \equiv 3 \pmod 4$ αρκεί και πρέπει τα 2 LSB bits του p να είναι 11.

Είναι $p'' = p >> 2$ (απορρίπτονται τα LSB 2 δυαδικά ψηφία). Συνεπώς για να είναι $p'' \equiv 1 \pmod 4$ αρκεί και πρέπει τα 4ο-3ο LSB bits να είναι $'01'$.

Επίσης, για να έχει λ bits ο p αρκεί και πρέπει το MSB bit στην λ-οστή θέση να είναι 1.

Έτσι για να βρούμε έναν "SafeSafe prime" p, με $p \equiv 3 \pmod 4$ επαναλαμβάνουμε μέχρι την επιτυχία τα εξής:

1. $r \xleftarrow{R} \{0,1\}^{\lambda-5}$
2. $p \leftarrow' 1' \,||\, r \,||' \, 0111'$
3. Αν δεν ισχύει $isPrime(p)$ (στην πράξη αν δεν ισχύει $MillerRabin(p)$) δηλώνουμε αποτυχία.
4. $p' \leftarrow \frac{p-1}{2}$ (θα πετύχει για $n > 2$ αφού ο μόνος άρτιος πρώτος είναι το 2)
5. Αν δεν ισχύει $isPrime(p')$ (στην πράξη αν δεν ισχύει $MillerRabin(p')$) δηλώνουμε αποτυχία.
6. $p'' \leftarrow \frac{p-1}{2}$ (θα πετύχει για $n > 3$ αφού ο μόνος άρτιος πρώτος είναι το 2)
7. Αν δεν ισχύει $isPrime(p'')$ (στην πράξη αν δεν ισχύει $MillerRabin(p'')$) δηλώνουμε αποτυχία.
8. Δηλώνουμε επιτυχία.

Αναμένονται $\Theta(n^3)$ εκτελέσεις όπου κάθε εκτέλεση απαιτεί $\Theta(kn^3)$ χρόνο (k ο αριθμός γύρων του $MillerRabin(n)$) οπότε θα απαιτηθεί χρόνος (kn^6) για την παραπάνω διαδικασία.

Το k επιλέγεται μικρή σταθερά καθώς το MillerRabin απαντά λάθος με 4^{-k} πιθανότητα, οπότε για $k = 30$ η πιθανότητα ψευδούς απάντησης είναι 1 στις 10^{18}.

Για $\lambda = 20$, $k = 30$ θα είναι $k\lambda^6 = 30 \cdot 20^6 \approx 2cdot10^9$ που θεωρείται λίγο μεγάλο, όμως υπολογιστικά ανεκτό. (προβλέπεται να πάρει μάλλον δευτερόλεπτα ή λίγα λεπτά)

Συμπλήρωση: Όταν θα γίνει δειγματοληψία για τον q πρέπει να διασφαλιστεί ότι $p \neq q$.

9.2 Εύρεση αρχικού αριθμού s_0

9.2.1 Αλγόριθμος εύρεσης ομοιόμορφα τυχαίου γεννήτορα κυκλικής ομάδας

9.2.1.1 Αλγόριθμος ελέγχου αν στοιχείο g κυκλικής ομάδας G είναι γεννήτορας

Στην γενική περίπτωση, έστω μια κυκλική ομάδα G τάξης $\text{ord}(G) = n$ όπου ο n έχει ανάλυση σε πρώτους παράγοντες $n = q_1^{e_1} \cdot q_2^{e_2} \cdot \cdots$. Θα βρούμε ένας αλγόριθμο $isGen(g)$ που ελέγχει αν ένα στοιχείο g είναι γεννήτορας της G.

Ένα στοιχείο g είναι γεννήτορας της G αν και μόνο αν $\text{ord}_G(g) = \text{ord}(G) = n$.

Από το Θεώρημα Lagrange $\text{ord}_G(g) \mid \text{ord}(G)$.

Έστω $\text{ord}(G) = c \cdot \text{ord}_G(g)$.

Διακρινούμε περιπτώσεις:

1. $c = 1$ και τότε $\text{ord}_G(g) = \text{ord}(G)$, οπότε το g είναι γεννήτορας.
2. $c \neq 1$, οπότε $\text{ord}_G(g) < \text{ord}(G)$ και τότε το g δεν είναι γεννήτορας.

   Έστω q ένας πρώτος παράγοντας του c. Το q είναι επίσης πρώτος παράγοντας και του $\text{ord}(G) = n$ (αφού $\text{ord}(G) = c \cdot \text{ord}_G(g)$).

   Συνεπώς $q = q_i$ για κάποιο πρώτο παράγοντα q_i του n.

   Είναι $\text{ord}(G) = c \cdot \text{ord}_G(g) = q \cdot c' \cdot \text{ord}_G(g)$.

   Τότε:

$$g^{\text{ord}_G(g)} = e \Rightarrow \left(g^{\text{ord}_G(g)}\right)^{c'} = e^{c'} \Rightarrow g^{c' \cdot \text{ord}_G(g)} = e \Rightarrow g^{\frac{n}{q_i}} = e \tag{9.1}$$

Συνεπώς αν το g δεν είναι γεννήτορας, υπάρχει ένας πρώτος παράγοντας q_i του $\text{ord}(G) = n$ ώστε $g^{\frac{n}{q_i}} = e$.

Αντίστροφα, αν για αριθμό $q_i > 1$ είναι $g^{\frac{n}{q_i}} = e$ τότε $\text{ord}_G(g) < n = \text{ord}(G)$ οπότε το g δεν είναι γεννήτορας.

Άρα για να εξετάσουμε αν ένα στοιχείο g είναι γεννήτορας βρίσκουμε όλους τους πρώτους παράγοντες q_i του $\text{ord}(G) = n$ και εξετάζουμε αν για κάποιον από αυτούς ισχύει $g^{\frac{n}{q_i}} = e$. Αν ισχύει για κάποιον τότε το g δεν ειναι γεννήτορας, αλλιώς είναι.

Δυστυχώς ο αλγόριθμος απαιτεί παραγοντοποίηση του $\text{ord}(G)$, ωστόσο στην δική μας περίπτωση την έχουμε, όπως θα δούμε.

9.2.1.2 Αλγόριθμος εύρεσης τυχαίου γεννήτορα

Υλοποιούμε την διαδικασία $Gen(G)$ για την εύρεση ομοιόμορφα τυχαίου γεννήτορα κυκλικής ομάδας G.

Έστω $n = \text{ord}(G)$ η τάξη της ομάδας.

Η ομάδα είναι κυκλική. Συνεπώς αν g είναι ένας γεννήτορας της, τότε όλα τα στοιχεία g^k με k, n σχετικά πρώτους είναι γεννήτορες, και αντίστροφα όλοι οι γεννήτορες είναι αυτής της μορφής.

Συνεπώς το πλήθος γεννητόρων της $\mathbb{Z}_p^*$ είναι $\phi(n) = O(n)$.

Υποθέτουμε ότι ξέρουμε όλους τους πρώτους παράγοντες q_i του n (εδώ δεν απαιτούνται οι εκθετές. ωστόσο το πρόβλημα δεν είναι ευκολότερο, διότι μπορούμε να βρούμε τον εκθέτη σε πολυωνυμικό χρόνο με απλή γραμμική αναζήτηση – ή και με δυαδική αναζήτηση).

Εκτελούμε την εξής διαδικασία μέχρι να επιτυχεί:

1. $g \xleftarrow{R} G$
2. Αν όχι $isGen(g)$ τότε δήλωσε αποτυχία (και άρα ξαναπροσπάθησε).

