ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

3η Σειρά Ασκήσεων

Υπολογιστική Κρυπτογραφία

Ανδρέας Στάμος

Αριθμός μητρώου: 03120***

Διεύθυνση ηλεκτρονικού ταχυδρομείου: stamos.aa@gmail.com

9ο Εξάμηνο — 2024-2025

Περιεχόμενα

1 Άσκηση 1η: Exploit σε λανθασμένη υλοποίηση του RSA

1.1 Περιγραφή συστήματος

Αρχικά εκτελούμε την διαδικασία γέννησης κλειδιών RSA $(p, q, e, d) \leftarrow KGen_{RSA}(1^\lambda)$.

Διαθέτουμε ένα μαντείο $\mathcal{O}^{p,q,e}(m)$ που υπολογίζει το εξής:

1. $c_p \leftarrow m^e \pmod p$
2. $c_q \leftarrow m^e + 1 \pmod q$.
3. Με βάση το Κινέζικο Θεώρημα Υπολοίπων, λύνει το εξής Σύστημα Ισοτιμιών:

$$\begin{cases} c \equiv c_p \pmod p \\ c \equiv c_q \pmod q \end{cases}$$

Λαμβάνεται μια (μοναδική) απάντηση $c \pmod{pq}$ και αυτή επιστρέφεται ως απάντηση του μαντείου.

Το μαντείο αυτό υλοποιείται από την συσκευή κρυπτογράφησης που έδωσε ο διευθυντής στην γραμματέα του.

1.2 Παραβίαση ορθότητας

Από τον διευθυντή έγινε άμεσα αντιληπτό ότι υπάρχει πρόβλημα, διότι η συσκευή κρυπτογράφησης με είσοδο μηνύματα παράγει κρυπτοκείμενα που δεν αποκρυπτογραφούνται στο αρχικό κείμενο. Αυτό, φυσικά, μπορεί να έγινε διαισθητικά αντιληπτό επειδή τα μηνύματα δεν βγάζανε νόημα, ωστόσο, μπορεί να υπήρχε και κάποια μέθοδος ανίχνευσης σφαλμάτων (ακόμα και για μη κρυπτογραφική χρήση, π.χ. για ανίχνευση σφαλμάτων μετάδοσης) που θα αποτύγχανε πάντα.

Θα αποδείξουμε ότι η συσκευή κρυπτογράφησης γυρνάει πάντα κρυπτοκείμενα που δεν αποκρυπτογραφούνται σωστά. Έστω ένα αρχικό μήνυμα m. Έστω ότι ένα κρυπτοκείμενο c αποκρυπτογραφείται σωστά. Τότε:

$$m \equiv c^d \pmod n \Rightarrow m^e \equiv c^{de} \equiv c \pmod n$$

Με άλλα λόγια, μόνο το προβλεπόμενο κρυπτοκείμενο του RSA αποκρυπτογραφείται σωστά. Ωστόσο αποκλείεται $c \equiv m^e \pmod n$ διότι τότε $c_p \equiv m^e + 1 \pmod n \Rightarrow m^e \equiv m^e + 1 \pmod n \Rightarrow 0 \equiv 1 \pmod n$, που είναι αδύνατο.

Συνεπώς, όλα τα κρυπτοκείμενα της συσκευής αποκρυπτογραφούνται λάθος.

1.3 Εύρεση ιδιωτικού κλειδιού

Σκοπός είναι η κατασκευή ενός αλγορίθμου $\mathcal{A}^{\mathcal{O}^{p,q,e},(e,n)}$ που με είσοδο το μαντείο (δηλαδή την συσκευή) καθώς και το δημόσιο κλειδί $(e, n) = (e, pq)$ υπολογίζει το ιδιωτικό κλειδί d.

Η γραμματέας έτρεξε τον αλγόριθμο αυτό θεωρώντας μαντείο την συσκευή της.

Δίνουμε ως είσοδο το $m = 0$ στο μαντείο.

Το μαντείο επιστέφει αριθμό c ώστε:

1. $c \equiv 0^e \equiv 0 \pmod p$
2. $c \equiv 0^e + 1 \equiv 1 \pmod q$

Άρα ο αριθμός c έχει παράγοντα τον p αλλά όχι τον q.

Συνεπώς υπολογίζουμε:

$$\gcd(c, n) = \gcd(c, pq) \overset{p|c, q\nmid c}{=} p$$

Διαθέτουμε έναν πρώτο παράγοντα p του αριθμού n. Κατά τα γνωστά, σπάσαμε το σύστημα RSA, διότι υπολογίζουμε το $q = \frac{n}{p} = \frac{pq}{p}$ και έπειτα υπολογίζουμε το d όπως η $KGen_{RSA}$. Βρίσκουμε το $\phi(n) = \phi(pq) = \phi(p)\phi(q) = (p-1)(q-1)$ και με χρήση του Εκτεταμένου Αλγόριθμου Ευκλειδή (EGCD) υπολογίζουμε τον d ως το $e^{-1} \pmod{\phi(n)}$ (με είσοδο $(e, \phi(n))$ ο EGCD επιστρέφει τους συντελεστές Bézout (E, N). Είναι $E \equiv e^{-1} \pmod{\phi(n)}$).

Ο $\mathcal{A}$ εκτελείται σε χρόνο $O(\lambda^2)$, καθώς εκτελεί μόνο 2 φορές τον αλγόριθμο Ευκλείδη.

(ο αλγόριθμος Ευκλείδη έχει πολυπλοκότητα $O(\lambda^2)$ και όχι $O(\lambda^3)$ όπως φαινομενικά φαίνεται, διότι καθώς προχωρά η εκτέλεση οι αριθμοί μικραίνουν)

2 Άσκηση 2η: Συνάρτηση Κατακερματισμού βασισμένη στο πρόβλημα αντιστροφής κλειδιού RSA

2.1 Περιγραφή συνάρτησης

Αρχικά εκτελούμε την διαδικασία γέννησης κλειδιών RSA $(N, e, d) \leftarrow KGen_{RSA}(1^\lambda)$.

Επιλέγουμε έναν ομοιόμορφα τυχαίο αριθμο $y \xleftarrow{R} \mathbb{Z}_N^*$.

Θεωρούμε ως "κλειδί" της συνάρτησης κατακερματισμού το $s = (N, e, y)$.

Κατασκευάζουμε μια συνάρτηση κατακερματισμού $H^s(\chi) : \{0,1\}^{3n} \mapsto \mathbb{Z}_N^*$.

Θεωρούμε την συνάρτηση $f_b^s(x) = y^b x^e \pmod N$ με $b \in \{0,1\}$.

Τότε ορίζουμε την $H^s(x)$ ως:

$$H^s(x) = f_{x_1}^s\left(f_{x_2}^2\left(\cdots(1)\cdots\right)\right) \text{ 3n εφαρμογές συναρτήσεων} \tag{2.1}$$

2.2 Απόδειξη Collision-Resistance (ισοδύναμη με Πρόβλημα Αντιστροφής Κλειδιού RSA)

Αρχικά δείχνουμε μια απλούστερη μορφή της $H_s(x)$.

Λήμμα 2.1. *Ισχύει ότι:*

$$H^s(x) = y^{x_1 + x_2 e + x_3 e^2 + \cdots + x_{3n} e^{3n}} = y^{\sum_{i=1}^{3n} x_i e^{i-1}} \pmod N \tag{2.2}$$

Απόδειξη. Η απόδειξη γίνεται με επαγωγή στο n.

Για $n = 1$ είναι: $H_1^s(x) = f_{x_1}^s(1) = y^{x_1} 1^e = y^{x_1}$, οπότε η πρόταση ισχύει.

Έστω ότι η πρόταση ισχύει για n. Θα δείξουμε ότι ισχύει για $n+1$.

Για $n+1$:

$$\begin{aligned} H_{n+1}^s(x) &= f_{x_1}^s\left(f_{x_2}^2\left(\cdots(1)\cdots\right)\right) = \\ &= f_{x_1}^s\left(H_n\left(x >> 1\right)\right) = \\ &= y^{x_1}\left(y^{x_2 + x_3 e + \cdots + x_{3n} e^{3n-1}}\right)^e = \\ &= y^{x_1 + x_2 e + x_3 e^2 + \cdots + x_{3n} e^{3n}} \end{aligned}$$

οπότε η πρόταση ισχύει. □

Θα δείξουμε αναγωγή του προβλήματος αντιστροφής κλειδιού RSA (εύρεση $d = e^{-1}$ από το δημόσιο κλειδί (n, e)) στο πρόβλημα εύρεσης συγκρούσης για την συνάρτηση κατακερματισμού $H_s(x)$ (δηλαδή της εύρεσης $u \neq v$ ώστε $H^s(u) = H^s(v)$).

Ισχύει επίσης ότι το πρόβλημα παραγοντοποίησης ανάγεται στο πρόβλημα αντιστροφής κλειδιού RSA. Συνεπώς τελικά το πρόβλημα εύρεσης συγκρούσεων της $H_s(x)$ θα είναι τουλάχιστον τόσο δύσκολο όσο της παραγοντοποίησης.

Στην εκφώνηση ζητείται αναγωγή του προβλήματος RSA, που ενίοτε εννοείται το πρόβλημα εύρεσης m από ομοιομόρφα τυχαίο στοιχείο c ώστε $c \equiv m^e \pmod n$. Το πρόβλημα αυτό δεν είναι δυσκολότερο της παραγοντοποίησης, σε αντίθεση με το πρόβλημα αντίστροφης κλειδιού RSA.

Έστω ένας αλγόριθμος $\mathcal{A}^s$ που με είσοδο το "κλειδί" s της συνάρτησης κατακερματισμού μας επιστρέφει μια σύγκρουση (u, v), $u \neq v$, $H^s(u) = H_s(v)$.

Κατασκευάζουμε αλγόριθμο αντίστροφης κλειδιού RSA, που με είσοδο το δημόσιο κλειδί $(n = pq, e)$ επιστρέφει το $d \equiv e^{-1} \pmod n$.

Επιλέγουμε έναν arbitrary αριθμό $y \in \mathbb{Z}_n^*$. Ενδεικτικά θέτουμε $y = 2$ που δίνει $\gcd(2, n) = 1$ εκτός αν n άρτιος, οπότε θα είχαμε την παραγοντοποίηση του n και άρα εύκολα (όπως η $KGen_{RSA}$ θα βρίσκαμε το d).

Θεωρούμε το κλειδί $s = (n, e, y)$.

Εκτελούμε τον $\mathcal{A}^s$ και μας επιστρέφονται δύο αριθμοί $u, v \in \{0,1\}^{3n}$.

Είναι:

$$\begin{gathered} H^s(u) \equiv H^s(v) \pmod n \Rightarrow \\ y^{u_1+u_2e+u_3e^2+\cdots} \equiv y^{v_1+v_2e+v_3e^2+\cdots} \pmod{\phi(n)} \Rightarrow \\ u_1+u_2e+u_3e^2+\cdots \equiv v_1+v_2e+v_3e^2+\cdots \pmod{\phi(n)} \Rightarrow \\ (u_1-v_1)+(u_2-v_2)e+(u_3-v_3)e^2+\cdots \equiv 0 \pmod{\phi(n)} \Rightarrow \\ \sum_{i=1}^{3n}(u_i-v_i)e^{i-1} \equiv 0 \pmod{\phi(n)} \end{gathered} \tag{2.3}$$

Θα δείξω ότι $\sum_{i=1}^{3n}(u_i-v_i)e^{i-1} \neq 0$ (υπολογισμοί χωρίς υπόλοιπα).

Επειδή $u_i, v_i \in \{0,1\}$ είναι $(u_i - v_i) \in \{-1,0,1\}$. Δείχνω το εξής λήμμα.

Λήμμα 2.2. *Έστω αριθμοί $a_i \in \{-1,0,1\}$, πλήθους n, και φυσικός $e > 0$. Ισχύει ότι:*

$$\sum_{i=1}^{n} a_i e^{i-1} \neq 0$$

Απόδειξη. Έστω ότι $\sum\limits_{i=1}^{n} a_i e^{i-1} = 0$.

Για $a_i = 0$, οι όροι $a_i e^{i-1} = 0$, οπότε μπορούν να αγνοηθούν.

Έστω τα σύνολα:

$$A_1 = \{i \mid a_i = 1\}$$
$$A_{-1} = \{i \mid a_i = -1\}$$

Τότε:

$$0 = \sum_{i=1}^{n} a_i e^{i-1} = \sum_{i\in A_1} a_i e^{i-1} + \sum_{i\in A_{-1}} a_i e^{i-1} = \sum_{i\in A_1} 1e^{i-1} + \sum_{i\in A_{-1}} (-1)e^{i-1} \Rightarrow \sum_{i\in A_1} e^{i-1} = \sum_{i\in A_{-1}} e^{i-1}$$

Έστω $i^* = \min(A_1 \sqcup A_{-1})$. Τα A_1, A_{-1} είναι ξένα μεταξύ τους, οπότε το i^* ανήκει σε μόνο ένα από τα δύο.

Έστω ότι $i^* \in A_1$ (όμοια αν ανήκει στο A_{-1}). Τότε:

$$\sum_{i\in A_1} e^{i-1} = \sum_{i\in A_{-1}} e^{i-1} \Rightarrow e^{i^*} \sum_{i\in A_1} e^{i-i^*-1} = e^{i^*} \sum_{i\in A_{-1}} e^{i-i^*-1} \Rightarrow \sum_{i\in A_1} e^{i-i^*-1} = \sum_{i\in A_{-1}} e^{i-i^*-1} \tag{2.4}$$

Επειδή $i^* \in A_1$, το αριστερό άθροισμα έχει έναν όρο με δύναμη 0, δηλαδή ίσο με 1, ενώ το δεξί άθροισμα δεν έχει τέτοιο όρο.

Συνεπώς το δεξί άθροισμα έχει παράγοντα το e. Εξαιρώντας τον όρο με τιμή 1, όλο το υπόλοιπο αριστερό άθροισμα αποτελείται από δυνάμεις του e με εκθέτες ≥ 1, συνεπώς έχει παράγοντα το e. Με άλλα λόγια το αριστερό άθροισμα είναι ισότιμο του $1 \pmod e$, ενώ το δεξί άθροισμα είναι ισότιμο του $0 \pmod e$, που είναι άτοπο. □

Άρα $\sum_{i=1}^{3n}(u_i-v_i)e^{i-1} \neq 0$.

Υπολογίζω συνεπώς το $x \leftarrow \sum_{i=1}^{3n}(u_i-v_i)e^{i-1}$ (υπολογισμοί χωρίς υπόλοιπα).

Όπως δείξαμε $x = \sum_{i=1}^{3n}(u_i-v_i)e^{i-1} \equiv 0 \pmod{\phi(n)}$, οπότε το x είναι πολλαπλάσιο του $\phi(n)$.

