Υπολογιστική Κρυπτογραφία
Τρίτη σειρά ασκήσεων

Μέη Αρετή, A.M.:03120062

January 28, 2025

1 Άσκηση 1

Ο διευθυντής σύντομα αντιλαμβάνεται ότι κάτι δεν πάει καλά διότι τα αποκρυπτογραφημένα μηνύματα που λαμβάνει είναι λάθος. Αυτό συμβαίνει λόγω του εργοστασιακού λάθους στη συσκευή, όπου υπολογίζεται $c'_q = m^{e+1} \mod q$ αντί για $c_q = m^e \mod q$.

Η γραμματέας, γνωρίζοντας τη δυσλειτουργία της συσκευής και έχοντας παρακολουθήσει μαθήματα κρυπτογραφίας, καταφέρνει να υπολογίσει το ιδιωτικό κλειδί του διευθυντή ως εξής:

1. Η συσκευή παράγει c' που ικανοποιεί:

$$c' \equiv m^e \mod p$$

και

$$c' \equiv m^{e+1} \mod q.$$

2. Έχοντας στη διάθεσή της το c' και γνωρίζοντας το δημόσιο κλειδί (n, e), η γραμματέας μπορεί να δοκιμάσει διαφορετικές τιμές του m (δοκιμαστικά μηνύματα) και να παρατηρήσει τα αποτελέσματα.

3. Για δύο διαφορετικά μηνύματα m_1 και m_2, η συσκευή παράγει:

$$c'_1 \equiv m_1^e \mod p \quad \text{και} \quad c'_1 \equiv m_1^{e+1} \mod q,$$

$$c'_2 \equiv m_2^e \mod p \quad \text{και} \quad c'_2 \equiv m_2^{e+1} \mod q.$$

4. Λαμβάνοντας υπόψη τη διαφορά των εξισώσεων (για παράδειγμα $c'_1 - c'_2$), μπορεί να απομονώσει πληροφορίες για τα p και q λόγω της ασυμβατότητας στις εκθέσεις e και $e + 1$.

5. Ειδικότερα, η γραμματέας μπορεί να υπολογίσει τον μέγιστο κοινό διαιρέτη (GCD) των διαφορών, π.χ., $GCD(c'_1 - m_1^e, n)$, για να ανακτήσει τον έναν από τους πρώτους p ή q.

6. Γνωρίζοντας έναν από τους δύο πρώτους, π.χ., p, μπορεί να υπολογίσει τον άλλον ως $q = \frac{n}{p}$.

7. Με τους δύο πρώτους αριθμούς p και q στη διάθεσή της, μπορεί να υπολογίσει τον εκθέτη αποκρυπτογράφησης d, που ικανοποιεί:

$$d \cdot e \equiv 1 \mod \varphi(n),$$

όπου $\varphi(n) = (p-1)(q-1)$.

Έτσι, η γραμματέας ανακτά το ιδιωτικό κλειδί (p, q, d) και μπορεί να αποκρυπτογραφήσει οποιοδήποτε μήνυμα χωρίς τη συσκευή.

2 Άσκηση 2

Για να δείξουμε ότι αν το πρόβλημα RSA είναι δύσκολο, τότε η H_s είναι collision-resistant, θα χρησιμοποιήσουμε την υπόθεση ότι αν υπάρχει μια αποδοτική μέθοδος για να βρεθούν $u, v \in \{0,1\}^{3n}$ με $u \neq v$ και $H_s(u) = H_s(v)$, τότε μπορούμε να λύσουμε το πρόβλημα RSA, κάτι που έρχεται σε αντίθεση με την υπόθεση ότι το RSA είναι δύσκολο.

Απόδειξη

1. Ορισμός του H_s Η συνάρτηση $H_s(x)$ ορίζεται ως:

$$H_s(x) = f_{s,x_1}\left(f_{s,x_2}\left(\cdots f_{s,x_{3n}}(1)\cdots\right)\right),$$

όπου $f_{s,b}(z) = y^{bz^e} \mod N$ για $b \in \{0,1\}$, και το $s = (N, e, y)$ παράγεται από τη γεννήτρια κλειδιών RSA.

2. Υπόθεση σύγκρουσης Έστω ότι υπάρχει αποδοτικός αλγόριθμος που βρίσκει $u, v \in \{0,1\}^{3n}$ με $u \neq v$ και $H_s(u) = H_s(v)$. Τότε:

$$H_s(u) = H_s(v) \implies f_{s,u_1}\left(f_{s,u_2}\left(\cdots f_{s,u_{3n}}(1)\cdots\right)\right) = f_{s,v_1}\left(f_{s,v_2}\left(\cdots f_{s,v_{3n}}(1)\cdots\right)\right).$$