Είναι $\text{ord}(G) = n = O(n)$ και επιτυγχάνουν $O(n)$ στοιχεία της G. Συνεπώς απαιτείται $O(1)$ αριθμός δοκιμών μέχρι την επιτυχία, δηλαδή την εύρεση ενός γεννήτορα g.

9.2.1.3 Εξειδίκευση αλγόριθμου για ομάδα $\mathbb{Z}_p^*$ όπου p Sophie-Germain πρώτος

Οι πρώτοι αριθμοί Sophie-Germain είναι πρώτοι αριθμοί p της μορφής $p = 2p' + 1$ (όλοι οι πρώτοι εκτός του 2 έχουν αυτή την μορφή) όπου p' πρώτος.

Έστω p Sophie-Germain πρώτος, οπότε $p = 2p' + 1$, όπου p' πρώτος.

Είναι $\text{ord}\left(\mathbb{Z}_p^*\right) = p - 1 = 2p'$.

Συνεπώς οι πρώτοι παράγοντες του $\text{ord}\left(\mathbb{Z}_p^*\right)$ είναι ακριβώς οι 2 και $p' = \frac{p-1}{2}$.

Συνεπώς στην διαδικασία $isGen(g)$ θα ελέγξουμε αν $g^{\frac{2p'}{2}} = g^{p'} \equiv 1 \pmod p$ ή αν $g^{\frac{2p'}{p'}} = g^2 \equiv 1 \pmod p$.

Μάλιστα, επειδή p πρώτος το 1 έχει δύο τετραγωνικές ρίζες $\mod p$, το 1 και το -1. Συνεπώς αντί να ελέγξουμε αν $g^2 \equiv 1 \pmod p$ μπορούμε ισοδύναμα να ελέγξουμε αν $g \equiv 1 \pmod p$ ή $g \equiv -1 \pmod p$, και επειδή συνήθως η αριθμητική στον H/Υ γίνεται στο διάστημα $[0, p-1]$ μπορούμε τότε ισοδύναμα να ελέγξουμε αν $g = 1$ ή $g = p - 1$.

Αν τύχει $g = 1$ ή $g = p - 1$ θα απορρίψουμε αυτον τον αριθμό, οπότε μπορούμε εξαρχής να τους εξαιρέσουμε από το σύνολο που δειγματοληπτούμε.

Συνεπώς μπορούμε να απλοποιήσουμε την διαδικασία εύρεσης γεννήτορας ως εξής:

Εκτελούμε την εξής διαδικασία μέχρι να επιτυχεί:

1. $g \xleftarrow{R} [2, p-2]$
2. Αν $g^{\frac{p-1}{2}} \equiv 1 \pmod{p}$ τότε δήλωσε αποτυχία (και άρα ξαναπροσπάθησε).

Όπως δείξαμε παραπάνω απαιτούνται $O(1)$ επαναλήψεις.

Ο υπολογισμός του $g^{\frac{p-1}{2}}$ μπορεί να γίνει με επαναλαμβανόμενο τετραγωνισμό σε $O(\lambda^3)$.

Συνεπώς ο αλγόριθμος απαιτεί χρόνο $O(\lambda^3)$.

9.2.2 Αλγόριθμος επίλυσης γραμμικού συστήματος ισοτιμιών

Έστω ένα σύστημα k γραμμικών ισοτιμιών $x \equiv a_i \pmod{n_i}$ όπου n_i σχετικά πρώτοι μεταξύ τους.

Το Κινέζικο Θεώρημα Υπολοίπων μας δίνει ότι υπάρχει μοναδική λύση $\mod \prod_{i=1}^{k} n_i$.

Θα βρούμε την λύση αυτή. Πρακτικά ακολουθούμε την απόδειξη του Κινέζικου Θεωρήματος Υπολοίπων.

Έστω $N = \prod_{i=1}^{k} n_i$. Σκοπός είναι η εύρεση του $x \mod N$.

Έστω $N_i = \frac{N}{n_i}$. Αφού οι αριθμοί n_i είναι σχετικά πρώτοι μεταξύ τους, τότε οι N και n_i είναι σχετικά πρώτοι, οπότε $\gcd(N_i, n_i) = 1$. Συνεπώς ορίζεται ο πολλαπλασιαστικός αντίστροφος $N_i^{-1} \pmod{n_i}$.

Υπολογίζουμε το $N_i^{-1} \pmod{n_i}$ με τον Εκτεταμένο Αλγόριθμο Ευκλείδη.

Πιο συγκεκριμένα εκτελούμε τον Εκτεταμένο Αλγόριθμο Ευκλειδή για τους αριθμούς N_i, n_i (θα περιγραφεί σε επόμενη παράγραφο) και μας επιστρέφει αριθμούς $M_i, m_i, \gcd(N_i, n_i)$ για τους οποιούς ισχύει η ταυτότητα Bézout:

$$M_i N_i + m_i n_i = \gcd(N_i, n_i) = 1 \Rightarrow M_i N_i = 1 - m_i n_i \Rightarrow M_i N_i \equiv 1 \pmod{n_i} \Rightarrow M_i \equiv N_i^{-1} \pmod{n_i} \quad (9.2)$$

Άρα το $N_i^{-1} \pmod{n_i}$, είναι το M_i που επιστρέφει ο Εκτεταμένος Αλγόριθμος Ευκλείδη.

Τελικά υπολογίζουμε:

$$x = \sum_{i=1}^{k} a_i \mathrm{M}_i N_i \quad (9.3)$$

Είναι:

$$a_i \mathrm{M}_i N_i \equiv a_i N_i^{-1} N_i \equiv a_i \pmod{n_i} \quad (9.4)$$

Για $j \neq i$ είναι $N_j = \frac{n}{n_j} = \frac{n_1 \cdot n_2 \cdot \ldots \cdot n_k}{n_j}$, οπότε $n_i \mid N_j$, και άρα:

$$a_j M_j N_j \equiv 0 \pmod{n_i} \quad (9.5)$$

Τελικά:

$$x = \sum_{i=1}^{k} a_i \mathrm{M}_i N_i \equiv a_i \pmod{n_i} \quad (9.6)$$

Συνεπώς το x που βρήκαμε είναι λύση του συστήματος.

Συνοψίζοντας τα υπολογιστικά βήματα είναι:

1. $N \leftarrow \prod_{i=1}^{k} n_i$
2. Για $i = 1 \ldots k$:
   (α′) $N_i \leftarrow \frac{N}{n_i}$
   (β′) Εκτέλεσε τον Εκτεταμένο Αλγόριθμο Ευκλείδη για τους αριθμούς N_i, n_i. Θα επιστραφούν 3 αριθμοί M_i, m_i, $d = \gcd(N_i, n_i)$. Κράτα μόνο το M_i.
3. $x \leftarrow \sum_{i=1}^{k} a_i \mathrm{M}_i N_i$

Ο Εκτεταμένος αλγόριθμος Ευκλείδη απαιτεί χρόνο $\mathrm{O}(\lambda^2)$ (όχι $O(\lambda^3)$ καθώς προχωρά η εκτέλεση μικραίνουν οι αριθμοί)

Συνεπώς απαιτείται χρόνος $O(k\lambda^2)$.

9.2.3 Κύριος αλγόριθμος εύρεσης

Όπως περιγράψαμε στην παράγραφο 8.2, θέλουμε να βρούμε αριθμό s_0 τέτοιο ώστε οι $s_0 \mod p$ και $s_0 \mod q$ να είναι γεννήτορες των ομάδων $\mathbb{Z}^p$ και $\mathbb{Z}^q$ αντίστοιχα.