Τότε και το $-x$ είναι πολλαπλάσιο του $\phi(n)$. Συνεπώς θεωρούμε $x \leftarrow |x|$. (αυτό γίνεται για λόγους απλότητας)

Με δυαδική αναζήτηση βρίσκω τον μέγιστο αριθμό l, ώστε το e^l να είναι παράγοντας του x.

Η δυαδική αυτή αναζήτηση γίνεται ως εξής. Δοκιμάζω τις τιμές $l \leftarrow 1,2,4,8,\cdots$ και σταματώ στην πρώτη τιμή (ελάχιστη) που δίνει $e^{l'} \nmid x$. (από το e^l βρίσκω το $e^{2l} = \left(e^l\right)^2$, χωρίς νέα εκθέτιση). Έτσι έχω ένα διάστημα $[0, l')$ και γνωρίζω ότι η ζητούμενη τιμή l είναι εντός αυτού. Στην συνέχεια εκτελώ δυαδική αναζήτηση στο διάστημα αυτό. Γνωρίζω πως αν το $e^m \mid x$, τότε $l \geq m$ οπότε η δυαδική αναζήτηση είναι ορθή.

Άρα είναι $x = e^l \tilde{x}$ όπου $e \nmid \tilde{x}$.

Υπολογίζω, με διαίρεση, αυτό το $\tilde{x} = \frac{x}{e^l}$.

Είναι:

$$x = \sum_i \pm e^i = e^l \left(1 + \sum_{i \geq 1} \pm e^i\right)$$

Οπότε $\tilde{x} = 1 + \sum_{i \geq 1} \pm e^i$.

Συνεπώς αν για κάποιο d είναι $e \equiv 0 \pmod{d}$, τότε $\tilde{x} \equiv 1 \pmod{d}$.

Συνεπώς τα e και $\tilde{x}$ δεν έχουν κοινούς παράγοντες, δηλαδή $\gcd(e, \tilde{x}) = 1$.

Είναι $\phi(n) \mid x = e^l \tilde{x}$. Η $KGen_{RSA}$ έδωσε το e ώστε $\gcd(e, \phi(n)) = 1$. Αφού και $\gcd(e, \tilde{x}) = 1$, τότε $\phi(n) \mid \tilde{x}$.

Συνεπώς έχουμε έναν αριθμό $\tilde{x}$ που είναι παράγοντας του $\phi(n)$ και σχετικά πρώτος με το e.

Θα βρω το $e^{-1} \pmod{\tilde{x}}$ και αυτό θα είναι και $e^{-1} \pmod{\phi(n)}$ όπως θα δείξουμε.

Υπάρχει ένα r ώστε $\tilde{x} = r \cdot \phi(n)$.

Εκτελώ τον Εκτεταμένο αλγόριθμο Ευκλείδη για τους αριθμούς $(e, \tilde{x})$ και λαμβάνω τους συντελεστές Bézout (E, X) ώστε να ισχύει η ταυτότητα Bézout:

$$eE + \tilde{x}X = \gcd(e, \tilde{x}) = 1 \Rightarrow eE = 1 - r\phi(n)X \equiv 1 \pmod{\phi(n)} \tag{2.5}$$

Συνεπώς $E \equiv e^{-1} \pmod{\phi(n)}$.

Άρα επιστρέφουμε το E καθώς είναι το κλειδί αποκρυπτογράφησης RSA.

3 Άσκηση 3η: Παραλλαγές Προβλημάτος Computational Diffie-Hellman (CDH)

3.1 Περιγραφή

Έστω μια κυκλική ομάδα G με τάξη πρώτο q και γεννήτορα g.

Ορίζουμε το Πρόβλημα Υπολογισμού Diffie-Hellman (CDH) ως εξής:
Δίνονται ο γεννήτορας g και στοιχεία g^x και g^y. Να υπολογιστεί το g^{xy}.

Ορίζουμε το Τετραγωνικό Πρόβλημα Diffie-Hellman (SDH) ως εξής:
Δίνονται ο γεννήτορας g και στοιχείο g^x. Να υπολογιστεί το g^{x^2}.

Ορίζουμε το Πρόβλημα Diffie-Hellman Αντιστρόφου (IDH) ως εξής:
Δίνονται ο γεννήτορας g και στοιχείο g^x. Να υπολογιστεί το $g^{x^{-1}}$.

Θα δείξουμε ότι SDH $\equiv_P$ CDH, IDH $\equiv_P$ CDH. Προκύπτει συνεπώς από αυτά και ότι SDH $\equiv_P$ IDH. (πολυωνυμικά ισοδύναμα)

3.2 Πολυωνυμική Ισοδυναμία CDH και SDH

3.2.1 SDH $\leq_P$ CDH

Θα δείξουμε ότι το CDH είναι τουλάχιστον όσο δύσκολο το SDH. (SDH $\leq_P$ CDH).

Περιγράφουμε, συνεπώς, αναγωγή του SDH στο CDH.

Έστω ένα μαντείο $\mathcal{A}^G(g, g^x, g^y)$ που λύνει το πρόβλημα CDH, δηλαδή επιστρέφει g^{xy}.

Περιγράφω έναν πολυωνυμικό αλγόριθμο για το SDH, δηλαδή με είσοδο g, g^x βρίσκει το g^{x^2}.

Είναι: $g^{x^2} = g^{x \cdot x}$.

Συνεπώς εκτελώ $\mathcal{A}^G(g, g^x, g^x)$, λαμβάνει το $g^{x \cdot x} = g^{x^2}$ και το επιστρέφω.

3.2.2 CDH $\leq_P$ SDH

Θα δείξουμε ότι το SDH είναι τουλάχιστον όσο δύσκολο το CDH. (CDH $\leq_P$ SDH).

Περιγράφουμε, συνεπώς, αναγωγή του CDH στο SDH.

Έστω ένα μαντείο $\mathcal{A}^G(g, g^x)$ που λύνει το πρόβλημα SDH, δηλαδή επιστρέφει g^{x^2}.

Περιγράφω έναν πολυωνυμικό αλγόριθμο για το CDH, δηλαδή με είσοδο g, g^x, g^y βρίσκει το g^{xy}.

Υπολογίζω $g^{x+y} = g^x \cdot g^y$.

Εκτελώ $\mathcal{A}^G(g, g^{x+y}) = g^{(x+y)^2}$.

Είναι: $g^{(x+y)^2} = g^{x^2+y^2+2xy}$.

Εκτελώ $\mathcal{A}^G(g, g^x) = g^{x^2}$.

Εκτελώ $\mathcal{A}^G(g, g^y) = g^{y^2}$.

Η G έχει τάξη q, οπότε από Θεώρημα Lagrange για κάθε στοιχείο $a \in G$: $a^q = e \Rightarrow a^{q-1}a = e$, οπότε $a^{-1} = a^{q-1}$.

Με επαναλαμβανόμενο τετραγωνισμό υπολογίζω τα $\left(g^{x^2}\right)^{q-1} = g^{-x^2}$ και $\left(g^{y^2}\right)^{q-1} = g^{-y^2}$.

Έπειτα υπολογίζω $g^{(x+y)^2} g^{-x^2} g^{-y^2} = g^{x^2+y^2+2xy} g^{-x^2} g^{-y^2} = g^{2xy}$.

Αφού $q \in \mathbb{P}$ τότε q περιττός. (αν $q = 2$ λύνω το πρόβλημα απευθείας.)

Συνεπώς ο $q+1$ άρτιος, οπότε ο $\frac{q+1}{2}$ είναι φυσικός.

Τότε υπολογίζω:

$$\left(g^{2xy}\right)^{\frac{q+1}{2}} = \left(g^{q+1}\right)^{xy} = \left(g^q g\right)^{xy} = g^{xy}$$

Επιστρέφω το g^{xy} που υπολόγισα.

3.3 Πολυωνυμική Ισοδυναμία CDH και IDH

3.3.1 IDH $\leq_P$ CDH

Θα δείξουμε ότι το CDH είναι τουλάχιστον όσο δύσκολο το IDH. (IDH $\leq_P$ CDH).

Περιγράφουμε, συνεπώς, αναγωγή του IDH στο CDH.

Έστω ένα μαντείο $\mathcal{A}^G(g, g^x, g^y)$ που λύνει το πρόβλημα CDH, δηλαδή επιστρέφει g^{xy}.

Περιγράφω έναν πολυωνυμικό αλγόριθμο για το IDH, δηλαδή με είσοδο g, g^x βρίσκει το $g^{x^{-1}}$.

Επειδή η ομάδα G έχει τάξη q ο αριθμός x^{-1} εννοείται ως $x^{-1} \pmod q$.

(αν δοθεί το $g^0 = e$ η απάντηση δεν ορίζεται. Άρα η είσοδος e μπορεί να απαγορευτεί, και εξάλλου, αν δοθεί την αναγνωρίζουμε.)

Η $\mathbb{Z}_q^*$, αφού $q \in \mathbb{P}$, έχει τάξη $q-1$.

Από Θεώρημα Lagrange για κάθε $x \in \mathbb{Z}_q^*$: $x^{q-2}x \equiv x^{q-1} \equiv 1 \pmod q \Rightarrow x^{-1} \equiv x^{q-2} \pmod q$.

Συνεπώς: $g^{x^{-1}} = g^{x^{q-1}}$.

Ο αριθμός q είναι γνωστός, οπότε και ο $q-1$.

Από g^x, g^y βρίσκω το g^{xy}. Συνεπώς βρίσκω το $g^{x^{q-1}}$ με επαναλαμβανόμενο τετραγωνισμό.

Πιο συγκεκριμένα για κάθε εκθέτη n και κάθε στοιχείο a υπολογίζω αναδρομικά:

1. Αν ο n είναι άρτιος, τότε $n = 2n'$. Άρα υπολογίζω αναδρομικά ως εξής: $a^{x^n} = a^{x^{2n'}} = \left(a^{x^{n'}}\right)^2$.
2. Αν ο n είναι περιττός, τότε $n = 2n' + 1$. Άρα υπολογίζω αναδρομικά ως εξής: $a^{x^n} = \left(a^{x^{n'}}\right)^2 \cdot a$.

Με άλλα λόγια, θέτω αρχικά $res \leftarrow 1$ και διατρέχω τον n από το LSB προς το MSB. Σε κάθε βήμα τετραγωνίζω την βάση $a \leftarrow a^2$, και κάθε φορά που βρίσκω 1 θέτω $res \leftarrow a \cdot res$.

Έτσι βρίσκω το $g^{x^{q-1}} = g^{x^{-1}}$ και το επιστρέφω.

Απαιτείται χρόνος $\log q = \lambda$, δηλαδή πολυωνυμικός.

3.3.2 CDH $\leq_P$ IDH

Θα δείξουμε ότι το IDH είναι τουλάχιστον όσο δύσκολο το CDH. (CDH $\leq_P$ IDH).

Περιγράφουμε, συνεπώς, αναγωγή του CDH στο IDH.

Έστω ένα μαντείο $\mathcal{A}^G(g, g^x)$ που λύνει το πρόβλημα IDH, δηλαδή επιστρέφει $g^{x^{-1}}$.

Περιγράφω έναν πολυωνυμικό αλγόριθμο για το CDH, δηλαδή με είσοδο g, g^x, g^y βρίσκει το g^{xy}.

Για το σκοπό αυτό κατασκευάζω έναν πολυωνυμικό αλγόριθμο $\mathcal{B}$ για το SDH, δηλαδή με είσοδο g, g^x υπολογίζει το g^{x^2}.

Έπειτα εκτελώ τον πολυωνυμικό αλγόριθμο για το CDH που περιγράφηκε στην παράγραφο 3.2.2, και που χρησιμοποιεί ένα μαντείο για το IDH. Δίνω ως είσοδο το μαντείο $\mathcal{B}$ και τα g, g^x, g^y. Επιστρέφω την απάντηση, που είναι το g^{xy}.

Τελικά ο συνολικός αλγόριθμος είναι πολυωνυμικός.

Συνοψίζοντας, το μαντείο $\mathcal{A}$ λύνει το IDH και χρειαζόμαστε έναν αλγόριθμο που με είσοδο g, βρίσκει το g^{x^2}.

Αν $g^x = e = g^0$, τότε $x \equiv 0 \pmod q \Rightarrow 2x \equiv 0 \pmod q$, οπότε επιστέφω τό $g^{2x} = e$.

Αλλιώς, επειδή $q \in \mathbb{P}$ είναι $\gcd(x \mod q, q) = 1$. Άρα ο $x^{-1} \pmod q$ ορίζεται.

Τότε το g^x είναι γεννήτορας της G. Το αποδεικνύω.

Έστω στοιχείο $a \in G$. Υπάρχει $k \in \mathbb{Z}_q$ ώστε $a = g^k = g^{k \cdot 1} = g^{kxx^{-1}} = \left(g^{kx}\right)^{x^{-1}}$.

Συγκεκριμένα για το στοιχείο g προκύπτει: $g = g^1 = (g^x)^{x^{-1}}$

Εκτελώ τον $\mathcal{A}$ για τον γεννήτορα g^x και το στοιχείο g και λαμβάνω ως απάντηση το $(g^x)^{(x^{-1})^{-1}} = g^{x^2}$, το οποίο και επιστρέφω.

4 Άσκηση 4η: Επίθεση Αποκρυπτογράφησης RSA με χρήσης της συνάρτησης loc

4.1 Περιγραφή της επίθεσης

Έστω (n, e) το δημόσιο κλειδί RSA.

Έστω μήνυμα $m \in \mathbb{Z}_{\mathbb{p}}^*$ και η κρυπτογράφηση του $c \equiv m^e \pmod{n}$.

Η συνάρτηση $loc^{pk}(c)$ ορίζεται ως 1 (αντίστοιχα 0) αν $m > \frac{n}{2}$ (αντίστοιχα αντίστροφα).

Επειδή το n δεν έχει παράγοντα το 2 (αν το έχει σπάμε απευθείας το RSA αφού έχουμε παραγοντοποίηση του n), το $\frac{n}{2}$ δεν είναι φυσικός. Συνεπώς αν $loc(c) = 0$ τότε $m < \frac{n}{2}$ (ισχύει γνήσια η ανισότητα).

Θεωρώντας την loc ως μαντείο, κατασκευάζω έναν πολυωνυμικό αλγόριθμο που υπολογίζει το $Dec(c)$.

Έστω ότι γνωρίζω για κάποιο m ότι ισχύει για κάποια γνωστά i,k:

$$i\frac{n}{2^k} \leq m < (i+1)\frac{n}{2^k}$$

Τότε $in \leq m2^k \leq (i+1)n \Rightarrow 0 \leq m2^k - in < n$.

Συνεπώς $m2^k \mod n = m2^k - in$. (πρόκειται για υπολογισμό υπολοίπου όχι ισοτιμία).

Υπολογίζω το $Enc(m2^k) = Enc(m)Enc(2^k) = cEnc(2^k)$.

Έπειτα υπολογίζω το $loc(Enc(m2^k))$.