3. Κατασκευή λύσης για το πρόβλημα RSA Ας θεωρήσουμε τη διαφορά μεταξύ των u και v. Έστω i η πρώτη θέση όπου $u_i \neq v_i$. Χωρίς απώλεια γενικότητας, έστω $u_i = 0$ και $v_i = 1$. Τότε, στο i-ο βήμα της επανάληψης, έχουμε:

$$f_{s,u_i}(z) = z^e \mod N,$$

και

$$f_{s,v_i}(z) = yz^e \mod N.$$

Από την υπόθεση ότι $H_s(u) = H_s(v)$, μπορούμε να γράψουμε:

$$z^e \equiv yz^e \mod N.$$

Απλοποιώντας, έχουμε:

$$1 \equiv y \mod N.$$

Ωστόσο, αυτό είναι αδύνατο να συμβαίνει για γενικό $y \in \mathbb{Z}_N^*$, εκτός αν μπορούμε να υπολογίσουμε e-οστές ρίζες $\mod N$. Επομένως, αν μπορούμε να βρίσκουμε u, v με $H_s(u) = H_s(v)$ αποδοτικά, τότε μπορούμε να σπάσουμε το RSA, κάτι που έρχεται σε αντίθεση με την υπόθεση ότι το RSA πρόβλημα είναι δύσκολο.

4. Συμπέρασμα Η H_s είναι collision-resistant υπό την υπόθεση ότι το RSA πρόβλημα είναι δύσκολο για την $KGen$. Αν κάποιος μπορούσε να βρει αποδοτικά σύγκρουση για τη H_s, τότε θα μπορούσε να λύσει το RSA πρόβλημα, κάτι που αναιρεί την αρχική υπόθεση δυσκολίας του RSA.

3 Άσκηση 3

Απόδειξη ισοδυναμίας των προβλημάτων SDH, IDH, και CDH

Έστω μια κυκλική ομάδα G με τάξη πρώτο αριθμό q και γεννήτορα g. Θα αποδείξουμε ότι τα προβλήματα Τετραγωνικό Diffie-Hellman (SDH), Αντιστρόφου Diffie-Hellman (IDH), και Υπολογιστικό Diffie-Hellman (CDH) είναι ισοδύναμα.

SDH IDH

IDH $\leq$ SDH

Υποθέτουμε ότι έχουμε ένα μαντείο O που, δεδομένων g και g^x, επιστρέφει g^{x^2}. Θα δείξουμε ότι μπορούμε να λύσουμε το IDH πρόβλημα, δηλαδή να υπολογίσουμε $g^{x^{-1}}$.

1. Παίρνουμε ως είσοδο το ζεύγος g^r, g, όπου r είναι τυχαίος ακέραιος. 2. Το μαντείο υπολογίζει:

$$O(g^r, g) = g^{r^2}.$$

3. Θυμόμαστε ότι $r \cdot r^{-1} \equiv 1 \pmod{q}$ και άρα $g^{r^{-1}} = g^{r^2 \cdot r^{-1}}$. Επομένως, με την έξοδο του μαντείου μπορούμε να υπολογίσουμε το $g^{r^{-1}}$ και συνεπώς να λύσουμε το IDH.

SDH $\leq$ IDH

Υποθέτουμε ότι έχουμε ένα μαντείο O που, δεδομένων g και g^x, επιστρέφει $g^{x^{-1}}$. Θα δείξουμε ότι μπορούμε να λύσουμε το SDH πρόβλημα, δηλαδή να υπολογίσουμε g^{x^2}.

1. Παίρνουμε ως είσοδο το ζεύγος g^r, g, όπου r είναι τυχαίος ακέραιος. 2. Το μαντείο υπολογίζει:

$$O(g^r, g) = g^{r^{-1}}.$$

3. Γνωρίζουμε ότι $(r^{-1})^{-1} = r$, και επομένως μπορούμε να υπολογίσουμε:

$$g^{r^2} = (g^r)^r.$$

4. Έτσι, μπορούμε να λύσουμε το SDH.

Συμπέρασμα

Αφού $IDH \leq SDH$ και $SDH \leq IDH$, έχουμε ότι $SDH \iff IDH$.

SDH CDH

SDH ≤ CDH

Υποθέτουμε ότι έχουμε ένα μαντείο O που, δεδομένων g, g^x, και g^y, επιστρέφει g^{xy}. Θα δείξουμε ότι μπορούμε να λύσουμε το SDH πρόβλημα, δηλαδή να υπολογίσουμε g^{x^2}.

1. Παίρνουμε ως είσοδο το ζεύγος g, g^r, όπου r είναι τυχαίος ακέραιος. 2. Επιλέγουμε τυχαίες τιμές $t_1, t_2 \in \mathbb{Z}_q$ και ρωτάμε το μαντείο:

$$O(g, g^{rt_1}, g^{rt_2}) = g^{r^2 t_1 t_2}.$$

3. Υπολογίζουμε το g^{r^2} ως εξής:

$$g^{r^2} = \frac{O(g, g^{rt_1}, g^{rt_2})}{g^{r^2 t_1 t_2}},$$

όπου όλοι οι όροι είναι γνωστοί, και συνεπώς μπορούμε να λύσουμε το SDH.