Όπως είχαμε δείξει, περίπου 25% των αριθμών που είναι σχετικά πρώτοι με το n ικανοποιούν αυτή την ιδιότητα. Οι σχετικά πρώτοι αριθμοί με τον n, που να είναι $< n$, είναι $\phi(n) = (p-1)(q-1) \approx \Theta(n)$, οπότε με $O(1)$ ομοιόμορφα τυχαία επιλογές για το s_0 προβλέπεται να βρούμε αριθμό s_0 που να ικανοποιεί αυτά που θέλουμε.

Θα το κάνουμε **πιο αποδοτικά**.

Κάθε αριθμός s_0 που είναι σχετικά πρώτος με τον $n = pq$ από το Κινέζικο Θεώρημα Υπολοίπων (αφού p,q σχετικά πρώτοι, αφού είναι πρώτοι) είναι μοναδική λύση του συστήματος (στο διάστημα $0 \leq s_0 < n$):

$$\begin{aligned} s_0 &\equiv s_{0p} \pmod{p} \\ s_0 &\equiv s_{0q} \pmod{q} \end{aligned} \tag{9.7}$$

όπου $s_{0p} = s_0 \pmod{p}$ και $s_{0q} = s_0 \pmod{q}$.

Συνεπώς χρησιμοποιώ τον αλγόριθμο που περιγράφηκε παραπάνω (τον συμβολίζω $Gen(p)$) για να βρω έναν ομοιόμορφα τυχαίο γεννήτορα της ομάδας $\mathbb{Z}_p^*$. ($p = 2p' + 1$ όπου p' πρώτος, οπότε ισχύει η βελτιωμένη εκδοχή).

Τότε βρίσκω τον ομοιόμορφα τυχαία αριθμό s_0 που ικανοποιεί τα ζητούμενα ως λύση του συστήματος:

$$\begin{aligned} s_0 &\equiv Gen(p) \pmod{p} \\ s_0 &\equiv Gen(q) \pmod{q} \end{aligned} \tag{9.8}$$

Οι υπολογισμοί $Gen(p)$, $Gen(q)$ απαιτούν χρόνο $O(\lambda^3)$ και η επίλυση του συστήματος απαιτεί χρόνο $O(\lambda^2)$.

Συνεπώς συνολικά για την εύρεση του ομοιόμορφα τυχαίου s_0 απαιτείται χρόνος $O(\lambda^3)$.

9.3 Προγραμματιστική Υλοποίηση σε Haskell και πειραματική επαλήθευση περιόδου

Οι παραπάνω αλγόριθμοι υλοποίηθηκαν σε Haskell (pure lazy functional language).

Ο πηγαίος κώδικας παρατίθεται στο παράρτημα A′ και έχει αποσταλλεί μαζί με την παρούσα αναφορά σε χωριστό φάκελο. Η μεταγλώττιση γίνεται μέσω του εργαλείου `cabal`.

Ο έλεγχος πρώτων αριθμών Miller-Rabin είχε υλοποιηθεί σε προηγούμενη σειρά ασκήσεων.

Εχουν γίνει τυχαιοποιημένοι property-based έλεγχοι των επιμέρους συναρτήσεων με χρήση του QuickCheck.

Όπου απαιτείται τυχαιότητα στο αρχικό setup της Blum-Blum-Shub (ενδεικτικά για την αναζήτηση πρώτων αριθμών, για την εύρεση γεννητόρων, κ.λπ.) έχει χρησιμοποιηθεί ένας Monad που κρύβει την υλοποίηση της γεννήτριας τυχαιότητας, ώστε ο τελικός χρήστης να μπορεί να χρησιμοποιήσει την γεννήτρια της επιλογής του.

Τελικά, η περίοδος ελέγχθηκε σε τυχαιοποιημένους ελέγχους (δοκιμές με setups της Blum-Blum-Shub) πως είναι πράγματι η μέγιστη και ίση με $2p''q''$ όπως προβλέφθηκε θεωρητικά.

Ο σχετικός έλεγχος έχει ενσωματωθεί στην γενικότερη σουίτα τυχαιοποιημένων ελέγχων (tests) που έχουν συνταχθεί, αφού είναι γενικά και αυτός ένας έλεγχος "ορθότητας" της γεννήτριας.

10 Άσκηση 10: Συναρτήσεις Σύνοψης

10.1 Ερώτημα α: Ομομορφικής ως προς XOR συνάρτηση σύνοψης και αντίσταση συγκρούσεων

Έστω μια συνάρτηση σύνοψης $H : \{0,1\}^* \mapsto \{0,1\}^n$ τέτοια ώστε για οποιαδήποτε a,b:

$$H(a \oplus b) = H(a) \oplus H(b) \tag{10.1}$$

Θα δείξουμε ότι η H δεν προσφέρει αντίσταση συγκρούσεων. Μάλιστα θα δείξουμε το ισχυρότερο, πως η H δεν προσφέρει αντίσταση δεύτερου ορίσματος. Περιγράφουμε έναν αλγόριθμο που βρίσκει για κάθε x ένα $x' \neq x$ ώστε $H(x) = H(x')$.

Έστω ότι είχαμε ένα στοιχείο $k \neq 0$ ώστε $H(k) = 0$.

Τότε $\mathrm{H}(x \oplus k) = H(x) \oplus H(k) = H(x) \oplus 0 = H(x)$. Αφού $k \neq 0$, τότε $x \oplus k \neq x$, οπότε το $x' = x \oplus k$ είναι μια αποδεκτή απάντηση.

Άρα αρκεί να βρούμε ένα $k \neq 0$, ώστε $H(k) = 0$.

Περιοριζόμαστε σε ορίσματα μεγέθους $n+1$ bits ($\subset \{0,1\}^*$). Από την αρχή της περιστεροφωλιάς, υπάρχει μια σύγκρουση (x, x'). Τότε

$$H(x) = H(x') \iff H(x) \oplus H(x') = 0 \iff H(x \oplus x') = 0 \tag{10.2}$$

Αφού $x \neq x'$. τότε $x \oplus x' \neq 0$. Άρα, για αρχή, το αναζητούμενο $k \neq 0$ ώστε $H(k) = 0$ γνωρίζουμε ότι υπάρχει.

Γενικά: $H(0) \overset{\forall a}{=} H(a \oplus a) = H(a) \oplus H(a) = 0$.

Σημείωση: *Η τελευταία σχέση θα μπορούσε φαινομενικά να σχολιαστεί ότι σημαίνει πως δεν προσφέρει αντίσταση πρώτου ορίσματος, ωστόσο η αντίσταση πρώτου ορίσματος αφορά σε δυσκολία αντιστροφής για $y = H(x)$ όπου το x επιλέγεται ομοιόμορφα τυχαία, οπότε δεν προκύπτει αυτό το συμπέρασμα.*

Συνεπώς αν βρούμε δύο διαφορετικές λύσεις της $H(x) = 0$ θα ξέρουμε ότι η μία είναι μη μηδενική.

Το σώμα GF(2) ορίζεται από το σύνολο $\{0,1\}$, την πράξη `xor` ($\oplus$) ως πρόσθεση και την πράξη `and` ($\otimes$) ως πολλαπλασιασμό (είναι οι συνηθείς πράξεις πρόσθεσης και πολλαπλασιασμού όμως $\mod 2$). Θα καταστρώσουμε ένα σύστημα γραμμικών εξίσωσεων στο GF(2), που μπορεί να λυθεί αποδοτικά με χρήση απαλοιφής Gauss.