Διακρίνω περιπτώσεις:

1. Αν $loc(Enc(m2^k)) = 1$.

   Τότε:
   $$m2^k \mod n > \frac{n}{2} \Rightarrow m2^k - in > \frac{n}{2} \Rightarrow m > (2i+1)\frac{n}{2^{k+1}}$$

   Άρα:
   $$(2i+1)\frac{n}{2^{k+1}} \leq m < (2i+2)\frac{n}{2^{k+1}}$$

2. Αν $loc(Enc(m2^k)) = 0$.

   Όμοια:
   $$m2^k \mod n < \frac{n}{2} \Rightarrow m2^k - in < \frac{n}{2} \Rightarrow m < (2i+1)\frac{n}{2^{k+1}}$$

   Άρα:
   $$2i\frac{n}{2^{k+1}} \leq m < (2i+1)\frac{n}{2^{k+1}}$$

Συνεπώς αποκτήσαμε μια νέα ανισότητα όπως αυτή που είχαμε όμως για $k' = k+1$, οπότε αυτό το αλγοριθμικό βήμα μπορεί να εκτελεστεί εκ νέου.

Σημείωση: Επειδή $(2i+1)\frac{n}{2^{k+1}} = \frac{1}{2} \cdot (i\frac{n}{2^k} + (i+1)\frac{n}{2^k})$, στην πραγματικότητα πρόκειται για ένα βήμα δυαδικής ανάζητησης.

Συνεπώς προκύπτει ο εξής αλγόριθμος.

Αρχικά γνωρίζω ότι $0 \leq m < n \Rightarrow 0\frac{n}{2^0} \leq m < 1\frac{n}{2^0}$.

Εκτελώ επαναληπτικά σε αυτή την ανισότητα το παραπάνω αλγοριθμικό βήμα μέχρι να προκύψει ανισότητα της μορφής:

$$i\frac{n}{2^k} \leq m < (i+1)\frac{n}{2^k}$$

όπου να ισχύει ότι:

$$(i+1)\frac{n}{2^k} - i\frac{n}{2^k} \leq 1 \iff 2^k \geq n$$

καθώς τότε γνωρίζω ότι:

$$m = \lfloor (i+1)\frac{n}{2^k} \rfloor$$

Αυτό προκύπτει διότι πρόκειται για ακέραιο αριθμό m ανάμεσα σε δύο πραγματικούς x_1, x_2 που απέχουν λιγότερο από 1. Αν υπάρχει αυτός ο αριθμός, τότε είναι ο $\lfloor x_2 \rfloor$.

Σε κάθε επανάληψη το k αυξάνεται κατά 1. Αν l το πλήθος bits του αριθμού n (αφού και $2 \nmid n$), ο l είναι ο ελάχιστος αριθμός ώστε $2^l \geq n$.

Συνεπώς εκτελώ ακριβώς l επαναλήψεις, χωρίς να ελέγχω το παραπάνω κριτήριο κάθε φορά.

Στην j-οστή επανάληψη πρέπει να υπολογίσω το $Enc(m2^j)$. Ωστόσο στην προηγούμενη επανάληψη είχε υπολογιστεί το $Enc(m2^{j-1})$. Συνεπώς υπολογίζω: $Enc(m2^j) = Enc(m2^{j-1}2) = Enc(m2^{j-1}) \cdot Enc(2)$, όπου το $Enc(2)$ μπορεί να υπολογιστεί μόνο μια φορά στην αρχή.

Στην πράξη, κατά την εκτέλεση ο αλγόριθμος χρειάζεται να διατηρεί, εκτός του κρυπτοκειμένου της τελευταίας επανάληψης, μόνο την μεταβλητή i του αριστερού άκρου της ανισότητας, που είναι και η μόνη που χρειάζεται στο τέλος.

Αυτό προκύπτει, διότι όπως φαίνεται παραπάνω σε κάθε επανάληψη αν προκύψει $loc = 0$ θέτουμε $i' = 2i + 1$, ενώ αν προκύψει $loc = 1$ θέτουμε $i = 2i$.

Συνολικά, σε κάθε επανάληψη, εκτελούμε έναν πολλαπλασιασμό δύο αριθμών με το πολύ $|n|$ bits, μια κλήση του μαντείου loc και ένα binary shift αριθμού με $|l|$ bits (όσο το πλήθος των επαναλήψεων).

Γίνονται l επαναλήψεις. Συνεπώς ο αλγόριθμος έχει πολυπλοκότητα $O(|n|^2) = O(\lambda^2)$ (θεωρώντας ως μαντείο την loc).

Επίσης, αν η loc λειτουργεί κάνοντας αποκρυπτογράφηση γνωρίζοντας το ιδιωτικό κλειδί, όπως θα δοκιμάσουμε στην συνέχεια, τότε χρειάζεται χρόνο $O(\lambda^3)$ για τον υπολογισμό της, οπότε ο συνολικός υπολογισμός ανεβαίνει σε χρόνο $O(\lambda^4)$. Για $\lambda = 2048$ bits, είναι περίπου $\lambda^4 \approx 9 \cdot 10^9$, οπότε περιμένουμε τον αλγόριθμο να χρειάστει (σε έναν προσωπικό ηλεκτρονικό υπολογιστή) χρόνο της τάξης των μερικών δευτερολέπτων ως δεκάδες δευτερόλεπτα.

4.2 Υλοποίηση

Η υλοποίηση έγινε σε γλώσσα προγραμματισμού Python με χρήση της Κρυπτογραφικής Βιβλιοθήκης PyCryptodome για την παραγωγή του RSA κλειδιού. (Θα μπορούσαμε να την είχαμε υλοποιήσει μόνοι μας, και αυτό θα ήταν εύκολο, αφού από τις προηγούμενες σειρές ασκήσεων, διαθέτουμε υλοποιημένη την γέννηση πρώτου και τον υπολογισμό πολλαπλασιαστικού αντιστρόφου. Ωστόσο αφού αυτά έχουν ήδη γίνει, είναι ευκαιρία να δοκιμάσουμε και μια έτοιμη υλοποίηση.)

Στον κώδικα γίνονται τυχαιοποιημένοι έλεγχοι για τυχαία μηνύματα, πως η επίθεση επιτυγχάνει. Πράγματι, βρίσκουμε ότι επιτυγχάνει σε όλους τους τυχαίους ελέγχους που έγιναν. Για κλειδί 1024 bits (μέγεθος αριθμού n) το εύρος ζώνης αποκρυπτογράφησης μετρήθηκε περίπου 150 bps (bits per second) ενώ για κλειδί 2048 bits μετρήθηκε περίπου 23 bps (bits per second). Με 2-πλασιασμό κλειδιού, έπρεπε να πρόκυπτει 8-πλάσιος χρόνος, δηλαδή 8 φορές χαμηλότερο εύρος ζώνης, που πράγματι προσεγγιστικά ισχύει.

Παραθέτουμε τον κώδικα Python:

```python
from Crypto.PublicKey import RSA
import Crypto.Random.random
import tqdm

L = 1024
NUM_TESTS = 10


def setup():
    key = RSA.generate(L)
    enc = lambda m: pow(m, key.e, key.n)
    loc = lambda c: pow(c, key.d, key.n) > key.n // 2
    return key.public_key(), enc, loc

```

```
15
16  def adversary(pk, enc, loc, c):
17      c2 = enc(2)
18      i = 0
19      for _ in tqdm.tqdm(range(pk.n.bit_length()), position=1):
20          b = loc(c)
21          c = (c2 * c) % pk.n
22          if b == True:
23              i = 2 * i + 1
24          else:
25              i = 2 * i
26      return ((i + 1) * pk.n) >> pk.n.bit_length()
27
28
29  def test():
30      pk, enc, loc = setup()
31      m = Crypto.Random.random.getrandbits(L) % pk.n
32      c = enc(m)
33      m1 = adversary(pk, enc, loc, c)
34      assert m == m1
35
36
37  if __name__ == "__main__":
38      for i in tqdm.tqdm(range(1, NUM_TESTS + 1), position=0):
39          test()
40      print("TESTS OK")
```

5 Άσκηση 5η: Χρήση του συμβόλου Legendre για προβλήματα Διακριτού Λογάριθμου στο $\mathbb{Z}_p^*$

5.1 Περιγραφή αλγόριθμου για το Decisional Diffie-Hellman

Η ιδέα ξεκινά από έναν γνωστό αλγόριθμο $\mathcal{A}$ για την επίλυση του DDH στο $\mathbb{Z}_p^*$.

Ο αλγόριθμος $\mathcal{A}$ πρέπει με είσοδο τρία στοίχεια g^x, g^y, g^z να διακρίνει αν $g^z = g^{xy}$, και να επιστρέψει σωστά NAI/OXI με μη-αμελητέα πιθανότητα.

Πιο αυστήρα, για τον $\mathcal{A}$ πρέπει να ισχύει ότι η συνάρτηση:

$$|\mathbb{P}\left[\mathcal{A}(g^x, g^y, g^z) = 1\right] - \mathbb{P}\left[\mathcal{A}(g^x, g^y, g^{xy}) = 1\right]| \tag{5.1}$$

όπου τα $x, y, z \xleftarrow{R} Z_q$, δεν είναι αμελητέα ως προς την παράμετρο ασφάλειας, που είναι γενικά το μέγεθος της εισόδου, και εδώ θεωρείται το $|p| = n$.

Ισχύει για το σύμβολο Legendre:

$$\left(\frac{g^x}{p}\right) \equiv (g^x)^{\frac{p-1}{2}} \equiv \left(g^{\frac{p-1}{2}}\right)^x \pmod p \tag{5.2}$$

Από το Θεώρημα Lagrange: $g^{p-1} \equiv 1 \pmod p$. Επειδή $p \in \mathbb{P}$, το 1 έχει δύο τετραγωνικές ρίζες $\bmod p$, τις 1 και -1.

Αποκλείεται $g^{\frac{p-1}{2}} = 1$, διότι $\text{ord}\,(g) = p$. Άρα: $g^{\frac{p-1}{2}} \equiv -1 \pmod p$.

Άρα: $\left(\frac{g^x}{p}\right) \equiv \left(g^{\frac{p-1}{2}}\right)^x \equiv (-1)^x \pmod p$.

Συνεπώς:

$$\left(\frac{g^x}{p}\right) = \begin{cases} 1 & x \text{ άρτιος} \\ -1 & x \text{ περιττός} \end{cases} \tag{5.3}$$

Τότε ο άλγοριθμος $\mathcal{A}(g^x, g^y, g^z)$ είναι ο εξής:

Επιστρέφει NAI, ή αντίστοιχα OXI, αν ισχύει, ή αντίστοιχα δεν ισχύει, η ισότητα:

$$\left(\left(\frac{g^x}{p}\right) = 1\right) \vee \left(\left(\frac{g^y}{p}\right) = 1\right) = \left(\left(\frac{g^z}{p}\right) = 1\right) \tag{5.4}$$

Ισοδύναμα, με βάση τα προηγούμενα, ο $\mathcal{A}$ εξετάζει αν το x ή το y είναι άρτιος, ισοδύναμα αν το xy είναι άρτιος, και έπειτα ελέγχει αν ισχύει το ίδιο για το z.

Αν δοθεί $g^z = g^{xy}$, τότε ο αλγόριθμος επιστρέφει NAI (δηλαδή 1) με πιθανότητα 1.

Αν δοθεί το g^z με το ομοιόμορφα τυχαίο z, τότε η πιθανότητα το z να είναι άρτιο είναι $\frac{1}{2}$, και κατ'επέκταση η πιθανότητα να έχει ίδια αρτιότητα με το xy, οπότε ο αλγόριθμος να απαντήσει NAI (δηλαδή 1), είναι $\frac{1}{2}$, λόγω της ανεξαρτησίας της z, από τις x, y.

Έτσι προκύπτει:

$$|\mathbb{P}\left[\mathcal{A}(g^x, g^y, g^z) = 1\right] - \mathbb{P}\left[\mathcal{A}(g^x, g^y, g^{xy}) = 1\right]| = \left|1 - \frac{1}{2}\right| = \frac{1}{2} \tag{5.5}$$

που δεν είναι αμελητέο.

5.2 Ιδέα επέκτασης στο Πρόβλημα Διακριτού Λογάριθμου

Σκεφτόμαστε πως θα μπορούσαμε να επεκτείνουμε αυτή την ιδέα για τον υπολογισμό ενός διακριτού λογάριθμου στο $\mathbb{Z}_p^*$.

Πιο συγκεκριμένα, δοθέντος ενός γεννήτορα g και ενός στοιχείου g^x, σκοπός είναι η κατασκευή ενός (πολυωνυμικού) αλγορίθμου που βρίσκει το x.

Όπως δείξαμε παραπάνω το $\left(\frac{g^x}{p}\right)$ επιστρέφει το LSB bit (parity) του x.

Αν μπορούσαμε (αποδοτικά) να υπολογίσουμε το $g^{x>>1}$ θα μπορούσαμε έπειτα να βρούμε το LSB bit του $x >> 1$ (δεξιό shift κατά 1) δηλαδή το 2ο LSB bit του x, και έτσι επαναληπτικά να βρούμε όλο τον αριθμό χ. Όταν φτάσουμε σε στοιχείο $g^{x>>k} = e$ έχουμε προσδιορίσει πλήρως όλα τα bits του x.

Σκοπός εφεξής είναι ο υπολογισμός του $g^{x>>1}$ από το g^x.

Διακρίνουμε περιπτώσεις:

1. Έστω ότι βρήκαμε ότι ο $(\frac{g}{x}) = -1$, οπότε ο x είναι περιττός, δηλαδή $x = 2x' + 1$. Είναι $x' = x >> 1$.

   Τότε $g^x = g^{2x'}g \Rightarrow g^{2x'} = g^x g^{-1}$.

   Ισχύει ότι $g^{-1} = g^{p-1}$, οπότε το βρίσκουμε αποδοτικά με επαναλαμβανόμενο τετραγωνισμό (αυτό χρειάζεται να γίνει μόνο μια φορά στην αρχή της εκτέλεσης και όχι σε κάθε επανάληψη.)

   Άρα αρκεί να μπορούμε αποδοτικά να υπολογίσουμε το $g^{x'}$ από το $g^{2x'}$ και το αποτέλεσμα είναι το $g^{x>>1}$.

2. Έστω ότι βρήκαμε ότι ο $(\frac{g}{x}) = 1$, οπότε ο x είναι άρτιος, δηλαδή $x = 2x'$. Είναι $x' = x >> 1$.

   Τότε $g^x = g^{2x'}$.

   Άρα αρκεί και πάλι να μπορούμε αποδοτικά να υπολογίσουμε το $g^{x'}$ από το $g^{2x'}$ και το αποτέλεσμα είναι το $g^{x>>1}$.