CDH ≤ SDH

Υποθέτουμε ότι έχουμε ένα μαντείο O που, δεδομένων g και g^x, επιστρέφει g^{x^2}. Θα δείξουμε ότι μπορούμε να λύσουμε το CDH πρόβλημα, δηλαδή να υπολογίσουμε g^{xy}.

1. Παίρνουμε ως είσοδο g, g^x, g^y. 2. Επιλέγουμε τυχαίες τιμές $s_1, s_2, t_1, t_2 \in \mathbb{Z}_q$ και ρωτάμε το μαντείο:

$$O(g, g^{xs_1}), \quad O(g, g^{ys_2}), \quad O(g, g^{xs_1 t_1 + ys_2 t_2}).$$

3. Λύνουμε το σύστημα εξισώσεων για να απομονώσουμε τον όρο g^{xy}, καθώς όλοι οι υπόλοιποι όροι είναι γνωστοί.

Συμπέρασμα

Αφού $SDH \leq CDH$ και $CDH \leq SDH$, έχουμε ότι $SDH \iff CDH$.

Τελικό Συμπέρασμα

Από τις παραπάνω αποδείξεις έχουμε ότι:

$$SDH \iff IDH \iff CDH.$$

4 Άσκηση 4

Ανάλυση Κώδικα RSA Επίθεσης

Περιγραφή Κώδικα

Η παρακάτω Python υλοποίηση πραγματοποιεί επίθεση σε κρυπτογραφημένα μηνύματα RSA εκμεταλλευόμενη τη διαρροή πληροφορίας μέσω μαντείου.

```python
import rsa

# Σ
def encrypt_message(e, message, n):
    return pow(message, e, n)

# Σ
def oracle_simulation(d, ciphertext, n):
    decrypted_message = pow(ciphertext, d, n)
    return 1 if decrypted_message > (n // 2) else 0

# T
def perform_attack(e, d, ciphertext, n):
    lower_bound = 0
    upper_bound = n
    prev_upper = upper_bound
    prev_lower = lower_bound
    i = 1

    # A
    position = oracle_simulation(d, ciphertext, n)

    while lower_bound < upper_bound:
        if position == 0:
            prev_upper = upper_bound
            upper_bound -= n // (2 ** i)
        else:
            prev_lower = lower_bound
            lower_bound += n // (2 ** i)

        # E
        adjusted_ciphertext = ciphertext * encrypt_message(e, 2 ** i, n)
        position = oracle_simulation(d, adjusted_ciphertext, n)
        i += 1

        #
        if prev_upper == upper_bound and prev_lower == lower_bound:
            break

    # E
    for potential_message in range(lower_bound, upper_bound):
        if ciphertext == pow(potential_message, e, n):
            print("Recovered Message: " + str(potential_message))
            return

# K
```

```python
def main():
    # Δ      RSA
    keys = rsa.newkeys(128)
    n = keys[0].n
    e = keys[0].e
    d = keys[1].d

    print(f"n = {n}\ne = {e}\nd = {d}\n")

    # Λ
    message = int(input("Enter the Message: "))
    print(f"Original Message: {message}")

    # K
    ciphertext = encrypt_message(e, message, n)

    # E
    perform_attack(e, d, ciphertext, n)

main()
```

Listing 1: Κώδικας Επίθεσης RSA

Περιγραφή Συναρτήσεων

- `encrypt_message(e, message, n)`: Κρυπτογραφεί το μήνυμα m εκτελώντας $c = m^e \mod n$.
- `oracle_simulation(d, ciphertext, n)`: Αποκρυπτογραφεί το c και επιστρέφει 1 αν $m > n/2$, διαφορετικά 0.
- `perform_attack(e, d, ciphertext, n)`: Υλοποιεί τη διχοτόμηση του διαστήματος $[0, n]$ χρησιμοποιώντας το μαντείο για να ανακαλύψει το αρχικό μήνυμα.
- `main()`: Δημιουργεί τα RSA κλειδιά, λαμβάνει το μήνυμα από τον χρήστη, το κρυπτογραφεί, και εκτελεί την επίθεση.