Γενικά έστω ένας αριθμός x $(n+1)$-bits. Είναι:

$$x = x_n x_{n-1} \dots x_0 = \left(x_n \otimes (1 \parallel 0^{n-1})\right) \oplus \left(x_{n-1} \otimes (01 \parallel 0^{n-2})\right) \oplus \cdots \oplus \left(x_0 \otimes (0^{n-1} \parallel 1)\right) \tag{10.3}$$

Σημειώνεται ότι $H(0 \otimes x) = H(0) = 0 = 0 \otimes H(x)$ και $H(1 \otimes x) = H(x) = 1 \otimes H(x)$.

Συνεπώς $H(x) = x_n \otimes H(1 \parallel 0^{n-1}) \oplus x_{n-1} \otimes H(01 \parallel\parallel 0^{n-2}) \oplus x_0 \otimes H(0^{n-1} \parallel 1)$.

Με άλλα λόγια η H είναι μια γραμμική απεικόνιση $\{0,1\}^{n+1} \mapsto \{0,1\}^n$.

Οι λύσεις x, της $H(x) = 0$ είναι ο πυρήνας $\ker H$ της H.

Είναι: $\dim \ker H = n + 1 - \dim \operatorname{im} H \geq n + 1 - n = 1$

Εξάλλου, γνωρίζουμε, όπως εξηγήθηκε παραπάνω, ότι ο $\ker H$ έχει τουλάχιστον 2 στοιχεία, οπότε αποκλείεται $\dim \ker H = 0$.

Ο πίνακας της γραμμικής απεικόνισης H είναι:

$$H = [H(10^{n-1}) \mid H(01^{n-2}) \mid \cdots \mid H(0^{n-1}1)] \tag{10.4}$$

Εκτελούμε τον αλγόριθμο Gauss που μας επιστρέφει μια βάση του $\ker H$ (σε χρόνο $O(n^3)$).

(εδώ https://en.wikipedia.org/wiki/Kernel_(linear_algebra)#Computation_by_Gaussian_elimination υπάρχουν λεπτομέρειες για τον τρόπο χρήσης του αλγόριθμου Gauss ώστε να βρίσκει μια βάση του πυρήνα μιας γραμμικής απεικόνισης.)

Επιλέγουμε ένα διάνυσμα k της βάσης (ή το μοναδικό αν είναι $\dim \ker H = 1$), αυτό είναι $k \neq 0$ αφού είναι διάνυσμα βάσης, και ισχύει $H(k) = 0$.

Άρα βρήκαμε ένα $k \neq 0$, ώστε $H(k) = 0$. Άρα εφαρμόζουμε την αρχική ιδέα και επιστρέφουμε $x' = x \oplus k$, ώστε $H(x') = H(x \oplus k) = H(x) \oplus H(k) = H(x) \oplus 0 = H(x)$.

Ο $\mathcal{A}$ είναι πολυωνυμικός. Συνεπώς η H δεν προσφέρει αντίσταση δευτέρου ορίσματος. (συνεπώς δεν προσφέρει και αντίσταση συγκρούσεων, αφού π.χ. για είσοδο 1^{n+1} εντοπίζεται μια σύγκρουση $x' \neq 1^{n+1}$.)

10.2 Ερώτημα β: Παράθεση συναρτήσεων σύνοψης

Έστω H μια συνάρτηση σύνοψης $H(x) = H_1(x) \mid\mid H_2(x) \mid\mid H_3(x)$ όπου τουλάχιστον μία εκ των H_1, H_2, H_3 διαθέτει δυσκολία εύρεσης συγκρούσεων. Θα δείξουμε ότι και η H διαθέτει δυσκολία εύρεσης συγκρούσεων.

Έστω μια σύγκρουση (x, x') για την H $(H(x) = H(x'))$.

Τότε:

$$\begin{gathered} H(x) = H(x') \Rightarrow \\ H_1(x) \mid\mid H_2(x) \mid\mid H_3(x) = H_1(x') \mid\mid H_2(x') \mid\mid H_3(x') \Rightarrow \\ H_1(x) = H_1(x') \wedge H_2(x) = H_2(x') \wedge H_3(x) = H_3(x') \end{gathered} \tag{10.5}$$

Συνεπώς διαιρώντας τα $H(x)$, $H(x')$ σε 3 ίσα μέρη, λαμβάνουμε (σε πολυωνυμικό χρόνο) μια σύγκρουση και για τις 3 συναρτήσεις H_1, H_2, H_3, συνεπώς και για εκείνη εκ των τριών που έχει υπολογιστική δυσκολία εύρεσης συγκρούσεων. Άρα και η H έχει υπολογιστική δυσκολία εύρεσης συγκρούεσων.

11 Άσκηση 11: Ασφάλεια Merkle tree

Έστω μια συνάρτηση σύνοψης $H_1 : \{0,1\}^{2n} \mapsto \{0,1\}^n$. Η συνάρτηση αυτή χρησιμοποιείται για την κατασκευή δένδρου Merkle ύψους h με είσοδο μια ακολουθία $x_0, x_1, \cdots, x_{2h}$ όπου κάθε x_i είναι μια δυαδική ακολουθία μεγέθους n bits ($2h$ συμβολοσειρές ώστε το δέντρο να είναι πλήρες).

Λαμβάνοντας την τιμή της ρίζας, το δένδρο Merkle μπορεί να θεωρηθεί καθ' αυτό ως μια συνάρτηση σύνοψης H που συμπιέζει συμβολοσειρές μεγέθους $n \cdot 2h$ σε συμβολοσειρές μεγέθους n.

Θα δείξουμε ότι αν η H_1 διαθέτει δυσκολία εύρεσης συγκρούσεων τότε και η H διαθέτει δυσκολία εύρεσης συγκρούσεων.

Έστω μια σύγκρουση $(x, x') \in (\{0,1\}^n)^{n2^h}$ της συνάρτησης h. Επειδή $x \neq x'$, οι δύο ακολουθίες διαφέρουν τουλάχιστον σε ένα στοιχείο τους, έστω στο i-οστό, δηλαδή $x_i \neq x'_i$.

Ανεβαίνοντας τα δένδρα Merkle των x, x' από τα φύλλα x_i και x'_i προς την ρίζα (ένα επίπεδο την φορά) κάποια στιγμή φτάνουμε σε κόμβο που και στα δύο δέντρα έχει ίδια τιμή. Είναι σίγουρο ότι φτάνουμε σε τέτοιο κόμβο διότι η ρίζα των δύο δέντρων έχει ίδια τιμή. Έστω ο πρώτος κόμβος που θα βρούμε με την ίδια τιμή. Επειδή είναι ο πρώτος κόμβος σημαίνει ότι το παιδί του που εξετάσαμε προηγουμένως έχει διαφορετική τιμή στα δύο δέντρα (αυτό ειναι δεδομένο ότι συμβαίνει διότι ξεκινάμε από φύλλα x_i, x'_i με διαφορετική τιμή). Συνεπώς έχουμε βρεί έναν κόμβο, εφεξής τον ονομάζουμε p, που έχει ίδια τιμή στα δύο δέντρα, όμως έχει ένα αριστερό (ή δεξί) παιδί που στα δύο δέντρα έχει διαφορετική τιμή. Όμως η τιμή του p, προκύπτει από εφαρμογή της H_1 στην παράθεση των τιμών των δύο παιδιών του.

Συνεπώς έχουμε βρει δύο διαφορετικά όρισματα, την παράθεση των τιμών των δύο παιδιών του p χωριστά στα δύο δέντρα Merkle των x,x', τέτοια ώστε η εφαρμογή της H_1 σε αυτά δίνει ίδια τιμή (την τιμή του p).