Με άλλα λόγια, αν έχουμε έναν αλγόριθμο που για κάθε $0 \leq x' < \frac{p-1}{2}$, με είσοδο το $g^{2x'}$ υπολογίζει το $g^{x'}$ ακολουθώντας την παραπάνω διαδικασία βρίσκουμε διαδοχικά τα bits του x από το LSB προς το MSB και έτσι προσδιορίζουμε πλήρως τον διακριτό λογάριθμο x από το g^x.

Φαινομενικά, αυτός είναι ένας γνωστός αλγόριθμος, διότι το $g^{x'}$ είναι τετραγωνική ρίζα του $g^{2x'}$.

Ο αλγόριθμος Tonelli-Shanks βρίσκει αποδοτικά μια τετραγωνική ρίζα ενός τετραγωνικού υπόλοιπου $\mod p$.

Ωστόσο $\mod p$ κάθε τετραγωνικό υπόλοιπο έχει δύο ρίζες. Πιο συγκεκριμένα ένας αριθμός $x^2 \pmod p$ έχει τετραγωνικές ρίζες το x και το $-x$.

Αν είχαμε έναν αποδοτικό αλγόριθμο που διακρίνει μεταξύ των δύο τετραγωνικών ριζών $g^{x'}$ και $-g^{x'}$ του $g^{2x'}$, τότε θα τρέχαμε τον Tonelli-Shanks για το $g^{2x'}$ θα παίρναμε ως απάντηση το $\pm g^{x'}$, θα το δίναμε στον αλγόριθμο, και έτσι θα βρίσκαμε το $g^{x'}$. Έτσι, με βάση όλα τα παραπάνω, θα βρίσκαμε πράγματι αποδοτικά τον διακριτό λογάριθμο x.

Αν δεν μπορούμε να διακρίνουμε μεταξύ των δύο ρίζων, σκεφτόμαστε ότι θα μπορούσαμε να δοκιμάσουμε και τις δύο και όταν στο τέλος βρούμε κάποιο άριθμο $\tilde{x}$, να εξετάσουμε αν αυτός είναι πράγματι $g^{tildex} = g^x$. Ωστόσο, επειδή σε κάθε βήμα από το LSB προς το MSB θα προκύπτουν 2 τετραγωνικές ρίζες, τελικά θα χρειαστεί νε εξετάσουμε 2^n περιπτώσεις (διότι $|x| = O(n)$), οπότε ο αλγόριθμος δεν θα είναι πολυωνυμικός όπως ζητάμε και άρα δεν θα είναι αποδεκτός.

Γενικά, δεν φαίνεται να υπάρχει κάποιος αλγόριθμος που να μπορεί να διακρίνει αυτές τις δύο τετραγωνικές ρίζες, και αυτό είναι γενικά λογικό, διότι το πρόβλημα διακριτού λογάριθμου στο $\mathbb{Z}_p^*$ πιστεύεται πως δεν λύνεται σε πολυωνυμικό χρόνο.

Μάλιστα, ο παραπάνω αλγόριθμος περιγράφει μια πολυωνυμική αναγωγή του υπολογισμού διακριτού λογάριθμου στο $\mathbb{Z}_p^*$ στο πρόβλημα της υπολογισμού της διάκρισης των δύο τετραγωνικών ριζών $g^{x'}$ και $-g^{x'}$ του $g^{2x'}$. Συνεπώς, το πρόβλημα αυτό είναι τουλάχιστον όσο δύσκολο είναι το πρόβλημα υπολογισμού διακριτού λογάριθμου στο $\mathbb{Z}_p^*$. Στην πραγματικότητα, ισχύει και το ανάποδο, διότι αν βρούμε τον διακριτό λογάριθμο $2x'$ του $g^{2x'}$ προσδιορίζουμε το $g^{x'}$ οπότε διακρίνουμε τις δύο τετραγωνικές ρίζες, και άρα τελικά τα δύο πρόβληματα είναι πολυωνυμικά ισοδύναμα.

Συνεπώς, αφού το πρόβλημα διακριτού λογάριθμου στο $\mathbb{Z}_p^*$ έχει υποτεθεί υπολογιστικά δύσκολο, τότε και το πρόβλημα διάκρισης τετραγωνικών ριζών είναι με βάση την ίδια υπόθεση υπολογιστικά δύσκολο. (αυτό δεν χρειάζεται την ανάποδη δυσκολία, μόνο την παραπάνω αναγωγή.)

6 Άσκηση 6η: Παραλλαγή Κρυπτοσυστήματος ElGamal

6.1 Περιγραφή συστήματος

Έστω μια ομάδα G με τάξη $q \in \mathbb{P}$.

Αρχικά επιλέγονται δύο ομοιομόρφα τυχαία στοιχεία $g, h \xleftarrow{R} G - \{e\}$. Τα στοιχεία αυτά είναι γεννήτορες της G, διότι η G έχει τάξη πρώτο αριθμό q. Υποθέτουμε ότι αυτά τα στοιχεία επιλέγονται εξαρχής με αυτόν τον τρόπο (trusted setup) και είναι γνωστά σε όλους.

Τότε αρχικά η $KGen$ ορίζεται ως εξής:

1. Επιλέγεται ομοιόμορφα τυχαίο στοιχείο $x \xleftarrow{R} \mathbb{Z}_q^*$.
2. Επιστρέφεται το ζεύγος $(sk, pk) = (x, g^x)$. (sk: secret key, pk: public key)

Η συνάρτηση κρυπτογράφησης ορίζεται ως:

1. Επιλέγεται ομοιόμορφα τυχαίο στοιχείο $r \xleftarrow{R} \mathbb{Z}_q^*$.
2. Επιστρέφεται το:
$$Enc(pk, m) = (pk^r, g^r h^m)$$

6.2 Συνάρτηση αποκρυπτογράφησης

6.2.1 Αποκρυπτογράφηση με επίλυση διακριτού λογάριθμου

Έστω ότι διαθέτουμε ένα μαντείο $DLog^{G,g}(v)$ που επιστέφει x ώστε $g^x = v$.

Τότε ορίζεται η συνάρτηση αποκρυπτογράφησης $Dec\left(pk^r, g^r h^m\right) = m$ ως εξής.

Υπολογίζω το $\left(pk^r\right)^x = g^{xr}$.

Έπειτα βρίσκω το $g^{-xr} = \left(g^{xr}\right)^{q-1}$ (με επαναλαμβανόμενο τετραγωνισμό).

Υπολογίζω επίσης το; $\left(g^r h^m\right)^x = g^{xr} h^{mx}$.

Τέλος υπολογίζω $g^{xr} h^{mx} g^{-xr} = h^{mx}$.

Καλώ το $DLog^{G,h}\left(h^{mx}\right)$, επιστρέφεται το mx.

Βρίσκω το $x^{-1} \pmod q$ με εκτεταμένο αλγόριθμο Ευκλείδη.

Τέλος υπολογίζω $mx \cdot x^{-1} \equiv m \pmod q$ και επιστρέφω το m.

Θεωρώντας το $DLog^{G,g}(v)$ ως μαντείο ο παραπάνω αλγόριθμος είναι πολυωνυμικός.

6.2.2 Υπολογιστική ισοδυναμία της αποκρυπτογράφησης γνωρίζοντας το ιδιωτικό κλειδί με τον Διακριτό Λογάριθμο

Σε πρώτη ανάλυση, η παραπάνω συνάρτηση αποκρυπτογράφησης φαίνεται προβληματική, διότι ο διάκριτος λογάριθμος (σε αρκετές ομάδες) έχει υποτεθεί ένα υπολογιστικό δύσκολο πρόβλημα.

Ωστόσο, θα δείξουμε ότι η αποκρυπτόγραφηση αυτού του συστήματος με γνώση του ιδιωτικού κλειδιού είναι τουλάχιστον τόσο δύσκολη όσο ο υπολογισμός διακριτού λογάριθμου στην G.

Έστω ένα μαντείο $\mathcal{A}^G\left(g, h, x, g^{xr}, g^r h^m\right)$ που επιστρέφει το m, το οποίο δηλαδή λύνει το πρόβλημα αποκρυπτογράφησης.

Κατασκευάζω τότε έναν πολυωνυμικό αλγόριθμο για το πρόβλημα διακριτού λογάριθμου στην G, που δηλαδή με είσοδο h, h^m, επιστρέφει το m.

Επιλέγω ένα arbitrary στοιχείο $g \in G - \{e\}$ και δύο arbitrary φυσικούς αριθμούς $r, x \in \mathbb{Z}_q^*$.

Υπολογίζω τα g^{xr}, $g^r h^m$ και καλώ το $\mathcal{A}^G\left(g, h, x, g^{xr}, g^r h^m\right)$, που μου επιστρέφει το m, το οποίο και επιστρέφω.

Συνεπώς το πρόβλημα αποκρυπτογράφησης είναι τουλάχιστον τόσο δύσκολο όσο και το πρόβλημα του διακριτού λογάριθμου G, οπότε η παραπάνω συνάρτηση αποκρυπτογράφησης δεν μπορεί να βελτιωθεί (με αυτό εννοούμε ότι αν το $DLog$ δεν υπολογίζεται πολυωνυμικά, ούτε και η αποκρυπτογράφηση υπολογίζεται πολυωνυμικά).

6.2.3 Υπολογιστική ισοδυναμία της αποκρυπτογράφησης γνωρίζοντας το ιδιωτικό κλειδί με τον Διακριτό Λογάριθμο – ακόμα και με εκ των προτέρων γνώση του $DLog(g, h)$

Έστω ότι $h = g^k$.

Θα δείξουμε ότι ακόμα και αν ο αλγόριθμος αποκρυπτογράφησης γνωρίζει το k, το πρόβλημα αποκρυπτογράφησης παραμένει τουλάχιστον τόσο δύσκολο όσο το πρόβλημα διακριτού λογάριθμου.

Έστω ένα μαντείο $\mathcal{A}^G\left(g, h, k, x, g^{xr}, g^r h^m\right)$ που επιστρέφει το m, το οποίο δηλαδή λύνει το πρόβλημα αποκρυπτογράφησης. Για την είσοδο πρέπει να ισχύει $h = g^k$, αλλιώς ο αλγόριθμος αποτυγχάνει.

Κατασκευάζω τότε έναν πολυωνυμικό αλγόριθμο για το πρόβλημα διακριτού λογάριθμου στην G, που δηλαδή με είσοδο g, g^m, επιστρέφει το m.

Επιλέγω arbitrary $k, r, x \in \mathbb{Z}_q$.

Υπολογίζω $h = g^k$, g^{xr} και $g^r h^m = g^r \left(g^m\right)^k$.

Καλώ το μαντείο $\mathcal{A}^G\left(g, h, k, x, g^{xr}, g^r h^m\right)$ και μου επιστρέφει το m, το οποίο και επιστρέφω.

Συνεπώς το πρόβλημα αποκρυπτογράφησης είναι τουλάχιστον τόσο δύσκολο όσο και το πρόβλημα του διακριτού λογάριθμου G, ακόμα και αν γνωρίζαμε τον διακριτό λογάριθμο του h με γεννήτορα g.

6.2.4 Αποκρυπτογράφηση με διακριτό λογάριθμο χωρίς το ιδιωτικό κλειδί

Έστω ότι διαθέτουμε ένα μαντείο $DLog^{G,g}(v)$ που επιστέφει x ώστε $g^x = v$.

Κατασκευάζω τότε έναν πολυωνυμικό αλγόριθμο για το πρόβλημα αποκρυπτογράφησης χωρίς το ιδιωτικό κλειδί, δηλαδή με είσοδο το δημόσιο κλειδί g^x και το κρυπτοκείμενο επιστρέφει το μήνυμα m.

Υπολογίζω το $DLog^{G,g}(pk) = DLog^{G,g}(g^x)$, επιστρέφεται το x, πλέον διαθέτω το ιδιωτικό κλειδί και άρα εκτελώ τον αλγόριθμο αποκρυπτογράφησης με χρήση του ιδιωτικού κλειδιού και του μαντείου $DLog$ που περιγράφηκε παραπάνω.

6.2.5 Σχολιασμός

Προκειμένου το κρυπτοσυστήμα να είναι ένα αποδεκτό κρυπτοσύστημα, πρέπει η αποκρυπτογράφηση να μπορεί να γίνει. Αυτό σημαίνει ότι πρέπει να ισχύουν ένα από τα εξής:

1. Το πρόβλημα διακριτού λογάριθμου, συγκεκριμένα για την ομάδα G, λύνεται αποδοτικά (σε πολυωνυμικό χρόνο). Τότε η αποκρυπτογράφηση πράγματι μπορεί να γίνει. Όμως τότε, μπορεί αποδοτικά και κάθε αντίπαλος να κάνει αποκρυπτογράφηση, όπως δείξαμε, οπότε το κρυπτοσύστημα δεν προσφέρει ασφάλεια.
2. Ο χρήστης που αποκρυπτογραφεί θεωρείται ότι δεν είναι πολυωνυμικά φραγμένος, δηλαδή δεχόμαστε ότι μπορεί να λύσει προβλήματα που παίρνουν παραπάνω από πολυωνυμικό χρόνο, οπότε όντως μπορεί να λύσει το πρόβλημα διακριτού λογάριθμου και άρα να κάνει αποκρυπτογράφηση.

   Τότε, όμως, ενδεχομένως να υπάρχει και κάποιος τέτοιος αντίπαλος που μπορεί να λύσει προβλήματα που απαιτούν παραπάνω από πολυωνυμικό χρόνο, οπότε και αυτός θα μπορέσει να σπάσει το σύστημα αποκρυπτογράφησης.

Με άλλα λόγια, η δυνατότητα αποκρυπτογράφησης και η ασφάλεια του συστήματος θεωρούνται ασυμβίβαστοι στόχοι.

6.3 Ασφάλεια

Στην παρούσα παράγραφο, θα μελετήσουμε ιδιότητες ασφάλειες με βάση την υπολογιστική δυσκολία προβλημάτων τύπου DLog, όπως ενδεικτικά το ίδιο το DLog, το DDH, κ.λπ. που θεωρείται ότι ισχύει σε κάποιες ομάδες.

Με βάση τα σχόλια των προηγούμενων παραγράφων, αφού θα δεχθούμε ότι το DLog είναι υπολογιστικά δύσκολο, οπότε η αποκρυπτογράφηση δεν μπορεί να γίνει αποδοτικά. Αν το DLog είναι εύκολο, τότε ούτως ή αλλως το σύστημα σπάει όπως είδαμε, οπότε δεν έχει εξαρχής νόημα η μελέτη της ασφάλειας του.

6.4 OW-CPA (One-Way under Chosen Plaintext Attack)

6.4.1 Ορισμός OW-CPA

Αρχικά ορίζουμε τι σημαίνει ότι ένα κρυπτοσύστημα δημόσιου κλειδιού $(KGen, Enc, Dec)$ προσφέρει ασφάλεια OW-CPA.