5 Άσκηση 6

Παραλλαγή του Πρωτοκόλλου ElGamal

Δίνεται το πρωτόκολλο κρυπτογράφησης ElGamal που λειτουργεί σε μια ομάδα G με τάξη πρώτο αριθμό q και γεννήτορες g, h. Οι λειτουργίες του πρωτοκόλλου είναι:

- **KGen(λ):** Δημιουργία κλειδιών.

$$\text{Δημιουργούμε } x \xleftarrow{\$} \mathbb{Z}_q^* \quad \text{και υπολογίζουμε } g^x.$$

Επιστρέφονται τα μυστικά και δημόσια κλειδιά:

$$(\mathrm{sk}, \mathrm{pk}) = (x, g^x).$$

- **Enc(pk, m):** Κρυπτογράφηση.

$$\text{Επιλέγουμε } r \xleftarrow{\$} \mathbb{Z}_q^*, \quad \text{και υπολογίζουμε το κρυπτοκείμενο:}$$

$$\text{Ciphertext } C = (g^r, g^r hm)\,.$$

1. Ορισμός της Συνάρτησης Αποκρυπτογράφησης

Η αποκρυπτογράφηση βασίζεται στη χρήση του μυστικού κλειδιού x. Αν το κρυπτοκείμενο είναι $C = (c_1, c_2)$, τότε ο παραλήπτης υπολογίζει:

$$m = \frac{c_2}{c_1^x} \mod q.$$

Αναλυτικότερα:

$$c_1 = g^r,$$
$$c_2 = g^r hm.$$

Υπολογίζουμε:

$$c_1^x = (g^r)^x = g^{rx}.$$

Από την τιμή c_2, έχουμε:

$$\frac{c_2}{c_1^x} = \frac{g^r hm}{g^{rx}} = hm \cdot \frac{g^r}{g^r} = m \mod q.$$

Απόδειξη Ορθότητας

Η διαδικασία αποκρυπτογράφησης είναι ορθή, καθώς επαναφέρει το μήνυμα m:

$$c_2 = g^r hm, \quad c_1^x = g^{rx}.$$

Άρα:

$$\frac{c_2}{c_1^x} = \frac{g^r hm}{g^{rx}} = hm \cdot \frac{g^r}{g^r} = m \mod q.$$

2. Μελέτη Ασφάλειας

Θα αναλύσουμε την ασφάλεια του πρωτοκόλλου βάσει των παρακάτω ιδιοτήτων:

(α) Ασφάλεια ως προς OW-CPA (One-Wayness under Chosen Plaintext Attack)

Η ιδιότητα OW-CPA απαιτεί ότι ένας αντίπαλος δεν μπορεί να υπολογίσει το μήνυμα m από το κρυπτοκείμενο (c_1, c_2) σε πεπερασμένο χρόνο. Στο παρόν πρωτόκολλο, η δυσκολία βασίζεται στην υπόθεση του Διακριτού Λογαρίθμου (DL). Επειδή η τιμή h είναι γεννήτορας με άγνωστο διακριτό λογάριθμο ως προς το g, ένας αντίπαλος δεν μπορεί να υπολογίσει το m χωρίς να λύσει το πρόβλημα DL ή χωρίς γνώση του x.

(β) Ασφάλεια ως προς IND-CPA (Indistinguishability under Chosen Plaintext Attack)

Η ιδιότητα IND-CPA απαιτεί ότι ένας αντίπαλος δεν μπορεί να διακρίνει δύο κρυπτοκείμενα που αντιστοιχούν σε διαφορετικά μηνύματα m_0, m_1 υπό τις ίδιες παραμέτρους. Στο παρόν πρωτόκολλο, η τιμή r επιλέγεται τυχαία για κάθε κρυπτογράφηση, καθιστώντας το κρυπτοκείμενο ομοιόμορφα κατανεμημένο. Άρα, ακόμα κι αν ο αντίπαλος γνωρίζει το δημόσιο κλειδί g^x, δεν μπορεί να συνδέσει το c_2 με το m λόγω του τυχαίου r.

(γ) Ασφάλεια ως προς IND-CCA (Indistinguishability under Chosen Ciphertext Attack)

Η ιδιότητα IND-CCA απαιτεί ότι ένας αντίπαλος δεν μπορεί να διακρίνει δύο κρυπτοκείμενα ακόμα και αν έχει πρόσβαση σε μαντείο αποκρυπτογράφησης, εκτός από το κρυπτοκείμενο στόχο. Το παρόν πρωτόκολλο δεν είναι ασφαλές ως προς IND-CCA. Αν ο αντίπαλος μπορεί να υποβάλλει ερωτήματα αποκρυπτογράφησης, μπορεί να χρησιμοποιήσει το μαντείο για να υπολογίσει την τιμή m από το c_2 με τρόπο που καταστρατηγεί την ασφάλεια.

Συμπέρασμα

Το πρωτόκολλο είναι ασφαλές ως προς OW-CPA και IND-CPA, υπό την υπόθεση του Διακριτού Λογαρίθμου και της σωστής επιλογής παραμέτρων g, h. Ωστόσο, δεν είναι ασφαλές ως προς IND-CCA λόγω της δυνατότητας χρήσης μαντείου αποκρυπτογράφησης.