Συνεπώς σε πολυωνυμικό χρόνο, από μια σύγκρουση της H βρήκαμε μια σύγκρουση της H_1. Συνεπώς αν η H_1 διαθέτει υπολογιστική δυσκολία εύρεσης συγκρούσεων, τότε και H διαθέτει υπολογιστική δυσκολία εύρεσης συγκρούσεων.

12 Άσκηση 12: CPA και CCA Ασφάλεια

12.1 Ύπαρξη αλγόριθμου εξαγωγής κλειδιού απο κρυπτοκείμενο

Δίνεται ένα κρυπτοσύστημα $\mathcal{CS}$ και αντίπαλος $\mathcal{A}$ που μπορεί να ανακτήσει το κλειδί από κρυπτοκείμενο του $\mathcal{CS}$ με μη-αμελητέα πιθανότητα.

Θα δείξουμε ότι το $\mathcal{CS}$ δεν παρέχει ασφάλεια CPA.

Θα κατασκευάσουμε έναν αντίπαλο $\mathcal{A}'$ που κερδίζει το πείραμα μη διαχωρισιμότητας CPA (*CPA indistinguishability experiment*) $PrivK^{cpa}_{\mathcal{A}',\mathcal{CS}}(n)$ με πιθανότητα $\frac{1}{2} + f(n)$, όπου $f(n)$ μη-αμελητέα, οπότε το $\mathcal{CS}$ δεν προσφέρει CPA ασφάλεια.

Ο αντίπαλος $\mathcal{A}'$ είναι ο εξής:

1. Επιλέγει δύο ομοιόμορφα τυχαία μηνύματα m_0, m_1 (η τυχαιότητα έχει σημασία σε ένα corner case που μάλλον δεν είναι πιθανό να συμβαίνει, ωστόσο την επιβάλλουμε) και τα στέλνει.
2. Του επιστρέφεται ένα κρυπτοκείμενο c. Εκτελεί τον αλγόριθμο $\mathcal{A}$. Με μη αμελητέα πιθανότητα, έστω ίση με $p \geq \frac{1}{q(n)}$, όπου $q(n)$ πολυώνυμο, του επιστρέφεται το κλειδί κρυπτογράφησης k που επιλέχθηκε στο πρώτο βήμα του $PrivK^{cpa}_{\mathcal{A}',\mathcal{CS}}(n)$ (γέννεση κλειδιού).

   Έπειτα υπολογίζει τα $Enc_k(m_0)$ και $Enc_k(m_1)$. Αν ένα από τα δύο μηνύματα είναι ίσο με c, επιστρέφει 0 ή 1 αντίστοιχα με το αν το m_0 ή το m_1 έδωσε το c. Αν δεν προκύψει το c ίσο με κανένα από τα δύο (ή αν ο $\mathcal{A}$ αποτύχει gracefully, δηλώνοντας ότι απέτυχε αντί να δώσει μια λάθος απάντηση), τότε επιστρέφεται μια arbitrary τιμή, ας πούμε 0.

Αν ο $\mathcal{A}$ πετύχει, οπότε επιστρέψει το πραγματικό κλειδί κρυπτογράφησης που χρησιμοποιήθηκε για την κρυπτογράφηση του c, τότε ο $\mathcal{A}'$ επιτυγχάνει πάντα.

Έστω ότι ο $\mathcal{A}$ αποτύχει. Αν απέτυχε gracefully ή έδωσε ένα κλειδί που δίνει $Enc_k(m_0) \neq c$ και $Enc_k(m_1) \neq c$, τότε επιστρέφεται 0, οπότε η πιθανότητα να είναι ίσο με το ομοιμόρφα τυχαίο $b \xleftarrow{R} \{0,1\}$ είναι $\frac{1}{2}$.

Έστω ότι αυτό δεν συνέβη, δηλαδή ο $\mathcal{A}$ απέτυχε, όμως μας έδωσε ένα κλειδί $k' \neq k$ που δίνει $Enc_{k'}(m_0) = c$ ή $Enc_{k'} m_1 = c$. Ονομάζουμε αυτό το ενδεχομένο "κακή Βυζαντινή αποτυχία" (πέραν του διαισθητικού επιπέδου, δεν υπάρχει καμία μαθηματική συσχέτιση με τα Βυζαντινά Σφάλματα των Κατανεμημένων Συστημάτων). Λόγω της ορθότητας της Enc, αποκλείεται να ισχύουν και οι δύο, διότι τότε δύο κείμενα θα δίναν ίδιο κρυπτοκείμενο, οπότε δεν θα μπορούσε να γίνει σωστά αποκρυπτογράφηση. Αν ο $\mathcal{A}$ αποτύχει με τρόπο που εν τέλει δίνει την σωστή απάντηση b, αυτό δεν αφορά στην παρούσα περίπτωση, και θεώρουμε μια τέτοια εκτέλεση του $\mathcal{A}$ επιτυχήμενη. Συνεπώς θεωρούμε ως "κακή Βυζαντινή αποτυχία" να λάβουμε ένα κλειδί k' ώστε να ισχύει $Enc_{k'}(m_{\neg b}) = c$. Η πιθανότητα να συμβεί αυτό είναι αμελητέα, διότι ο $\mathcal{A}$ έχει εισόδους που είναι στατιστικά ανεξάρτητες του m_{negb} (το c προκύπτει από κρυπτογράφηση του m_b). Σημειώνεται ότι αυτή η περίπτωση είναι που μας αναγκάζει να βάλουμε τυχαία μηνύματα m_0, m_1. Διαφορετικά θα μπορούσαμε arbitrary μηνύματα (π.χ. 0^n και 1^n).

Άρα η πιθανότητα "κακής Βυζαντινής αποτυχίας" είναι αμελητέα.

Η πιθανότητα επιτυχίας είναι:

$$\begin{aligned}
&\mathbb{P}\left[PrivK^{cpa}_{\mathcal{A}',\mathcal{CS}}(n)\right] \\
&\geq \mathbb{P}\left[PrivK^{cpa}_{\mathcal{A}',\mathcal{CS}}(n) \mid \mathcal{A} \text{ πετυχαίνει}\right] \cdot \mathbb{P}\left[\mathcal{A} \text{ πετυχαίνει}\right] \\
&+\mathbb{P}\left[PrivK^{cpa}_{\mathcal{A}',\mathcal{CS}}(n) \mid \mathcal{A} \text{ αποτυγχάνει χωρίς "κακή Βυζαντινή Αποτυχία"}\right] \\
&\cdot\mathbb{P}\left[\mathcal{A} \text{ αποτυγχάνει χωρίς "κακή Βυζαντινή Αποτυχία"}\right] \\
&+\mathbb{P}\left[PrivK^{cpa}_{\mathcal{A}',\mathcal{CS}}(n) \mid \mathcal{A} \text{ αποτυγχάνει με "κακή Βυζαντινή Αποτυχία"}\right] \\
&\cdot\mathbb{P}\left[\mathcal{A} \text{ αποτυγχάνει χωρίς "κακή Βυζαντινή Αποτυχία"}\right] \\
&= 1 \cdot p + \frac{1}{2} \cdot (1 - negl(n)) + p' \cdot negl(n) \overset{0 \leq p' \leq 1}{=} p + \frac{1}{2} + negl(n) \geq \frac{1}{q(n)} + \frac{1}{2} + negl(n)
\end{aligned} \tag{12.1}$$

Άρα το $\mathcal{CS}$ δεν προσφέρει CPA ασφάλεια.