Έστω ένας PPT αντίπαλος $\mathcal{A}$. Εκτελούμε το εξής τυχαίο πείραμα:

1. $(pk, sk) \leftarrow KGen(1^n)$
2. $m \overset{R}{\leftarrow} \mathcal{M}$
3. $c \leftarrow Enc(pk, m)$
4. $m' \leftarrow \mathcal{A}(1^n, pk, c)$
5. Αν $m = m'$ επιστρέφω 1, αλλιώς 0.

Ένα κρυπτοσύστημα $(KGen, Enc, Dec)$ προσφέρει ασφάλεια OW-CPA, αν για κάθε PPT αντίπαλο $\mathcal{A}$ το αποτέλεσμα του παραπάνω τυχαίο πειράματος είναι 1 με πιθανότητα $negl(n)$.

6.4.2 Ιδιότητα OW-CPA στην παραλλαγή του ElGamal

Θα αποδείξουμε ότι αν κάθε PPT αντίπαλος λύνει το πρόβλημα διακριτού λογάριθμου με αμελητέα πιθανότητα, τότε το σύστημα προσφέρει OW-CPA ασφάλεια.

Αρχικά ορίζουμε την υπόθεση για το πρόβλημα διακριτού λογάριθμου.

Έστω μια κυκλική ομάδα G τάξης $q \in \mathbb{P}$, όπου $|q| = n$ (n bits) και g γεννήτορας. Έστω το εξής τυχαίο πείραμα για έναν PPT αντίπαλο $\mathcal{A}$:

1. $y \overset{R}{\leftarrow} G$
2. $x \leftarrow \mathcal{A}(g, y)$
3. Αν $g^x = y$ επιστρέφω 1, αλλιώς 0.

Η υπόθεση είναι ότι ισχύει στην ομάδα G πως για κάθε PPT αντίπαλο $\mathcal{A}$ το αποτέλεσμα του παραπάνω τυχαίου πείραματος είναι 1 με πιθανότητα $negl(n)$.

Προχώραμε στην απόδειξη OW-CPA ασφάλειας. Έστω ένας PPT αντίπαλος $\mathcal{A}$ που κερδίζει το τυχαίο πείραμα OW-CPA που αφορά στο μελετούμενο κρυπτοσύστημα με μη αμελητέα πιθανότητα.

Κατασκευάζω τότε έναν PPT αντίπαλο για το τυχαίο πείραμα του προβλήματος διακριτού λογάριθμου στην G, δηλαδή με είσοδο h^m υπολογίζει το m.

Επιλέγω ένα arbitrary στοιχείο $g \in G - \{e\}$ και ένα arbitrary φυσικό αριθμό $r \in \mathbb{Z}_q^*$. (μια απλή επιλογή είναι $g = h$ και $r = 1$, αλλά μπορούμε να επιλέξουμε οποιαδήποτε στοιχεία.)

Υπολογίζω το $(h^r, g^r \cdot h^m)$ και καλώ το $\mathcal{A}\left((g, h, g^x), (h^r, g^r h^m)\right)$. που μου επιστρέφει ένα m', το οποίο και επιστρέφω.

Αν ο $\mathcal{A}$ πετύχει τότε $m' = m$, οπότε ο αλγόριθμος βρίσκει πράγματι τον διακριτό λογάριθμο και έτσι στο τυχαίο πείραμα του διακριτού λογάριθμου η απάντηση είναι 1.

Αν το τυχαίο πείραμα OW-CPA δώσει 1 τότε ο $\mathcal{A}$ πετυχαίνει, που συμβαίνει συνεπώς με μη αμελητέα πιθανότητα.

Άρα τελικά έχουμε έναν PPT αλγόριθμο που δίνει στο τυχαίο πείραμα του διακριτού λογαριθμού 1 με μη αμελητέα πιθανότητα. Αυτό όμως είναι άτοπο. Συνεπώς το σύστημα προσφέρει ασφάλεια OW-CPA.

6.5 IND-CPA (Indistinguishable under Chosen Plaintext Attack)

Έστω ένας PPT αντίπαλος $\mathcal{A}$ που κερδίζει το τυχαίο πείραμα IND-CPA με πιθανότητα $f(n)$.

Κατασκευάζω έναν PPT αντίπαλο για το τυχαίο πείραμα του Decisional Diffie-Hellman, δηλαδή με είσοδο g, g^x, g^y, g^z εξετάζει αν ισχύει ότι $z = xy$.

Επιλέγω ένα arbitrary στοιχείο $h \leftarrow G - \{e\}$. Το h είναι γεννήτορας, αφού $\text{ord}\,(G) = q \in \mathbb{P}$.

Θεωρώ ότι $pk = (g, h, g^x)$. Χρησιμοποιώ ως μαντείο κρυπτογράφησης την γνωστή συνάρτηση κρυπτογράφησης του κρυπτοσυστήματος για το δημόσιο κλειδί pk.

Εκτελώ τον $\mathcal{A}$ με το μαντείο κρυπτογράφησης και το δημόσιο κλειδί pk και μου επιστρέφει δύο μηνύματα m_0, m_1.

Επιλέγω ομοιόμορφα τυχαίο $b \leftarrow \{0, 1\}$.

Υπολογίζω το $c = (g^z, g^y \cdot h^{m_b})$ και το στέλνω στον $\mathcal{A}$.

Τελικά μου επιστρέφει ένα b^*. Επιστρέφω 1 αν $b^* = b$, αλλιώς 0.

Έστω ότι $z = xy$. Τότε το c είναι μια έγκυρη κρυπτογράφηση του m_b.

Συνεπώς το παραπάνω τυχαίο πείραμα είναι μια πραγματοποίηση του IND-CPA τυχαίου πειράματος. Άρα θα προκύψει $b^* = b$ με μη αμελητέα πιθανότητα $f(n)$.

Έστω ότι το z επιλέχθηκε ομοιόμορφα τυχαία. Ο μόνος τρόπος που λαμβάνει ο $\mathcal{A}$ το b είναι μέσω του c.

Ωστόσο: $c = (g^z, g^y \cdot h^{m_b})$. Ο μόνος τρόπος που ο $\mathcal{A}$ λαμβάνει το g^y είναι μέσω του γινομένου $g^y \cdot h^{m_b}$. Το y είναι ομοιόμορφα τυχαίο και ανεξάρτητο κάθε άλλης τυχαίας μεταβλητής, συνεπώς το ίδιο ισχύει και για το g^y (ως μετάθεση) και συνεπώς και για το $g^y \cdot h^{m_b}$ (επίσης ως μετάθεση).

Συνεπώς όλη η είσοδος που λαμβάνει ο $\mathcal{A}$ είναι ανεξάρτητη του b.

Συνεπώς το b είναι ομοιόμορφα τυχαίο και ανεξάρτητο του b^*. Άρα $\mathbb{P}\left[b = b^*\right] = \frac{1}{2}$.

Τελικά για τον αλγόριθμο $\mathcal{A}'$ που περιγράψαμε:

$$\left|\mathbb{P}\left[\mathcal{A}'(g, g^x, g^y, g^z) = 1\right] - \mathbb{P}\left[\mathcal{A}'(g, g^x, g^y, g^{xy}) = 1\right]\right| = \left|f(n) - \frac{1}{2}\right|$$

Λόγω της υπόθεσης υπολογιστικής δυσκολίας του DDH πρέπει να ισχύει:

$$\left|f(n) - \frac{1}{2}\right| \leq negl(n) \Rightarrow f(n) \leq \frac{1}{2} + negl(n) \tag{6.1}$$

Το τελευταίο ισχύει για κάθε PPT αντίπαλο $\mathcal{A}$ για το IND-CPA.

Συνεπώς το κρυπτοσύστημα προσφέρει ασφάλεια IND-CPA.

6.6 IND-CCA (Indistinguishable under Chosen Claintext Attack)

Θα δείξουμε ότι το σύστημα δεν προσφέρει ασφάλεια IND-CCA.

Κατασκευάζω έναν PPT αντίπαλο που κερδίζει το τυχαίο πείραμα IND-CCA με μη αμελητέα πιθανότητα.

Αρχικά δίνονται ως είσοδος το δημόσιο κλειδί $pk = (g, h, g^x)$.

Κατασκευάζω δύο arbitrary μήνυματα $m_0, m_1 \in \mathbb{Z}_p^*$ (με $m_0 \neq m_1$) και τα στέλνω.

Λαμβάνω ένα κρυπτοκείμενο $c = (g^{xr}, g^r h^{m_b})$.

Επιλέγω μια ομοιόμορφα τυχαία τιμή $r' \leftarrow \mathbb{Z}_p^*$.

Υπολογίζω το $c' = \left(\left(g^{xr}\right)^{r'}, \left(g^r h^{m_b}\right)^{r'}\right) = (g^{xrr'}, g^{rr'} h^{m_b r'})$.

Το c' είναι ένα έγκυρο κρυπτοκείμενο για το μήνυμα $m_b r'$ (θεωρώντας ότι επιλέχθηκε η τυχαιότητα $\tilde{r} = rr'$).

Στέλνω το c' στο μαντείο αποκρυπτογράφησης. Επιστρέφεται το μήνυμα $m_b r'$.

Υπολογίζω το $r'^{-1} \pmod{q}$ (με χρήση του Εκτεταμένου Αλγόριθμου Ευκλείδη).

Υπολογίζω το $m_b r' \cdot r^{-1} \equiv m_b \pmod{q}$.

Έχοντας βρει το m_b, επιστρέφω 0 αν $m_b = m_0$ και 1 αν $m_b = m_1$.

Ο αντίπαλος αυτός κερδίζει το τυχαίο πείραμα IND-CCA πάντα, δηλαδή με πιθανότητα 1, που δεν είναι αμελητέα. Συνεπώς το κρυπτοσύστημα δεν προσφέρει ασφάλεια IND-CCA.

7 Άσκηση 7η: Σ-πρωτόκολλο βασισμένο στο σχήμα Pedersen

7.1 Ιδιότητα Σ-πρωτοκόλλου

7.1.1 Πληρότητα (Completeness)

Αν ο Prover εκτελεί τίμια το πρωτόκολλο, θα δείξουμε ότι ο Verifier αποδέχεται πάντα.

Ισχύει ότι:

$$g^{s_1}h^{s_2} = g^{t_1+em}h^{t_2+er} = g^{t_1}g^{em}h^{t_2}h^{er} = g^{t_1}h^{t_2}\left(g^m h^r\right)^e = Tc^e$$

Συνεπώς αν ο Prover είναι τίμιος, ο Verifier αποδέχεται.

7.1.2 Ειδική Ορθότητα (Special Soundness)

Ειδική Ορθότητα για ένα Σ-πρωτόκολλο σημαίνει ότι υπάρχει ένας PPT αλγόριθμος $\mathcal{E}$ (ονομάζεται knowledge extractor), ο οποίος με πρόσβαση σε δύο transcripts (com, c_1, r_1), (com, c_2, r_2) με $c_1 \neq c_2$ που γίνονται και τα δύο αποδεκτά, υπολογίζει τον witness w, που εδώ είναι η δυάδα (m, r).

Στην πραγματικότητα ο $\mathcal{E}$ κάνει "rewind" το μαντείο $\mathcal{P}$ στο σήμειο που μόλις του έδωσε το com και του δίνει ένα νέο $c_2 \neq c_1$ οπότε προκύπτει ένα νέο r_2.

Αρχικά, σημειώνεται ότι χρειάζεται να βρούμε έναν witness και όχι τον witness που χρησιμοποιεί ο prover.

Παρατηρούμε ότι για κάθε c ύπαρχει ένα m, το $m = DLog(g, c)$ ώστε το $(m, 0)$ να είναι ένας witness καθώς:

$$g^m h^r = g^{DLog(g,c)}h^0 = c$$

Από την άλλη ο witness της μορφής $(m, 0)$ είναι μοναδικός διότι:

$$g^m h^r = c \implies g^m h^0 = c \implies m = DLog(g, c)$$

Θα βρούμε τον witness τέτοιας μορφής. Σημειώνεται ότι δεν υποθέτουμε ότι χρησιμοποιήθηκε αυτός ο witness από τον Prover, αλλά απλά ότι το c έχει μια μοναδική τέτοια αναπαράσταση.

Συνεπώς έστω τα δύο transcripts $(t, e, (s_1, s_2))$ και $(t, e', (s_1', s_2'))$.

Τα δύο transcripts γίνονται αποδεκτά. Συνεπώς:

$$\left.\begin{array}{l} g^{s_1}h^{s_2} = tc^e = tg^{me}h^{0e} \implies g^{s_1-me} = t \\ g^{s_1'}h^{s_2'} = tc^{e'} = tg^{me'}h^{0e'} \implies g^{s_1'-me'} = t \end{array}\right\} \implies g^{s_1-me} = g^{s_1'-me'} \implies$$

$$s_1 - me \equiv s_1' - me' \pmod q \implies$$

$$(e-e')m \equiv s_1' - s_1 \pmod q \overset{e-e' \not\equiv 0 \pmod q,\ q\in\mathbb{P}}{\implies}$$

$$m \equiv (s_1' - s_1)(e-e')^{-1} \pmod q$$

Συνεπώς βρήκαμε τον witness $(m, r) = ((s_1' - s_1)(e-e')^{-1}, 0)$.

7.1.3 Μηδενική Γνώση για Τίμιους Επαληθευτές (Honest Verifier Zero Knowledge)

Κατσκευάζω τον Simulator S.

1. Επιλέγω ομοιόμορφα τυχαία $t_1, t_2 \overset{R}{\leftarrow} \mathbb{Z}_q^*$ και υπολογίζω το $t = g^{t_1} \cdot g^{h_2}$
2. Δίνω στον $\mathcal{V}$ το $t = g^{t_1} \cdot g^{h_2}$.
3. Ο $\mathcal{V}$ απαντά με ένα e.
4. Επιλέγω ομοιόμορφα τυχαία $s_1, s_2 \overset{R}{\leftarrow} \mathbb{Z}_q^*$ και υπολογίζω $t' = c^{-e} \cdot g^{s_1}h^{s_2}$.
5. Κάνω rewind τον $\mathcal{V}$ στην αρχή.
6. Δίνω στον $\mathcal{V}$ το t'.
7. Επειδή ο $\mathcal{V}$ είναι honest, βρίσκει το e από το random tape του, που είναι το ίδιο, οπότε απαντά και πάλι με e.

8. Του απαντώ (s_1, s_2).

Επειδή $t'c^e = c^{-e}g^{s_1}h^{s_2}c^e = g^{s_1}h^{s_2}$, το transcript μεταξύ S και $\mathcal{V}$ δίνει αποδεκτή απόδειξη.