6 Άσκηση 7

Ανάλυση του Πρωτοκόλλου Π

Δίνεται το πρωτόκολλο δέσμευσης του Pedersen με $c = g^m h^r$, όπου $m, r \in \mathbb{Z}_q^*$. Το πρωτόκολλο Π λειτουργεί ως εξής:

1. Ο prover επιλέγει ομοιόμορφα $t_1, t_2 \in \mathbb{Z}_q^*$ και υπολογίζει:

$$t = g^{t_1} h^{t_2}.$$

Στη συνέχεια, στέλνει το t στον verifier.

2. Ο verifier επιλέγει ομοιόμορφα $e \in \mathbb{Z}_q^*$ και το στέλνει στον prover.
3. Ο prover υπολογίζει:

$$s_1 = t_1 + em \mod q, \quad s_2 = t_2 + er \mod q,$$

και στέλνει τα s_1, s_2 στον verifier.

4. Ο verifier αποδέχεται αν και μόνο αν:

$$g^{s_1} h^{s_2} = t c^e.$$

Θα εξετάσουμε τις ιδιότητες πληρότητας, ειδικής ορθότητας και μηδενικής γνώσης για τίμιους επαληθευτές.

1. Είναι το Π Σ-πρωτόκολλο;

(α) Πληρότητα

Αν ο prover γνωρίζει m, r τέτοια ώστε $c = g^m h^r$, τότε:

$$t = g^{t_1} h^{t_2},$$

και ο verifier λαμβάνει:

$$g^{s_1} h^{s_2} = g^{t_1 + em} h^{t_2 + er} = g^{t_1} g^{em} h^{t_2} h^{er} = t \cdot c^e.$$

Άρα η συνθήκη $g^{s_1} h^{s_2} = t c^e$ ισχύει πάντα, επομένως το πρωτόκολλο έχει **πληρότητα**.

(β) Ειδική Ορθότητα

Έστω δύο διαφορετικές εκτελέσεις του πρωτοκόλλου για την ίδια τιμή t, αλλά με διαφορετικές προκλήσεις e και e', που οδηγούν σε απαντήσεις (s_1, s_2) και (s_1', s_2'). Τότε:

$$s_1 - s_1' = em - e'm \mod q \quad \text{και} \quad s_2 - s_2' = er - e'r \mod q.$$

Από τις παραπάνω σχέσεις, μπορούμε να υπολογίσουμε τα m και r ως:

$$m = \frac{s_1 - s_1'}{e - e'} \mod q, \quad r = \frac{s_2 - s_2'}{e - e'} \mod q.$$

Άρα, ο verifier μπορεί να ανακτήσει το m και r αν λάβει δύο διαφορετικές προκλήσεις, επομένως το πρωτόκολλο έχει **ειδική ορθότητα**.

(γ) Μηδενική Γνώση για Τίμιους Επαληθευτές

Για να δείξουμε ότι το πρωτόκολλο είναι **μηδενικής γνώσης**, κατασκευάζουμε έναν εξομοιωτή που παράγει συνομιλίες (t, e, s_1, s_2) που είναι αδιακρίτως ίδιες με τις πραγματικές συνομιλίες. Ο εξομοιωτής λειτουργεί ως εξής:

1. Επιλέγει τυχαία $e, s_1, s_2 \in \mathbb{Z}_q^*$.
2. Υπολογίζει:
$$t = g^{s_1} h^{s_2} c^{-e}.$$

Η συνομιλία (t, e, s_1, s_2) που παράγεται από τον εξομοιωτή έχει την ίδια κατανομή με την πραγματική συνομιλία. Επομένως, το πρωτόκολλο είναι **μηδενικής γνώσης για τίμιους επαληθευτές**.

2. Witness Indistinguishability

Για να δείξουμε ότι το Π είναι **witness indistinguishable**, εξετάζουμε αν ένας κακόβουλος verifier μπορεί να διακρίνει δύο διαφορετικά witness (m, r) και (m', r') που ικανοποιούν τη σχέση $c = g^m h^r$. Στο Π, το t εξαρτάται από τυχαία επιλεγμένα t_1, t_2, ενώ οι τιμές s_1, s_2 είναι κρυπτογραφικά δεσμευμένες από το e. Επομένως, ο verifier δεν μπορεί να διακρίνει μεταξύ δύο witness, καθώς οι απαντήσεις έχουν την ίδια κατανομή ανεξαρτήτως του (m, r). Άρα, το πρωτόκολλο είναι **witness indistinguishable**.

3. Το Πρωτόκολλο Π′

Στο Π′, ο prover υπολογίζει και στέλνει:

$$a = g^{t_1}, \quad b = h^{t_2}.$$

Ο verifier ελέγχει αν:

$$a \cdot b \cdot c^e = g^{s_1} h^{s_2}.$$

(α) Ιδιότητα Σ-Πρωτοκόλλου

Το Π′ εξακολουθεί να ικανοποιεί τις ιδιότητες πληρότητας, ειδικής ορθότητας και μηδενικής γνώσης, καθώς η μόνη αλλαγή αφορά τη διάσπαση του t στις τιμές a, b, που δεν επηρεάζουν τη βασική λογική του πρωτοκόλλου. Επομένως, το Π′ είναι επίσης **Σ-πρωτόκολλο**.