12.2 Χρήση τροποποιημένου CBC Block Cipher Mode Of Operation

Έστω ένα σύστημα κρυπτογράφησης με μια συνάρτηση κρυπτογράφησης $Enc_k(x)$.

Από γνωστό Θεώρημα, αν $Enc_k(x)$ είναι ψευδοτυχαία μετάθεση, τότε το κλασικό Cipher Block Chaining (CBC) Mode of Operation προσφέρει CPA ασφάλεια.

Τροποποιούμε το CBC ώστε στην κρυπτογράφηση μιας ακολουθίας μηνύματων m_1, m_2, ...να χρησιμοποιηθεί διαδοχικά αυξανόμενα IV. Πιο συγκεκριμένα στην κρυπτογράφηση του i-οστού μηνύματος να χρησιμοποιεί για IV το $IV + i$.

Θα δείξουμε ότι αυτό το τροποποιημένο σχήμα δεν προσφέρει CPA ασφάλεια.

Θα κατασκευάσουμε έναν αντίπαλο $\mathcal{A}$ που κερδίζει το $PrivK^{cpa}_{\mathcal{A},CBC'}(n)$ με με πιθανότητα $\frac{1}{2} + f(n)$, όπου $f(n)$ μη-αμελητέα, οπότε το σύστημα δεν προσφέρει CPA ασφάλεια.

Σε διαισθητικό επίπεδο αξιοποιούμε το γεγονός ότι αν ένας δυαδικός αριθμός x, n-bits, λήγει σε 0 τότε $x + 1 = x \oplus 0^{n-1}1$, ενώ αν λήγει σε 1 τότε ο $x + 1$ λήγει σε 0.

Ο αλγόριθμος $\mathcal{A}$ είναι ο εξής:

Δημιούργησε δύο arbitrary μηνύματα, arbitrary μεγέθους m_0, m_1 (π.χ. 0^n, 1^n). Πριν τα στείλεις, θα στείλουμε μερικά μηνύματα στο μαντείο, ώστε να εξασφαλίσουμε ότι όταν υπολογιστεί το $c = Enc_k(m_b)$ το IV_i θα λήγει σε 0. (το παρόν σύστημα κρυπτογράφησης, οπότε και το μαντείο, σημειώνεται ότι είναι εξ ορισμού stateful).

Ρώτα το μαντείο ένα arbitrary μήνυμα x (π.χ. $x = 0^n$), οπότε θα επιστραφεί ένα y. Στην συνέχεια, ρώτα το $x \oplus 0^{n-1}1$, οπότε θα επιστραφεί ένα y'.

Συμβολίζω με δείκτη $_0$ το πρώτο block κειμένου/κρυπτοκειμένου.

Είναι $y_0 = Enc_k(IV \oplus x_0)$ και $y'_0 = Enc_k((IV + 1) \oplus x_0 \oplus 0^{n-1}1)$.

Αν το αρχικό IV έληγε σε 0, τότε $y'_0 Enc_k(IV \oplus 0^{n-1}1 \oplus x_0 \oplus 0^{n-1}1) = Enc_k(IV \oplus x_0) = y_0$.

Αν το αρχικό IV δεν έληγε σε 0 τότε $(IV + 1) \oplus x_0 \oplus 0^{n-1}1$ και $IV \oplus x_0$ είναι διαφορετικά, οπότε αφού η Enc είναι ψευδοτυχαία, τότε τα y_0, y'_0 είναι ίσα με αμελητέα πιθανότητα.

Συνεπώς γνωρίζω με πιθανότητα $1 - negl(n)$ πως αν $y_0 = y'_0$ τότε και μόνο τότε το IV λήγει σε 0.

Στην κρυπτογράφηση του m_b θα χρησιμοποιηθεί το $IV + 2$ αν δεν στείλω τίποτα άλλο και το $IV + 3$ αν στείλω ακόμα ένα μήνυμα.

Αν το IV λήγει σε 0 τότε και το $IV + 2$ λήγει σε 0, ενώ αν το IV λήγει σε 1 τότε το $IV + 3$ λήγει σε 0.

Συνεπώς, εξετάζω αν $y_0 = y'_0$ και αν δεν ισχύει, στέλνω στο μαντείο κρυπτοφράφησης ένα ακόμα arbitrary μήνυμα (π.χ. το 1^n).

Πλέον με πιθανότητα $1 - negl(n)$ γνωρίζω ότι το m_b θα κρυπτογραφηθεί με ένα IV' που λήγει σε 0.

Στέλνω τώρα τα m_0, m_1 πίσω.

Επιστρέφεται ένα κρυπτοκείμενο c.

Ρωτώ το μαντείο κρυπτογράφησης το μήνυμα $m_0 \oplus 0^{n-1}1$ και απαντά ένα c'.

Είναι $c' = Enc_k((IV' + 1) \oplus m_0 \oplus 0^{n-1}1) = Enc_k(IV' \oplus 0^{n-1}1 \oplus m_0 \oplus 0^{n-1}1) = Enc_k(IV' \oplus m_0)$.

Συνεπώς είναι $c = c'$ αν και μόνο αν $m_b = m_0$. (αποκλείεται και το $IV' \oplus m_0$ και το $IV' \oplus m_1$ να δώσουν ίδια κρυπτογράφηση c διότι το Enc είναι μετάθεση).

Άρα αν $c = c'$ επιστρέφω 0 αλλιώς επιστρέφω 0.

Όπως εξηγήθηκε με πιθανότητα $1 - negl(n)$ το μαντείο ρυθμίστηκε σωστά, ώστε IV' να λήγει σε 0. Με δεδομένο ότι συνέβη αυτό, ο $\mathcal{A}$ επιτυγχάνει πάντα.

Άρα:

$$\mathbb{P}\left[PrivK^{cpa}_{\mathcal{A}',\mathcal{CS}}(n)\right] \geq 1 - negl(n) \tag{12.2}$$

Δεν υπάρχει αμελητέα συνάρτηση $negl'(n)$ ώστε $\frac{1}{2} + negl'(n) = 1 - negl(n) \iff negl'(n) + negl(n) = \frac{1}{2}$, καθώς το άθροισμα αμελητέων δεν μπορεί να δώσει σταθερά.

Συνεπώς το τροποποιημένο σύστημα CBC δεν προσφέρει CPA ασφάλεια.

12.3 Απουσία CCA ασφάλειας στο OFB Block Cipher Mode Of Operation

Θα κατασκευάσουμε έναν αντίπαλο $\mathcal{A}$ που κερδίζει το $PrivK^{cca}_{\mathcal{A},OFB}(n)$ με με πιθανότητα $\frac{1}{2} + f(n)$, όπου $f(n)$ μη-αμελητέα, οπότε το σύστημα δεν προσφέρει CCA ασφάλεια.

Ο αντίπαλος $\mathcal{A}$ είναι ο εξής:

Επιλέγει δύο arbitrary μηνύματα m_0, m_1 ώστε να διαφέρουν στο πρώτο block (π.χ. $m_0 = 0^n$, $m_1 = 1^n$) και τα στέλνει.

Του επιστρέφεται ένα κρυπτοκείμενο c που περιλαμβάνει το IV που χρησιμοποιήθηκε.

Γενικά συμβολίζω x_{start} το πρώτο block ενός μηνύματος ή κρυπτοκειμένου x.

Είναι $c_{start} = Enc_k(IV) \oplus m_{b,start}$.

Στέλνει στο μαντείο αποκρυπτογράφησης το $\tilde{c} = m_0$ και το IV, οπότε επιστρέφεται ένα m'.