Το transcript με τον τίμιο Prover είναι $(g^{t_1}h^{t_2}, e, (t_1 + er \mod q, t_2 + em \mod q))$, όπου t_1, t_2, e ομοιόμορφα τυχαία ανεξάρτητες τυχαίες μεταβλητές. Αυτό σημαίνει ότι τα e και τα $(t_1 + er, t_2 + em \mod q)$ είναι ομοιόμορφα τυχαία και μεταξύ τους ανεξάρτητα, ενώ το $g^{t_1}h^{t_2}$ προκύπτει (με πιθανότητα 1) ως εφαρμογή της συνάρτησης $f(a, (b_1, b_2)) = g^{b_1}h^{b_2} \cdot c^{-e}$ σε αυτά τα δύο:

$$f(e, (s_1, s_2)) = g^{s_1}h^{s_2}c^{-e} = g^{t_1+er}h^{t_2+em}g^{-me}h^{-re} = g^{t_1}h^{t_2}$$

Αντίστοιχα το transcript με τον Simulator είναι $(c^{-e}g^{s_1}h^{s_2}, e, (s_1, s_2))$. Και πάλι τα e, (s_1, s_2) είναι ομοιόμορφα τυχαία ανεξάρτητες τυχαίες μεταβλητές, ενώ ο πρώτος όρος $c^{-e}g^{s_1}h^{s_2}$ προκύπτει (με πιθανότητα 1) ως εφαρμογή της ίδια συνάρτηση f με παραπάνω στους δύο τελευταίους όρους:

$$f(e, (s_1, s_2)) = g^{s_1}h^{s_2}c^{-e}$$

Συνεπώς τα δύο transcripts έχουν ίδια κατανομή πιθανότητας.

Συνεπώς ο $\mathcal{S}$ είναι ένας έγκυρος Simulator, καθώς καταφέρνει να εκτελέσει οτιδήποτε υπολογισμό μπορεί να εκτέλεσει ένας κακόβουλος $\mathcal{V}$ χωρίς όμως να διαθέτει έναν witness w.

Συνεπώς το Σ-πρωτόκολλο διαθέτει την ιδιότητα Μηδενικής Γνώσης για Τίμιους Επαληθευτές (Honest Verifier Zero Knowledge).

7.2 Witness Indistinguishability

Θα δείξουμε ότι το Σ-πρωτόκολλο προσφέρει την ιδιότητα του Witness Indistinguishability για Τίμιους Επαληθευτές λόγω της Μηδενικής Γνώσης για Τίμιους Επαληθευτές.

Έστω ότι δεν την προσφέρει, δηλαδή ότι υπάρχει ένας PPT Verifier $\mathcal{V}^*$ που μπορεί από το transcript να διαχωρίσει τους witness μεταξύ τους με βάση κάποιο κριτήριο της επιλογής του (οποιοδήποτε και αν είναι αυτό).

Τότε χρησιμοποιώ τον Simulator S, του δίνω τον PPT $\mathcal{V}^*$ και καταφέρνει να εκτελέσει τον $\mathcal{V}^*$ με το ίδιο αποτέλεσμα (ίδια κατανομή πιθανότητας στο αποτέλεσμα ακριβέστερα) που επιτυγχάνει ο $\mathcal{V}^*$ όταν επικοινωνεί με έναν τίμιο $\mathcal{P}$.

Ωστόσο, ο S δεν διαθέτει καν witness, οπότε είναι άτοπο ότι ο $\mathcal{V}^*$ επιτυγχάνει να διακρίνει τους witnesses. (προκύπτει ίδια κατανομή πιθανότητας στο αποτέλεσμά του, οποιοδήποτε witness και αν έχει ένας τίμιος Prover)

7.3 Παραλλαγή πρωτοκόλλου

Αλλάζουμε το παραπάνω πρωτόκολλο Π, ώστε στο πρώτο βήμα ο Prover αντί να στέλνει το $t = g^{t_1}h^{t_2}$ στον Verifier, στέλνει χωριστά τις τιμές $a = g^{t_1}$, $b = h^{t_2}$.

Ο Verifier θεωρούμε ότι απλά υπολογίζει μόνος του στην αρχή $t = ab$ και στην συνέχεια ακολουθεί το πρωτόκολλο Π ως πριν, δηλαδή επιλέγει ένα ομοιόμορφα τυχαίο c, και στο τέλος ελέγχει αν $tc^e = g^{s_1}h^{s_2}$.

Η ειδική ορθότητα ισχύει στο Π′, καθώς ισχύει στο Π, αφού για όλο το πρωτόκολλο μπορούμε να αντιμετωπίζουμε το $t = ab$ ως είσοδο.

Από την άποψη, ώστοσο της μηδενικής γνώσης με τίμιο επαληθευτή, αυτή χάνεται.

Ο Verifier αποκτά γνώση για τους witnesses.

Για κάθε στοιχείο $c \in G$ υπάρχουν περισσότεροι των ενός witnesses (m, r). Πιο συγκεκριμένα για κάθε στοιχείο $m \mod q$ είναι $c = g^m h^r \iff h^r = cg^{-m} \iff r = DLog(h, cg^{-m})$, οπότε για κάθε $m \mod q$ ορίζεται και ένας μοναδικός witness.

Ένας Verifier μπορεί να διακρίνει αυτούς τους Witnesses, αποκτώντας έτσι "γνώση για αυτούς".

Πιο συγκεκριμένα, αν $r = 0$ είναι $s_2 \equiv t_2 \pmod q$, οπότε $g^{s_2} = g^{t_2} = b$.

Αντίστροφα αν $g^{s_2} = b$ τότε:

$$g^{s_2} = g^{t_2} \implies s_2 \equiv t_2 \pmod q \implies t_2 + re \equiv t_2 \pmod q \implies re \equiv 0 \pmod q \overset{e \in \mathbb{Z}_q^*}{\implies} r \equiv 0 \pmod q$$

Έτσι ένας Verifier $\mathcal{V}^*$ εκτελεί τίμια το πρωτόκολλο και στο τέλος ελέγχει αν $g^{s_2} = b$ οπότε γνωρίζει (με βεβαιότητα) αν $r = 0$ ή $r \neq 0$, παραβιάζοντας έτσι την μηδενική γνώση.

Η παραβίαση της μηδενικής γνώσης, προκύπτει επειδή χωρίς τον τίμιο Prover είναι αδύνατο να υπολογιστεί ποιον witness έχει, καθώς για το ίδιο c, όπως δείξαμε, υπάρχουν περισσότεροι του ενός witness.

Συνεπώς το Π' δεν είναι Σ-πρωτόκολλο.

8 Άσκηση 8η: Σ-πρωτόκολλο για RSA κρυπτοκείμενα

8.1 Πληρότητα (Completeness)

Για έναν τίμιο Prover, θα δείξουμε ότι ο Verifier αποδέχεται πάντα.

Ισχύει ότι:

$$hy^c \equiv t^e(m^e)^c \equiv t^e m^{ec} \equiv (tm^e)^c \equiv r^e \pmod{n}$$

Συνεπώς, για έναν τίμιο Prover, ο Verifier αποδέχεται πάντα.

8.2 Ειδική Ορθότητα (Special Soundness)

Κατασκευάζουμε τον PPT αλγόριθμο $\mathcal{E}$.

Έστω δύο transcripts (h, c, r), (h, c', r'), $c \neq c'$ που γίνονται αποδεκτά από τον $\mathcal{V}$. Με είσοδο αυτά, το κρυπτοκείμενο $y \equiv m^e \pmod{n}$ και το δημόσιο κλειδί (e, n) ο $\mathcal{E}$ θα υπολογίσει το m.

Έστω (χωρίς βλάβη της γενικότητας) ότι $c > c'$. Αν δεν ισχύει, τότε παρακάτω αντιμεταθέτω πρώτα τα transcripts.

Αφού τα transcripts γίνονται αποδεκτά ισχύει ότι:

$$\left.\begin{array}{rl} hy^c \equiv r^e \pmod{n} & \overset{y\in U(\mathbb{Z}_n)}{\Longrightarrow} h \equiv r^e y^{-c} \\ hy^{c'} \equiv r'^e \pmod{n} & \overset{y\in U(\mathbb{Z}_n)}{\Longrightarrow} h \equiv r'^e y^{-c'} \end{array}\right\} \implies r^e y^{-c} \equiv r'^e y^{-c'} \pmod{n} \implies$$

$$r^e m^{-ec} \equiv r'^e m^{-ec'} \implies$$

$$(rm^{-c})^e \equiv \left(r'm^{-c'}\right)^e \pmod{n} \overset{\exists e^{-1}}{\Longrightarrow}$$

$$rm^{-c} \equiv r'm^{-c'} \pmod{n} \overset{*}{\Longrightarrow}$$

$$rr'^{-1} \equiv m^{c-c'} \pmod{n}$$

$*$ Πρέπει να ισχύει $\gcd(r, n) = 1$ για αυτό το βήμα. Συνεπώς το υπολογίζω εξαρχής. Αποκλείεται $r' \equiv 0 \pmod{n}$, διότι τότε $hy^c \equiv r^e \equiv 0 \pmod{n} \overset{y\in U(\mathbb{Z}_n)}{\Longrightarrow} h \equiv 0 \pmod{n}$, που αποκλείεται διότι υποθέτουμε ότι στέλνεται $h \in \mathbb{Z}_n^*$ (αλλιώς η απόδειξη αποτυγχάνει.) Άρα $n \nmid r$. Συνεπώς το $\gcd(r', n) \neq 1$ είναι ένας παράγοντας του $n = pq$. Έχοντας παραγοντοποίηση του n, υπολογίζω το $d = e^{-1} \pmod{\phi(n)}$ και επιστρέφω $m \equiv c^d \pmod{n}$. Αν λοιπόν δεν ισχύει αυτό συνεχίζω το με το $*$ και τα ακόλουθα.

Το $r'^{-1} \pmod{n}$ υπολογίζεται αποδοτικά με τον εκτεταμένο αλγόριθμο Ευκλείδη.

Επίσης, αφού $0 \leq c' < c \leq e - 1$ είναι $0 < c - c' \leq e - 1$. Συνεπώς αφού $e \in \mathbb{P}$ τότε $\gcd(c - c', e) = 1$.

Υπολογίζω το $k \leftarrow c - c'$.

Υπολογίζω και το $u \leftarrow rr^{-1} \pmod{n}$.

Με βάση τα προηγούμενα ισχύει ότι $u \equiv m^k \pmod{n}$.

Ισχύει επίσης ότι $c \equiv m^e \pmod{n}$.

Εκτελώ τον Εκτεταμένο Αλγόριθμο Ευκλείδη για τους αριθμούς e, k και επιστρέφονται οι συντελεστές Bézout (E, K) ώστε να ισχύει η ταυτότητα Bézout:

$$eE + kK = \gcd(e, k) = \gcd(e, c - c') = 1$$

Υπολογίζω τότε:

$$u^K c^E \equiv m^{kK} m^{eE} \equiv m^{kK+eE} \equiv m^1 \equiv m \pmod{n}$$

Συνοψίζοντας:

1. Υπολογίζω r^{-1} με Εκτεταμένο Αλγόριθμο Ευκλείδη και έπειτα $u \leftarrow rr^{-1}$
2. Εκτελώ Εκτεταμένο Αλγόριθμο Ευκλείδη για τους αριθμούς e, $k = c - c'$, και επιστρέφονται οι (E, K).
3. Υπολογίζω $u^K c^E \equiv m \pmod{n}$ και το επιστρέφω.

Συνεπώς το Σ-πρωτόκολλο ικανοποιεί την ιδιότητα της Ειδικής Ορθότητας (Special Soundness).

8.3 Μηδενική Γνώση για Τίμιους Επαληθευτές (Honest Verifier Zero Knowledge)

Κατασκευάζω τον Simulator S (που τρέχει κάθε τίμιο Verifier $\mathcal{V}$ υπολογίζοντας το ίδιο αποτέλεσμα – με ίδια κατανομή πιθανότητας – χωρίς όμως την γνώση ενός witness).

1. Επιλέγω ομοιόμορφα τυχαίο $t \overset{R}{\leftarrow} \mathbb{Z}_n^*$ και στέλνω το $t^e \pmod n$ στον $\mathcal{V}$.
2. Ο $\mathcal{V}$ απαντά με c.
3. Επιλέγω ομοιόμορφα τυχαίο $r \overset{R}{\leftarrow} \mathbb{Z}_N^*$. Υπολογίζω $h' \equiv r^e y^{-c} \pmod n$.
4. Κάνω "rewind" τον $\mathcal{V}$ στην αρχή.
5. Δίνω στον $\mathcal{V}$ το h'.
6. Ο $\mathcal{V}$ θεωρείται τίμιος, οπότε θα μας δώσει ένα c από το random tape του που είναι το ίδιο με προηγουμένως.
7. Του απαντάμε τέλος το r.

Ισχύει ότι $h'y^c \equiv r^e y^{-c} y^c \equiv r^e \pmod n$ οπότε το transcript είναι αποδεκτό.

Το transcript με έναν τίμιο Prover θα ήταν (t^e, c, tm^c) όπου τα t, c ομοιόμορφα τυχαίες ανεξάρτητες τυχαίες μεταβλητές των $\mathbb{Z}_n^*$ και $\mathbb{Z}_e$ αντίστοιχα.

Επειδή ορίζεται το $e^{-1} \pmod n$ είναι $(t^e)^{e^{-1}} \equiv t \pmod n$ οπότε το $t \mapsto t^e$ είναι μετάθεση του $\mathbb{Z}_n^*$. Συνεπώς και το t^e είναι ομοιόμορφα τυχαίο στο $\mathbb{Z}_n^*$. Συνεπώς οι $t^e \pmod n$, c είναι ομοιόμορφα τυχαίες ανεξάρτητες τυχαίες μεταβλητές των $\mathbb{Z}_n^*$ και $\mathbb{Z}_e$ αντίστοιχα.

Ο τρίτος όρος $tm^c = (t^e)^{e^{-1}} m^c$ προκύπτει με εφαρμογή της συνάρτησης $f(a,b) = a^{e^{-1}} m^b$ στους δύο πρώτους όρους: $f(t^e, c) = tm^c$, προκύπτει δηλαδή με πιθανότητα 1 από τους πρώτους δύο όρους.

Η ανεξαρτησία παραπάνω νοείται ως ανεξαρτησία μεταξύ τους αλλά και με κάθε άλλη τυχαία μεταβλητή.

Θα δείξουμε ότι την ίδια κατανομή πιθανότητα έχει και το transcript με τον Simulator, που είναι $(r^e y^{-c}, c, r)$ όπου τα r, c ομοιόμορφα τυχαίες ανεξάρτητες τυχαίες μεταβλητές των $\mathbb{Z}_n^*$ και $\mathbb{Z}_e$ αντίστοιχα.

Για τον ίδιο λόγο με παραπάνω το $r \mapsto r^e$ είναι μετάθεση του $\mathbb{Z}_n^*$. Επίσης για κάθε $x \in \mathbb{Z}_n^*$ είναι $(xy^{-c}) \cdot y^c \equiv x \pmod n$ οπότε το $x \mapsto xy^{-c}$ είναι επίσης μετάθεση $\mathbb{Z}_n^*$. Έτσι, τελικά, το $r \mapsto h' = r^e y^{-c}$ είναι μετάθεση του $\mathbb{Z}_n^*$.