7 Άσκηση 8

Ανάλυση του Πρωτοκόλλου

Το πρωτόκολλο αποδεικνύει ότι ο prover P γνωρίζει το μήνυμα $m \in \mathbb{Z}_n^*$ που αντιστοιχεί στο κρυπτοκείμενο $y = m^e \mod n$ με δημόσιο κλειδί (e, n), όπου e είναι πρώτος. Το πρωτόκολλο έχει τα παρακάτω βήματα:

1. Ο P επιλέγει τυχαία $t \in \mathbb{Z}_n^*$ και υπολογίζει:

$$h = t^e \mod n.$$

   Στη συνέχεια, στέλνει το h στον verifier V.

2. Ο V επιλέγει τυχαία $c \in \{0, 1, \ldots, e-1\}$ και το στέλνει στον P.

3. Ο P υπολογίζει:

$$r = tm^c \mod n,$$

   και στέλνει το r στον V.

4. Ο V επαληθεύει αν ισχύει η εξίσωση:

$$r^e \equiv hy^c \mod n.$$

Θα εξετάσουμε αν το πρωτόκολλο είναι Σ-πρωτόκολλο, δηλαδή αν ικανοποιεί τις ιδιότητες πληρότητας, ειδικής ορθότητας και μηδενικής γνώσης για τίμιους επαληθευτές (HVZK).

1. Πληρότητα

Αν ο P ακολουθήσει το πρωτόκολλο σωστά, τότε:

$$h = t^e \mod n,$$

και:

$$r = tm^c \mod n.$$

Ο verifier ελέγχει αν:

$$r^e = (tm^c)^e \mod n = t^e(m^c)^e \mod n = t^e m^{ce} \mod n.$$

Από τον ορισμό του y, έχουμε $y = m^e \mod n$, άρα $y^c = m^{ce} \mod n$. Επομένως:

$$r^e = t^e m^{ce} \mod n = t^e y^c \mod n = hy^c \mod n.$$

Η εξίσωση ισχύει, συνεπώς το πρωτόκολλο έχει **πληρότητα**.

2. Ειδική Ορθότητα

Έστω ότι ο P στέλνει δύο διαφορετικές απαντήσεις (r, c) και (r', c') για την ίδια τιμή h, όπου $c \neq c'$. Τότε:

$$r = tm^c \mod n, \quad r' = tm^{c'} \mod n.$$

Παίρνουμε τον λόγο:

$$\frac{r}{r'} = \frac{tm^c}{tm^{c'}} \mod n = m^{c-c'} \mod n.$$

Από εδώ μπορούμε να υπολογίσουμε το m ως:

$$m = \left(\frac{r}{r'}\right)^{(c-c')^{-1} \mod e} \mod n,$$

όπου $(c-c')^{-1} \mod e$ είναι το αντίστροφο του $c-c'$ στο $\mathbb{Z}_e^*$. Επομένως, ο V μπορεί να ανακτήσει το m, άρα το πρωτόκολλο έχει **ειδική ορθότητα**.

3. Μηδενική Γνώση για Τίμιους Επαληθευτές (HVZK)

Για να δείξουμε τη μηδενική γνώση, κατασκευάζουμε έναν εξομοιωτή S που παράγει συνομιλίες (h, c, r) αδιακρίτως ίδιες με αυτές που παράγονται από το πρωτόκολλο. Ο εξομοιωτής S λειτουργεί ως εξής:

1. Επιλέγει τυχαία $c \in \{0, 1, \ldots, e-1\}$ και $r \in \mathbb{Z}_n^*$.
2. Υπολογίζει:
$$h = r^e y^{-c} \mod n.$$
3. Επιστρέφει τη συνομιλία (h, c, r).

Ανάλυση Κατανομής

Οι συνομιλίες (h, c, r) που παράγονται από τον εξομοιωτή έχουν την ίδια κατανομή με αυτές του πρωτοκόλλου, επειδή:

- Το c επιλέγεται ομοιόμορφα από $\{0, 1, \ldots, e-1\}$.
- Το r είναι τυχαίο στο $\mathbb{Z}_n^*$.
- Η σχέση $h = r^e y^{-c} \mod n$ διασφαλίζει ότι $r^e \equiv h y^c \mod n$, όπως απαιτεί το πρωτόκολλο.

Πιθανότητα Εμφάνισης Transcript

Ένα transcript (h, c, r) έχει πιθανότητα εμφάνισης:

$$\Pr[(h, c, r)] = \frac{1}{e} \cdot \frac{1}{n},$$

καθώς υπάρχουν e δυνατές τιμές για το c και n δυνατές τιμές για το r. Αυτή η πιθανότητα είναι ίδια με την αντίστοιχη πιθανότητα στο πρωτόκολλο, αποδεικνύοντας τη μηδενική γνώση.