Είναι: $m'_{start} = Enc_k(IV) \oplus \tilde{c} = Enc_k(IV) \oplus m_{0,start}$.

Συνεπώς $m'_{start} = c_{start} \iff Enc_k(IV) \oplus m_{0,start} = Enc_k(IV) \oplus m_{b,start} \iff m_0 = m_b$.

Άρα αν το πρώτο block του m' και του c είναι ίδια επιστρέφει 0, αλλιώς επιστρέφει 1.

Ο $\mathcal{A}$ επιτυγχάνει ντετερμινιστικά. Συνεπώς το OFB δεν προσφέρει CCA ασφάλεια.

Παράρτημα

Α΄ Άσκηση 9: Κώδικας Haskell

Α΄.1 Blum-Blum-Shub package

Α΄.1.1 NTheory.hs

```haskell
module NTheory
  ( egcd
  , gcd_
  , modinv
  , isCoprime
  , crt
  , modexp
  , ordm_
  ) where

-- calcalates b^e mod m
modexp :: Integer -> Integer -> Integer -> Integer
modexp b e m =
  let aux _ 0 acc = acc
      aux b1 e1 acc =
        let acc' =
              if odd e1
                then (b1 * acc) `mod` m
                else acc
         in aux ((b1 * b1) `mod` m) (e1 `div` 2) acc'
   in aux b e 1



-- Returns (d,s,t) of the Bezout identity s*a + t*b = d
egcd :: Integer -> Integer -> (Integer, Integer, Integer)
egcd a b =
  let egcd_ rk sk tk 0 _ _ = (rk, sk, tk)
      egcd_ r0 s0 t0 r1 s1 t1 =
        let (q2, r2) = quotRem r0 r1
            s2 = s0 - q2 * s1
            t2 = t0 - q2 * t1
         in egcd_ r1 s1 t1 r2 s2 t2
   in egcd_ a 1 0 b 0 1

-- Calculates gcd of a, b. (simplification of egcd for efficiency)
gcd_ :: Integer -> Integer -> Integer
gcd_ d 0 = d
gcd_ a b = gcd_ b (a `mod` b)

-- Checks if a b are coprime
isCoprime :: Integer -> Integer -> Bool
isCoprime a b = gcd_ a b == 1

-- Return a^{-1} mod n. The caller must ensure a,n are coprime. a should be positive.
modinv :: Integer -> Integer -> Integer
modinv n a =
  let (_, _, t) = egcd n a
   in (n + (t `mod` n)) `mod` n

-- Solves Linear Congruence System x = a_i mod n_i (Chinese Remainder Theorem). Returns (x,
 ↪  prod ni)
```

```haskell
crt :: [Integer] -> [Integer] -> (Integer, Integer)
crt as ns =
  let n = product ns
      partial_ns = map (n `div`) ns
      ms = map (uncurry modinv) (zip ns partial_ns)
      x =
        foldl'
          (\acc (ai, mi, partial_ni) -> (acc + ai * mi * partial_ni) `mod` n)
          0
          (zip3 as ms partial_ns)
   in (x, n)

-- bruteforce find the order. for Testing Purposes. (like a functional specification)
ordm_ :: Integer -> Integer -> Integer
ordm_ a0 n = aux a0 1
  where
    aux 1 o = o
    aux a o = aux ((a*a0) `mod` n) (o+1)

```

A′.1.2 BBS.hs

```haskell
{-# LANGUAGE BangPatterns #-}

module BBS where

import Control.Monad.Random
import NTheory (crt, modexp)
import qualified MillerRabin (isPrime)

isPrime :: Integer -> Bool
isPrime = (flip MillerRabin.isPrime) 30

getGen :: (MonadRandom m) => Integer -> m Integer
getGen p = do
   g <- getRandomR (2,p-2)
   if modexp g ((p-1) `div` 2) p == 1 then getGen p else return g

get_s0 :: (MonadRandom m) => Integer -> Integer -> m Integer
get_s0 p q = do
  gp <- getGen p
  gq <- getGen q
  let (s0,_) = crt [gp,gq] [p,q]
  return s0

is_safesafe :: Integer -> Bool
is_safesafe p =
  let p' = (p-1) `div` 2
      p'' = (p'-1) `div` 2
  in isPrime p && isPrime p' && isPrime p''

get_safesafe :: (MonadRandom m) => Integer -> m Integer
get_safesafe l = do
  r <- getRandomR (0, 2^(l-5)-1)
  let p = (r+(2^(l-5)))*(2^4) + 7
  if is_safesafe p then return p else get_safesafe l


-- Returns (p,q,s_0). p,q with l bits. For BBS use n=p*q and initial value s1 = s0*s0.
get_bbs_setup :: (MonadRandom m) => Integer -> m (Integer, Integer, Integer)
get_bbs_setup l = do
```

```haskell
  p <- get_safesafe l
  q <- get_safesafe l
  if p==q then get_bbs_setup l else do
    s_0 <- get_s0 p q
    return (p,q,s_0)


-- takes n and returns the pseudorandom generator function. this function takes the previous value and outputs a new pseudorandom value.
bbs :: Integer -> (Integer -> Integer)
bbs n = (\s -> (s*s) `mod` n)

-- Finds the period of function f. if a value f xi = x0 is reached, then definitely the sequence will repeat afterwards.
findPeriod :: (Integer -> Integer) -> Integer -> Int
findPeriod f x0 = aux (f x0) 1
  where
    aux :: Integer -> Int -> Int
    aux !xi !cnt
      | xi == x0 = cnt
      | otherwise = aux (f xi) (cnt+1)