Συνεπώς, και εδώ οι δύο πρώτοι όροι $r^e y^{-c}$, c είναι ομοιόμορφα τυχαίες ανεξάρτητες τυχαίες μεταβλητές των $\mathbb{Z}_n^*$ και $\mathbb{Z}_e$ αντίστοιχα. Ο τρίτος όρος c προκύπτει ως εφαρμογή της ίδια συνάρτησης $f(a,b)$ με παραπάνω στους δύο πρώτους όρους:

$$f(r^e y^{-c}, c) \equiv (r^e y^{-c})^{e^{-1}} m^c \equiv r^{ee^{-1}} m^{-ece^{-1}} m^c \equiv r^1 m^{-c} m^c \equiv r^1 m^0 \equiv r \pmod n$$

(δεν έχει σημασία ο υπολογισμός, απλά ότι με δεσμευμένους τους πρώτους δύο προκύπτει με πιθανότητα 1 ο τρίτος όρος ως ίδια συνάρτηση των πρώτων δύο όρων.)

Συνεπώς τα δύο transcripts ακολουθούν την ίδια κατανομή πιθανότητας.

Συνεπώς το Σ-πρωτόκολλο ικανοποιεί την ιδιότητα Μηδενικής Γνώσης για Τίμιους Επαληθευτές (Honest Verifier Zero Knowledge).

9 Άσκηση 9η: Non-Interactive Schnorr και παραλλαγή στον υπολογισμό του Hash

9.1 Περιγραφή πρωτοκόλλου

Σκοπός είναι η απόδειξη γνώσης ενός διακριτού λογάριθμου x ως προς ένας γεννήτορας g δημόσιου στοιχείου y, δηλαδή γνώσης στοιχείου x ώστε $g^x = y$.

Ο τίμιος Prover λειτουργεί ως εξής:

1. Επιλέγει ένα ομοιόμορφα τυχαίο $t \xleftarrow{R} \mathbb{Z}_q$.
2. Υπολογίζει το $T = g^t$.
3. Υπολογίζει το $c = H(T)$.
4. Υπολογίζει το $s = t + cx \mod q$.
5. Δίνει το (T, c, s) στον Verifier.

Ο Verifier ελέγχει αν $c = H(T)$ και αποδέχεται αν $TY^c = g^s$.

Σημείωση: Ο Verifier θα μπορούσε να ελέγξει αν $c = H(g^s Y^{-c})$ παραλείποντας δηλαδή το T (που θα μπορούσε έτσι να μην μεταδοθεί). Αν ισχύει $c = H(T)$ και $TY^c = g^s$ τότε ισχύει και $c = H(g^s Y^{-c})$. Αντίστροφα έστω ότι $c = H(g^s H^{-c})$, αλλά $TY^c \neq g^s \implies T \neq g^s Y^{-c}$. Τότε $H(T) = c = H(g^s Y^{-c})$ αλλά $T \neq g^s H^{-c}$, οπότε θα είχε βρεθεί μια σύγκρουση. Ο malicious Prover θα είχε βρει μια σύγκρουση για την H η οποία μάλιστα για $c \neq 0$ είναι σε μήνυμα επιλογής εξαρτώμενο από το δημόσιο κλειδί Y. Αυτό έχει υποτεθεί πως γίνεται μόνο με αμελητέα πιθανότητα. Συνεπώς αν γίνει ο απλοποιημένος έλεγχος, ο Verifier έχει μόλις αμελητέα μεγαλύτερη πιθανότητα να πειστεί από malicious Prover, που θεωρείται αποδεκτό. Για $c = 0$ απαιτείται σύγκρουση μη εξαρτώμενη από το δημόσιο κλειδί, αλλά ούτως ή άλλως όπως θα δούμε ακόμα και αν δεν κάνουμε την βελτιστοποίηση με την παράλειψη του T πάλι προκύπτει αυτό το πρόβλημα. Για λόγους απλότητας και σαφήνειας, αφού οι έλεγχοι είναι (υπολογιστικά) ισοδύναμοι, θα μελετήσουμε τον έλεγχο που μεταδίδει και το T, οπότε ο Verifier ελέγχει $c = H(T) \wedge TY^c = g^s$.

Το πρωτόκολλο είναι πλήρες (complete) διότι για τίμιο Prover $g^s = g^{t+cx} = g^t \left(g^x\right)^c = TY^c$, οπότε η απόδειξη γίνεται δεκτή.

9.2 Ορθότητα

Το πρωτόκολλο πράγματι προσφέρει ορθότητα στο μοντέλο του Τυχαίου Μαντείου. (παρακάτω θα δείξουμε ότι είναι εφικτή μια κάπως διαφορετική επίθεση – που είναι ρεαλιστική ωστόσο)

Κατασκευάζουμε ένα PPT αλγόριθμο $\mathcal{E}^{\mathcal{P}^*}$ ο οποίος με πρόσβαση σε κάθε PPT αλγόριθμο $\mathcal{P}^*$ που παράγει αποδεκτές αποδείξεις (T, c) μπορεί να υπολογίσει με αμελητέα πιθανότητα τον διακριτό λογάριθμο x. Με αυτή την κατασκευή δείχνουμε ότι αν ο P δώσει μια αποδεκτή απόδειξη τότε γνωρίζει τον διακριτό λογάριθμο x, αλλιώς εκτελώντας των $\mathcal{E}$ θα είχαμε έναν PPT αλγόριθμο που υπολογίζει διακριτό λογάριθμο με μη αμελητέα πιθανότητα, που γενικά έχει υποτεθεί πως δεν είναι εφικτό.

Ο $\mathcal{E}$ λειτουργεί ως εξής:

Εκτελεί τον $\mathcal{P}^*$ και προσομοιώνει το τυχαίο μαντείο επιλέγοντας σε κάθε κλήση τυχαίες τιμές, σημειώνοντας τις, ώστε σε επόμενη κλήση να δίνει ίδια τιμή.

Ο $\mathcal{P}^*$ στο τέλος επιστρέφει μια απόδειξη (T, c, s) που γίνεται αποδεκτή, δηλαδή ισχύει $c = \mathrm{H(T)}$ και $TY^{H(T)} = g^s$.

Αν το c που επέστρεψε ο $\mathcal{P}^*$ καθορίζεται με οποιονδήποτε τρόπο πριν ρωτήσει για πρώτη φορά το $H(T)$ στο τυχαίο μαντείο (ή αν δεν το ρώτησε καν ποτέ αυτό), έχει αμελητέα πιθανότητα να ισχύει $c = H(T)$ που είναι άτοπο διότι παράγει αποδείξεις που πείθουν (και άρα για τις οποίες $c = H(T)$) με μη αμελητέα πιθανότητα. Συνεπώς ο $\mathcal{P}^*$ έχει καθορίσει οριστικά το T πριν ρωτήσει για πρώτη φορά το $H(T)$ το οποίο μάλιστα θα ρωτηθεί.

Έτσι ο $\mathcal{E}$, αφού τρέξει για πρώτη φορά τον $\mathcal{P}^*$, τον κάνει "rewind" ακριβώς στο σημείο που ρώτησε το $H(T)$. Από εκεί, επανεκκινεί τον $\mathcal{P}^*$ όμως δίνει νέες, ανεξάρτητες από πριν, τιμές σε ότι καινούριο ρωτηθεί στο μαντείο (συμπεριλαμβανόμενου του $H(T)$ που δεν έχει ρωτηθεί νωρίτερα αφού βρισκόμαστε στην πρώτη φορά που αυτό ρωτάται).

Στο τέλος λαμβάνουμε μια νέα απόδειξη (T, c', s').

Η απόδειξη αυτή γίνεται αποδεκτή. Συνεπώς $c' = H'(T)$. Τα $H(T), H'(T)$ είναι ομοιόμορφα τυχαία και ανεξάρτητα. Συνεπώς η πιθανότητα $c = c' \iff H(T) = H'(T)$ είναι αμελητέα, οπότε αν τύχει και συμβεί απλά δηλώνουμε αποτυχία.

Άρα έχουμε $c \neq c'$, $TY^c = g^s$ και $TY^{c'} = g^{s'}$, οπότε, κατά τα γνωστά, $g^s Y^{-c} = g^{s'} Y^{-c'} \implies s - cx \equiv s' - c'x \implies x \equiv (s - s')(c - c')^{-1} \pmod{q}$, όπου ο τελευταίος αντίστροφος ορίζεται διότι $q \in \mathbb{P}$ και $c - c' \not\equiv 0 \pmod{q}$.

Ο αλγόριθμος $\mathcal{E}$ επιτυγχάνει με μη αμελητέα πιθανότητα.

Συνεπώς τελικά, το πρωτόκολλο προσφέρει ορθότητα στο μοντέλο του τυχαίου μαντείου.

9.3 Μια πιο ισχυρή ορθότητα (η "αυξημένη ορθότητα")

Ένα κάπως ρεαλιστικό μοντέλο επίθεσης είναι να αφήσουμε τον P^* να κάνει κάποια προεργασία πριν του δώσουμε το στοιχείο για το οποίο θέλουμε να απόδειξη γνώση διακριτού λογάριθμου που δεν ξέρει. Θα επιτρέψουμε σε αυτήν την προεργασία να πάρει παραπάνω από πολυωνυμικό χρόνο, ωστόσο θα κρατήσουμε τον περιορισμό του πολυωνυμικού χρόνου αφού δει το στοιχείο. Περιοριζόυμε την μεταβίβαση πληροφορίας από την μία φάση στην άλλη σε πολυωνυμικό χώρο (διαφορετικά θα μπορούσαν απλά να έχουν προϋπολογιστεί τα πάντα για την 2η φάση). Προκειμένου να διατηρηθεί ο συμβολισμός και η ορολογία των Zero Knowledge Proofs θεωρούμε την πρώτη φάση απλά ως ένα μαντείο, ώστε να μην αλλάξουν οι συμβολισμοί της πολυπλοκότητας, κ.λπ.

Αυτό εκ πρώτης όψεως φαίνεται μια περίεργη υπόθεση. Αρχικά, είναι σίγουρο ότι αν έχουμε ασφάλεια σε αυτή την υπόθεση έχουμε και ασφάλεια έναντι πολυωνυμικά φραγμένων αντιπάλων, οπότε, για αρχή πρόκειται για μια ισχυροποίηση της ασφάλειας. Το μοντέλο αυτό είναι ρεαλιστικό, καθώς ένας κακόβουλος παίχτης μπορεί να προσπαθήσει να χρησιμοποιήσει πολύ υπολογιστική ισχύ με σκοπό να την ισομοιράσει σε πολλές επιθέσεις, οπότε να αποσβέσει το κόστος. Ενδεικτικά, αυτό θεωρείται μια ρεαλιστική επίθεση για τα hashes στους κωδικούς πρόσβασης, όπου χρησιμοποιούνταν τα γνωστά rainbow tables, οπότε φαίνεται ότι ίσως υπάρχει πράγματι κάποιο αντίκρυσμα στον πραγματικό κόσμο.

9.4 Ικανοποίηση "αυξημένης ορθότητας" υπό προϋποθέσεις

Υποθέτουμε για αρχή ότι οι απαντήσεις του τυχαίου μαντείου $H(x)$ είναι κωδικοποιημένες κατά τρόπο ώστε $H(x) \not\equiv 0 \pmod{q}$, δηλαδή περιοριζόνται στο σύνολο $\mathbb{Z}_q^*$ αντί του $\mathbb{Z}_q$. Τότε το πρωτόκολλο ικανοποιεί την παραπάνω "αυξημένη ορθότητα".

Αρχικά θα δείξουμε ότι ο c δεν ρωτά το $H(T)$ για πρώτη φορά εντός του μαντείου πρώτης φάσης, δηλαδή πριν μάθει το στοιχείο $y = g^x$, που θεωρείται μια ομοιόμορφη τυχαία μεταβλητή.

Θα δείξουμε ότι αν αυτό δεν ισχύει, τότε μη αμελητέα πιθανότητα με χρήση του $\mathcal{P}^*$ λύνουμε το πρόβλημα του διακριτού λογάριθμου, για ένα στοιχείο g^a.

Θεώρημα 9.1. *Το $H(T)$ ρωτάται για πρώτη φορά στην πρώτη φάση με αμελητέα πιθανότητα.*

Απόδειξη. Έστω λοιπόν ότι το $H(T)$ ρωτάται με μη αμελητέα πιθανότητα για πρώτη φορά στο μαντείο πρώτης φάσης, όποτε το T έχει καθοριστεί σε αυτό, δηλαδή πριν μαθευτεί το $y = g^x$. Τότε για όλες τις εισόδους με αμελητέα πιθανότητα προκύπτει το ίδιο T.

Εκτελώ τον $\mathcal{P}^*$ και του δίνω το g^b, όπου b μια ομοιόμορφα τυχαία τιμή. Επιστρέφει (T, c, s). Με μη αμελητέα πιθανότητα η απόδειξη είναι αποδεκτή, οπότε $c = H(T)$. Αν βρούμε $c \neq H(T)$, δηλώνουμε αποτυχία.

Κάνω "rewind" τον $\mathcal{P}^*$ ακριβώς αφού τελειώσει η πρώτη φάση. Από εκεί και πέρα του δίνω ανεξάρτητες νέες εισόδους στο τυχαίο μαντείο. Του δίνω το $g^a \cdot g^b$, και τελικά επιστρέφει (T', c', s'). Με μη αμελητέα πιθανότητα η απόδειξη είναι αποδεκτή.

Όπως εξηγήθηκε με μη αμελητέα πιθανότητα το T ρωτήθηκε στην πρώτη φάση οπότε $T = T'$ και $c = c' = H(T) = H(T')$.

Άρα έχουμε:

$$\left.\begin{array}{c} T\left(g^b\right)^c = g^s \\ T\left(g^a g^b\right)^c = g^{s'} \end{array}\right\} \implies s - bc \equiv s' - (a+b)c \pmod{q} \implies ac \equiv s' - s \pmod{q} \overset{c \not\equiv 0, q \in \mathbb{P}}{\implies}$$

$$a \equiv (s' - s)c^{-1} \pmod{q}$$

Συνεπώς με μη αμελητέα πιθανότητα βρήκαμε τον διακριτό λογάριθμο a, που έχει υποτεθεί αδύνατο, οπότε προκύπτει άτοπο. □

Αφού το $H(T)$ με μη αμελητέα πιθανότητα ρωτάται εκτός της πρώτης φάσης, τότε χρησιμοποιούμε κανονικά τον $\mathcal{E}$ από την απόδειξη ορθότητας από παραπάνω (για τους πλήρως πολυωνυμικά φραγμένους αντιπάλους).

Συνεπώς, υπό την προϋπόθεση ότι τα hashes δεν λαμβάνουν τις τιμές $H(x) \equiv 0 \pmod{q}$ το πρωτόκολλο προσφέρει την "αυξημένη ορθότητα".