Συμπέρασμα

Το πρωτόκολλο είναι Σ-πρωτόκολλο, καθώς ικανοποιεί τις ιδιότητες:

1. Πληρότητα.
2. Ειδική Ορθότητα.
3. Μηδενική Γνώση για Τίμιους Επαληθευτές (HVZK).

8 Άσκηση 9

Πρόβλημα

Η μη-αλληλεπιδραστική έκδοση του πρωτοκόλλου Schnorr, όπως ορίζεται στην διάλεξη ZK, υπολογίζει το challenge c ως εξής:

$$c := H(h||y),$$

όπου h είναι το δημόσιο κλειδί του prover και y το πρώτο μήνυμα του prover. Στην παραλλαγή αυτή, το challenge υπολογίζεται μόνο από το y, δηλαδή:

$$c := H(y).$$

Είναι ζητούμενο να αποδείξουμε ότι αυτή η παραλλαγή του πρωτοκόλλου δεν διαθέτει την ιδιότητα της ορθότητας.

Ανάλυση της Επίθεσης

Θα σχεδιάσουμε μια επίθεση στην οποία ένας κακόβουλος prover μπορεί να δημιουργήσει μια έγκυρη απόδειξη γνώσης του διακριτού λογαρίθμου κάποιου στοιχείου y της ομάδας G, χωρίς όμως να γνωρίζει το διακριτό λογάριθμο $\log_g y$.

Η Κανονική Διαδικασία του Πρωτοκόλλου Schnorr

Στο κανονικό πρωτόκολλο Schnorr, η απόδειξη γνώσης του διακριτού λογαρίθμου $\log_g y$ ακολουθεί τα εξής βήματα:

1. Ο prover επιλέγει τυχαία έναν αριθμό r από την ομάδα G.
2. Υπολογίζει το $h = g^r \mod p$ και το στέλνει στον verifier.
3. Ο verifier υπολογίζει το challenge $c := H(h||y)$.
4. Ο prover υπολογίζει το $s = r + c \log_g y \mod q$, το στέλνει στον verifier.
5. Ο verifier ελέγχει αν:
$$g^s \equiv h \cdot y^c \mod p.$$

Η παραπάνω διαδικασία διασφαλίζει ότι ο prover αποδεικνύει την γνώση του $\log_g y$ χωρίς να το αποκαλύπτει.

Η Παραλλαγή με $c := H(y)$

Στην παραλλαγή του πρωτοκόλλου, το challenge υπολογίζεται ως $c := H(y)$. Στην περίπτωση αυτή, το challenge εξαρτάται μόνο από το y, το οποίο είναι το πρώτο μήνυμα του prover. Αυτή η τροποποίηση επηρεάζει την ασφαλή διαδικασία και επιτρέπει σε έναν κακόβουλο prover να δημιουργήσει έγκυρη απόδειξη χωρίς να γνωρίζει το διακριτό λογάριθμο $\log_g y$.

Επίθεση από Κακόβουλο Prover

Ένας κακόβουλος prover μπορεί να επιλέξει αυθαίρετα ένα στοιχείο y' από την ομάδα G και να υπολογίσει το challenge $c := H(y')$. Στη συνέχεια, μπορεί να επιλέξει τυχαία έναν αριθμό r' και να υπολογίσει το $h' = g^{r'} \mod p$.

Στη συνέχεια, στέλνει στον verifier το ζεύγος (h', c, r'). Ο verifier, υπολογίζοντας το challenge $c := H(y')$, θα ελέγξει αν η εξίσωση:

$$g^{r'} \equiv h' \cdot y'^{c} \mod p$$

ισχύει, κάτι το οποίο θα συμβαίνει, καθώς έχει προεπιλέξει το challenge c και μπορεί να υπολογίσει το $h' = g^{r'} \mod p$. Ο verifier δεν μπορεί να διακρίνει ότι ο prover δεν γνωρίζει τον διακριτό λογάριθμο για το y', καθώς το challenge εξαρτάται μόνο από το y' και δεν περιλαμβάνει το δημόσιο κλειδί του prover.

Έτσι, ο κακόβουλος prover έχει δημιουργήσει μία έγκυρη απόδειξη γνώσης του διακριτού λογαρίθμου χωρίς να γνωρίζει το πραγματικό διακριτό λογάριθμο $\log_g y'$.

Συμπέρασμα

Από την ανάλυση αυτή προκύπτει ότι η παραλλαγή του πρωτοκόλλου Schnorr, όπου το challenge υπολογίζεται ως $c := H(y)$, δεν διαθέτει την ιδιότητα της ορθότητας. Ο κακόβουλος prover μπορεί να δημιουργήσει έγκυρες αποδείξεις χωρίς να γνωρίζει το διακριτό λογάριθμο, εκμεταλλευόμενος το γεγονός ότι το challenge εξαρτάται μόνο από το y, και όχι από το δημόσιο κλειδί του prover.