```

A′.1.3 Tests: Main.hs

```haskell
{-# LANGUAGE BangPatterns #-}

module Main
  ( main
  ) where

import NTheory
import BBS

import Test.Hspec
import Test.Hspec.QuickCheck
import Test.QuickCheck
import System.CPUTime

import Control.Monad.Random

newtype PositiveGt1 a = PositiveGt1
  { getPositiveGt1 :: a
  } deriving (Eq, Ord, Show, Read)

instance Functor PositiveGt1 where
  fmap f (PositiveGt1 x) = PositiveGt1 (f x)

instance (Num a, Ord a, Arbitrary a) => Arbitrary (PositiveGt1 a) where
  arbitrary = fmap PositiveGt1 (fmap abs arbitrary `suchThat` (> 1))
  shrink (PositiveGt1 x) = [PositiveGt1 x' | x' <- shrink x, x' > 1]

checkEGCD :: Integer -> Integer -> (Integer, Integer, Integer) -> Property
checkEGCD a b (d, s, t) =
  d === s * a + t * b .&&. a `rem` d === 0 .&&. b `rem` d === 0

propEGCD :: Positive Integer -> Positive Integer -> Property
propEGCD (Positive a) (Positive b) = checkEGCD a b (egcd a b)

propGCD :: Positive Integer -> Positive Integer -> Property
propGCD (Positive a) (Positive b) =
```

```
  let egcd_res@(d1, _, _) = egcd a b
      d2 = gcd_ a b
   in (checkEGCD a b egcd_res) .&&. d1 === d2

propModInv :: Positive Integer -> PositiveGt1 Integer -> Property
propModInv (Positive a) (PositiveGt1 n) =
  isCoprime a n
    ==> let a1 = modinv n a
         in 0 <= a1 .&&. a1 < n .&&. (a * a1) `mod` n === 1

genCRT :: Int -> Gen ([Integer], [Integer])
genCRT len = aux [] [] len
  where
    aux as ns 0 = return (as, ns)
    aux as ns n = do
      ni <-
        chooseInteger (2, 2 ^ (fromIntegral len))
          `suchThat` (\x -> all (isCoprime x) ns)
      ai <- chooseInteger (0, ni)
      aux (ai : as) (ni : ns) (n - 1)

propCRT :: ([Integer], [Integer]) -> Property
propCRT (!as, !ns) =
  let (x, n) = crt as ns
   in n === (product ns) .&&. 0
        <= x .&&. x
        < n .&&. all (\(ai, ni) -> x `mod` ni == ai `mod` ni) (zip as ns)

genGen :: Int -> Gen (Integer, Integer)
genGen len = do
  seed <- arbitraryBoundedIntegral
  let gp = do
        p <- get_safesafe (fromIntegral len+16)
        g <- getGen p
        return (g,p)
  return $ evalRand gp (mkStdGen seed)

propGen :: (Integer, Integer) -> Property
propGen (g, p) =
  let o = ordm_ g p
  in o === p - 1

genBBS :: Integer -> Gen (Integer, Integer, Integer)
genBBS len = do
  seed <- arbitraryBoundedIntegral
  let bbs_setup = get_bbs_setup len
  return $ evalRand bbs_setup (mkStdGen seed)


propBBSPeriod :: (Integer,Integer,Integer) -> Property
propBBSPeriod (p,q,s0) =
  let n = p*q
      s1 = (s0 * s0) `mod` n
      g = bbs n
      t_ideal = 2* ((p-3) `div` 4) * ((q-3) `div` 4)
      t = findPeriod g s1
  in fromIntegral t === t_ideal



main :: IO ()
```

```
main =
  hspec $ do
    describe "EGCD"
      $ modifyMaxSuccess (const 10000)
      $ modifyMaxSize (const $ 10 ^ 9)
      $ prop "Bezout identity holds, d|a, d|b (these hold iff d=gcd(a,b)."
      $ propEGCD
    describe "GCD"
      $ modifyMaxSuccess (const 10000)
      $ modifyMaxSize (const $ 10 ^ 9)
      $ prop
          "Produces the same result as EGCD and EGCD results checks correct as above."
      $ propGCD
    describe "modinv"
      $ modifyMaxSuccess (const 10000)
      $ modifyMaxSize (const $ 10 ^ 9)
      $ prop "a * (modinv n a) \\equiv 1 \\pmod{n}."
      $ propModInv
    describe "crt"
      $ modifyMaxSuccess (const 100)
      $ modifyMaxSize (const 100)
      $ prop "solution satisfies the system and n is the product of the moduli."
      $ forAll (sized genCRT)
      $ propCRT
    describe "getGen"
      $ modifyMaxSuccess (const 100)
      $ modifyMaxSize (const $ 4)
      $ prop "checking generator has order (with brute-force calculation) p-1."
      $ forAll (sized genGen)
      $ propGen
    describe "Blub-Blum-Shub" $ do
      modifyMaxSuccess (const 10)
        $ prop "checking period is maximum for 15 bits i.e T == 2p''q''."
        $ forAll (genBBS 15)
        $ propBBSPeriod
      modifyMaxSuccess (const 1)
        $ prop "checking period is maximum for 20 bits i.e T == 2p''q''. Just 1 test because
↪  it takes long time."
        $ forAll (genBBS 20)
        $ propBBSPeriod

```

A′.2 Miller-Rabin package

A′.2.1 MillerRabin.hs

```
module MillerRabin
  ( isPrime
  ) where

import Control.Monad
import System.Random.Stateful

modexp :: Integer -> Integer -> Integer -> Integer
modexp b e m =
  let aux _ 0 acc = acc
      aux b1 e1 acc =
        let acc' =
              if odd e1
                then (b1 * acc) `mod` m
                else acc
```

```
          in aux ((b1 * b1) `mod` m) (e1 `div` 2) acc'
    in aux b e 1

isPrime_pass :: Integer -> Integer -> Bool
isPrime_pass n b
  | modexp b (n - 1) n /= 1 = False
  | otherwise =
    let t = go (n - 1)
        go x
          | even x = go (x `div` 2)
          | otherwise = x
        aux 1 = False
        aux bt'
          | bt' == (n - 1) = True
          | otherwise = aux ((bt' * bt') `mod` n)
        bt = modexp b t n
     in if bt == 1 || bt == n - 1
          then True
          else aux bt

isPrime :: Integer -> Int -> Bool
isPrime 1 _ = False
isPrime n checks =
  let randomexp gen = do
        bs <- replicateM checks $ uniformRM (1, n - 1) gen
        return (all (isPrime_pass n) bs)
      pureGen = mkStdGen 42
   in runStateGen_ pureGen randomexp
```

A′.2.2 Tests: Main.hs

```
module Main (main) where

import System.Exit (exitFailure)
import MillerRabin (isPrime)
import Test.QuickCheck


isqrt :: Integer -> Integer
isqrt n
    | n < 2 = n
    | otherwise = aux 1 n
  where
    aux l r
        | r < l = r
        | m*m > n = aux l (m-1)
        | otherwise = aux (m+1) r
      where
        m = (l+r) `div` 2


isPrime_oracle :: Integer -> Bool
isPrime_oracle n
  | n <= 1 = False
  | n == 2 = True
  | even n = False
  | otherwise = null [x | x <- [3,5.. isqrt n], n `mod` x == 0]

checks :: Int
checks = 10

```

```haskell
test :: Integer -> IO ()
test n =
  let p = isPrime n checks
      p' = isPrime_oracle n
  in
  if p == p' then return () else do
     putStrLn $ "isPrime " ++ show n ++ " = " ++ show p ++ " , isPrime_oracle " ++ show n ++
  ↪  " = " ++ show p'
     exitFailure

mersennes :: [Integer]
mersennes = [ 2^p-1 | p <- [3217, 4253, 4423, 9689] ]
nonmersennes :: [Integer]
nonmersennes = [ 2^i-1 | i <- [1234, 4567, 9876, 5432] ]

test_big :: Bool -> Integer -> IO ()
test_big is n =
  let p = isPrime n checks
  in
  if p == is then return () else do
     putStrLn $ "(the following is (/is not) a prime), isPrime " ++ show n ++ " = " ++ show p
     exitFailure

genLargeInt :: Gen Integer
genLargeInt = choose (10^6, 10^9)

prop_isPrime :: Integer -> Bool
prop_isPrime n = isPrime_oracle n == (isPrime n checks)

main :: IO ()
main = do
  mapM_ test [1..10000]
  putStrLn "Checked all numbers from 1 to 10000."

  putStrLn "Random checking 10^4 numbers in [10^6, 10^9] using QuickCheck."
  quickCheck $ withMaxSuccess 10000 $ forAll genLargeInt prop_isPrime

  mapM_ (test_big True) mersennes
  putStrLn "Checked some bigint mersenne primes."

  mapM_ (test_big False) nonmersennes
  putStrLn "Checked some bigint, mersenne-like non-primes (2^i-1)."

  test_big True 67280421310721
  putStrLn "Checked 67280421310721, found correctly it is a prime."

  test_big False 170141183460469231731687303715884105 7
  putStrLn "Checked 170141183460469231731687303715884105 7, found correctly it is not a
  ↪  prime."

  test_big False (2^1001 - 1)
  putStrLn "Checked 2^1001 - 1, found correctly it is not a prime."

  test_big True (2^2281 - 1)
  putStrLn "Checked 2^2281 - 1, found correctly it is a prime."

  test_big True (2^9941 - 1)
  putStrLn "Checked 2^9941 - 1, found correctly it is a prime."

```