9.5 Επίθεση στο μοντέλο της αυξημένης ορθότητας όταν δεν ισχύει η προϋπόθεση

Με βάση όσα έχουν προηγηθεί, διερώτομαστε αν ο περιορισμός $H(x) \not\equiv 0 \pmod{q}$ είναι όντως αναγκαίος, εκτός από επαρκής.

Όπως θα δούμε, αν επιτραπεί αυτό, τότε ένας Prover μπορεί να με κάποια μη πολυωνυμική προεργασία να αποδείξει αποδοτικά, ότι γνωρίζει λογάριθμους οποιουδήποτε στοιχείου του ζητηθεί.

Ο $\mathcal{P}^*$ δουλεύει ως εξής:

1. Στην πρώτη φάση, που δεν έχει το στοιχείο ακόμα, βρίσκει ένα στοιχείο t, ώστε $H\left(g^t\right) \equiv 0 \pmod{q}$. Αυτό απαιτεί, στο μοντέλο του τυχαίου μαντείου, χρόνο $O(|Q|) = O(2^\lambda)$ δηλαδή εκθετικό χρόνο. Ωστόσο, όπως είπαμε, σε αυτή την πρώτη φάση αυτό είναι αποδεκτό. Ένας κακόβουλος παίχτης μπορεί μια μόνο φορά να τρέξει αυτόν τον υπολογισμό και μετά να εκτελέσει πάρα πολλές (arbitrary) επιθέσεις με αποδοτικό τρόπο.
2. Στην δεύτερη φάση, δίνεται το $y = g^x$. Επιστρέφει τότε $(T, c, s) = (g^t, H(g^t), t)$.

Ισχύει ότι $TY^c = g^t y^{H(g^t)} = g^t y^0 = g^t = g^s$.

Συνεπώς ο Verifier αποδέχεται την απόδειξη, παρόλο που ο $\mathcal{P}^*$ δεν γνώριζε το $x = DLog(g, y)$.

Συμπερασματικά, κατασκευάσαμε μια επίθεση κατά του πρωτοκόλλου, και αποδείξαμε στο μοντέλο της "αυξημένης ορθότητας" που ορίσαμε, πως αυτή η επίθεση είναι η *"βέλτιστη"*, καθώς διαφορετικά το πρωτόκολλο προσφέρει αφενός την κλασική ορθότητα, αφετέρου δε την "αυξημένη ορθότητα" που ορίστηκε.

10 Άσκηση 10η: Προγραμματιστική Υλοποίηση Πρωτοκόλλου Schnorr

Προκειμένου να υλοποιήσουμε το πρωτόκολλο του Schnorr, χρειαζόμαστε μια ομάδα όπου το πρόβλημα διακριτού λογάριθμου να θεωρείται δύσκολο.

Για τον λόγο αυτό, χρησιμοποιήσαμε το σχετικό πρότυπο RFC8235: "*Schnorr Non-interactive Zero-Knowledge Proof*" το οποίο προτείνει την χρήση υποομάδας τάξης $q \in \mathbb{P}$ της ομάδας $\mathbb{Z}_p^*$, όπου $p \in \mathbb{P}$.

Σχετικά με την επιλογή των αριθμών p,q καθώς και σχετικά με την επιλογή του γεννήτορα g της ομάδας που θα εργαστούμε, το πρότυπο παραπέμπει στο πρότυπο NIST FIPS186-4: "*Digital Signature Standard (DSS)*".

Χρησιμοποιούμε έναν συνδυασμό από τους προτεινόμενους συνδυασμούς μεγεθών του NIST FIPS186-4, που είναι 2048 bits για τον αριθμό p και 256 bits για τον αριθμό q.

Για την γέννεση των αριθμών p,q χρησιμοποιούμε ιδέες από το παράρτημα A.1.1.2 του NIST FIPS186-4, ωστόσο για απλότητα, (με δεδομένο ότι ο κώδικας μας δεν θα χρησιμοποιηθεί σε πραγματική κρυπτογραφική εφαρμογή) χρησιμοποιήσαμε μια απλή γεννήτρια τυχαίων αριθμών αντί για την προτεινόμενη κατασκευή αριθμών με την Συνάρτηση Κατακερματισμού.

Ειδικότερα, για την επιλογή των p, q εκτελούμε τα εξής:

1. Επιλέγουμε έναν probable prime q μεγέθους $N = 256$ bits (με τυχαία γέννεση και έπειτα έλεγχο με κριτήριο Miller-Rabin – σε έτοιμη υλοποίηση από PyCryptodome).
2. Επιλέγουμε έναν τυχαίο αριθμό x μεγέθους $L = 2048$ bits. Υπολογίζουμε το $c \leftarrow x \mod 2q$ και θέτουμε $p \leftarrow x - (c - 1)$.
3. Αν ο p αποτελείται από λιγότερα από $L = 2048$ bits δηλώνουμε αποτυχία και προσπαθούμε ξανά από την αρχή.
4. Αν ο p δεν είναι πρώτος επίσης δηλώνουμε αποτυχία και προσπαθούμε ξανά από την αρχή.

Επειδή $q \mid 2q \mid (x - c)$, ισχύει ότι $p \equiv x - (c - 1) \equiv x - c + 1 \equiv 1 \pmod q$, οπότε $q \mid p - 1$, δηλαδή υπάρχει ένας $r \in \mathbb{N}$ ώστε $p - 1 = qr \Rightarrow p = qr + 1$.

Ο λόγος που επιλέγουμε $c \leftarrow x \mod 2q$ και όχι $c \leftarrow x \mod q$ είναι διότι αυτό είναι ισοδύναμο με το να είναι ο r άρτιος. Πράγματι αν ο r είναι περιττός τότε qr περιττός (αφού $q \in \mathbb{P}$), οπότε $p = qr + 1$ άρτιος, που δεν γίνεται αφού $p \in \mathbb{P}$ (και $p > 2$).

Για την εύρεση του γεννήτορα g χρησιμοποιούμε ιδέες από το παράρτημα A.2.1 του NIST FIPS186-4. Επιλέγουμε τυχαία στοιχεία $h \in \mathbb{Z}_p^* - \{1\}$, ελέγχουμε αν $h^{\frac{p-1}{q}} \not\equiv 1 \pmod p$ και αν ισχύει επιστρέφουμε το $h^{\frac{p-1}{q}} \pmod p$, αλλιώς επαναλαμβάνουμε. (το $h = 1$ θα απορριφθεί ούτως ή αλλως.)

Δείχνουμε την ορθότητα αυτού. Έστω ένας γεννήτορας u της $\mathbb{Z}_p^*$. Τότε υπάρχει $k \in \mathbb{Z}_p^*$ ώστε $u^k = h$. Το στοιχείο είναι ισότιμο με:

$$h^{\frac{p-1}{q}} \equiv h^{\frac{qr+1-1}{q}} \equiv h^r \equiv u^{kr}$$

και έχει τάξη:

$$\frac{p-1}{\gcd(kr, p-1)} = \frac{qr}{\gcd(kr, qr)} = \frac{qr}{r\gcd(k,q)} = \frac{q}{\gcd(k,q)}$$

Το $g = h^{\frac{p-1}{q}} \mod p$ είναι γεννήτορας της υποομάδας τάξης q, αν και μόνο αν έχει τάξη q, δηλαδή αν και μόνο αν:

$$\frac{q}{\gcd(k,q)} = q \iff \gcd(k,q) = 1$$

Αν $h^{\frac{p-1}{q}} \not\equiv 1 \pmod p$ τότε:

$$1 \not\equiv h^{\frac{p-1}{q}} \not\equiv u^{kr} \pmod p \implies p - 1 \nmid kr \implies qr \nmid kr \implies q \nmid k$$

Τότε αφού $q \in \mathbb{P}$ είναι $\gcd(k,q) = 1$ οπότε το g είναι γεννήτορας υποομάδας τάξης q.

Θα μπορούσε να είχε γίνει απλούστερη απόδειξη, ωστόσο θέλαμε το ισοδύναμο κριτήριο $\gcd(k,q) = 1$. Επιλέγοντας ομοιόμορφα τυχαία h, ισοδύναμα επιλέγονται οποιόμορφα τυχαία $k = DLog(g,h)$ (ως μια μετάθεση). Ο ελέγχος επιτυγχάνει ισοδύναμα όταν βρεθεί $\gcd(k,q) = 1$ δηλαδή όταν επιλέγει ένας τυχαίος σχετικά πρώτος με τον πρώτο q, που είναι πλήθους $q - 1$.

Συνεπώς κάθε επανάληψη του αλγόριθμου επιλογής g επιτυγχάνει με πιθανότητα $\frac{q-1}{q} \geq 1 - \frac{1}{2^{N-1}}$ οπότε πρακτικά με 1 επανάληψη θα επιτύχει απευθείας.

Στον κώδικα γίνονται τυχαιοποιημένοι έλεγχοι για τυχαία μηνύματα, πως παράγονται υπογραφές που γίνονται verify. Πράγματι, βρίσκουμε ότι επιτυγχάνει σε όλους τους τυχαίους ελέγχους που έγιναν. Για λόγους επίδοσης, εκτελούμε πολλά tests με τους ίδιους αριθμούς p, q, g καθώς αυτοί απαιτούν αρκετό χρόνο (αρκετά δευτερόλεπτα) για τον υπολογισμό τους.

Παραθέτουμε τον κώδικα Python:

```python
import Crypto.Random.random
import hashlib
import tqdm

# ideas from RFC 8235 (for Schnorr protocol) and NIST FIPS186-4 (for the group generation)

MESSAGE_LENGTH = 16384  # arbitrary size
NUM_TESTS = 100

# suggested values from NIST FIPS186-4 paragraph 4.2
N = 256
L = 2048


def groupelem_bytes(x):
    return x.to_bytes(length=L // 8, byteorder="little")


def schnorr_prove(p, q, g, x, gx, m):
    t = Crypto.Random.random.randint(0, q - 1)  # Z_q
    gt = pow(g, t, p)

    c = hashlib.sha512(
        groupelem_bytes(g) + groupelem_bytes(gx) + groupelem_bytes(gt) + m
    ).digest()
    c = int.from_bytes(c, byteorder="little")
    c = c % p

    s = (t + c * x) % p

    return c, s


def schnorr_verify(p, q, g, gx, m, sig):
    (c, s) = sig

    gt = (pow(g, s, p) * pow(gx, -c, p)) % p

    c1 = hashlib.sha512(
        groupelem_bytes(g) + groupelem_bytes(gx) + groupelem_bytes(gt) + m
    ).digest()
    c1 = int.from_bytes(c1, byteorder="little")
    c1 = c1 % p

    return c == c1


def get_schnorr_primes(n, l):
    # ideas from NIST FIPS186-4 appendix A.1
    while True:
        q = Crypto.Util.number.getPrime(n)

```

```
        x = Crypto.Util.number.getRandomNBitInteger(l)
        p = x - (x % (2 * q) - 1)

        if p.bit_length() < l:
            continue
        if Crypto.Util.number.isPrime(p):
            return (p, q)


def get_schnorr_g(p, q):
    # ideas from NIST FIPS186-4 appendix A.2.1
    e = (p - 1) // q  # p == q*r+1
    while True:
        h = Crypto.Random.random.randint(2, p - 2)  # Z_q
        g = pow(h, e, p)
        if g != 1:
            return g


def test(p, q, g):
    for _ in tqdm.tqdm(range(NUM_TESTS)):
        x = Crypto.Random.random.randint(0, q - 1)
        gx = pow(g, x, p)

        m = Crypto.Random.get_random_bytes(MESSAGE_LENGTH)

        sig = schnorr_prove(p, q, g, x, gx, m)

        status = schnorr_verify(p, q, g, gx, m, sig)

        assert status == True


if __name__ == "__main__":
    print("Generating primes p, q...")
    p, q = get_schnorr_primes(N, L)

    print("Genrating generator g...")
    g = get_schnorr_g(p, q)

    print(f"Random testing ({NUM_TESTS} tests) for random private keys and messages...")
    test(p, q, g)

    print("TESTS OK")
```

11 Άσκηση 11η: Forgeable παραλλαγή σχήματος υπογραφών ElGamal

Θα κατασκευάσουμε έναν πολυωνυμικό αλγόριθμο $\mathcal{A}$ που με είσοδο το δήμοσιο κλειδί (g, y), την δημόσια παράμετρο p, και ένα μήνυμα m παράγει μια ψηφιακή υπογραφή που γίνεται δεκτή.

Μια ψηφιακή υπογραφή είναι μια τριάδα (m, a, b). Έστω ότι η τριάδα αυτή γίνεται δεκτή (στο τέλος θα επαληθεύσουμε). Τότε:

$$yb \equiv g^a \pmod{p} g^{H(m)} yb \equiv 1 \pmod{p}$$

Είναι:

$$g^0 \equiv 1 \equiv g^{H(m)} yb \equiv g^{H(m)} g^a \equiv g^{H(m)+a} \pmod{p} \implies 0 \equiv H(m)+a \pmod{\phi(p)} \implies a \equiv -H(m) \pmod{p-1}$$

Τότε:

$$yb \equiv g^a \equiv g^{-H(m)} \pmod{p} \implies b \equiv g^{-H(m)} y^{-1} \pmod{p-1}$$

Συνεπώς αν υπάρχει έστω και μια αποδεκτή υπογραφή για το m με δημόσιο κλειδί y (που υπάρχει, διότι αν δεν υπάρχει τότε ο αλγόριθμος υπογραφών δεν μπορεί να είναι ορθός), αυτή η υπογραφή είναι ακριβώς η $(a, b) = (-H(m) \mod p-1, g^{-H(m)} y^{-1})$.

Επειδή δεν έχουμε δείξει απόδειξη ορθότητας, θα δείξουμε ότι αυτή η υπόγραφή επαληθεύεται.

Ισχύουν για αυτή την υπογραφή ότι:

$$yb \equiv y g^{-H(m)} y^{-1} \equiv g^{-H(m)} \equiv g^a \pmod{p}$$
$$g^{H(m)} yb \equiv g^{H(m)} y g^{-H(m)} y^{-1} \equiv 1 \pmod{p}$$

Συνεπώς η υπογραφή επαληθεύεται.

Οι τιμές $-H(m) \mod p-1$, $g^{-H(m)} y^{-1}$ υπολογίζονται σε πολυωνυμικό χρόνο. (για την 2η τιμή με Εκτεταμένο Αλγόριθμο Ευκλειδή).

Συνεπώς ο $\mathcal{A}$ υπολογίζει αυτές τις δύο τιμές και επιστρέφει $(a, b) = (-H(m) \mod p-1, g^{-H(m)} y^{-1})$.

Ο $\mathcal{A}$ είναι πολυωνυμικός και επιτυγχάνει πάντα.

Συνεπώς το σχήμα υπογραφών δεν προστατεύει από επίθεση καθολικής πλαστογράφησης. Μάλιστα, δεν χρειάζεται καν χρήση, ούτε μια φορά, του Signing Oracle.