9 Άσκηση 10

Η υπογραφή Schnorr είναι ένα σχήμα υπογραφής δημόσιου κλειδιού που βασίζεται στη δυσκολία του προβλήματος του διακριτού λογαρίθμου. Χρησιμοποιείται ευρέως στην κρυπτογραφία και προσφέρει υψηλό επίπεδο ασφάλειας. Στην παρούσα υλοποίηση, η υπογραφή Schnorr θα χρησιμοποιήσει μια υποομάδα πρώτης τάξης q του Z_p^*.

Πρωτόκολλο Υπογραφής Schnorr

Η υπογραφή Schnorr ακολουθεί τα εξής βήματα:

1. Ο prover επιλέγει τυχαία έναν αριθμό r από την υποομάδα $\mathbb{Z}_q^*$, και υπολογίζει το $t = g^r \mod p$, όπου g είναι ο γεννήτορας της ομάδας.

2. Ο verifier επιλέγει τυχαία το challenge $c = H(t, m)$, όπου m είναι το μήνυμα που υπογράφει ο prover και H είναι μια συνάρτηση κατακερματισμού.

3. Ο prover υπολογίζει το $s = r + c \cdot x \mod q$, όπου x είναι το ιδιωτικό κλειδί του prover.

4. Ο prover στέλνει την υπογραφή (s, t) στον verifier.

5. Ο verifier επαληθεύει αν η εξίσωση $g^s \equiv t \cdot y^c \mod p$ ισχύει, όπου $y = g^x \mod p$ είναι το δημόσιο κλειδί του prover.

Υλοποίηση σε Python

Η υλοποίηση του παρακάτω αλγορίθμου χρησιμοποιεί την βιβλιοθήκη `pow` για τον υπολογισμό εκθετικών υπολοίπων, και την βιβλιοθήκη `hashlib` για τη συνάρτηση κατακερματισμού.

```
import random
import hashlib

# O
p = 104729  #
q = (p - 1) // 2  # T              Z_p^*
g = 2  # O
x = random.randint(1, q - 1)  # I           prover
y = pow(g, x, p)  # A           prover

def H(data):
    """ S             H """
    return int(hashlib.sha256(data.encode()).hexdigest(), 16) %
     q

def sign(m):
    """ T    Schnorr """
    r = random.randint(1, q - 1)  # E        r
    t = pow(g, r, p)  # T      t = g^r mod p
    c = H(str(t) + m)  # T        challenge c = H(t, m)
    s = (r + c * x) % q  # T        s = r + c * x mod q
    return s, t

def verify(m, s, t):
    """ E          Schnorr """
    c = H(str(t) + m)  # T        challenge c = H(t, m)
    lhs = pow(g, s, p)  # g^s mod p
    rhs = (t * pow(y, c, p)) % p  # t * y^c mod p
    return lhs == rhs  # E            g^s   t * y^c mod p

# P
message = "Hello, world!"
s, t = sign(message)  # A
print(f"T    : (s={s}, t={t})")
is_valid = verify(message, s, t)  # E
print(f"H            : {is_valid}")
```

Επεξήγηση Κώδικα

1. Αρχικά, ορίζουμε τις παραμέτρους της ομάδας p και q, καθώς και τον γεννήτορα g. Το ιδιωτικό κλειδί x επιλέγεται τυχαία, και το δημόσιο κλειδί y υπολογίζεται ως $y = g^x \mod p$.

2. Στη συνέχεια, υλοποιούμε τη συνάρτηση H, η οποία είναι η συνάρτηση κατακερματισμού, η οποία επιστρέφει το challenge c.

3. Η συνάρτηση `sign` υλοποιεί την υπογραφή του μηνύματος. Επιλέγεται τυχαία το r, υπολογίζεται το $t = g^r \mod p$, και το challenge $c = H(t, m)$ υπολογίζεται. Ο prover υπολογίζει και επιστρέφει την υπογραφή (s, t).

4. Η συνάρτηση `verify` ελέγχει την εγκυρότητα της υπογραφής. Υπολογίζει το challenge c και επαληθεύει αν η εξίσωση $g^s \equiv t \cdot y^c \mod p$ ισχύει.

10 Άσκηση 11

Περιγραφή του Σχήματος Υπογραφών

Για την υπογραφή του μηνύματος m, ο υπογράφων ακολουθεί τα εξής βήματα:

1. Επιλέγει $h \in \{0, \ldots, p-2\}$ ώστε να ικανοποιείται η σχέση:

$$H(m) + x + h \equiv 0 \pmod{p-1},$$

   όπου $H(m)$ είναι μια συνάρτηση σύνοψης (hash function) του μηνύματος m.

2. Η υπογραφή είναι η τριάδα:

$$\text{Sign}(x, m) = (m, (x+h) \mod (p-1), g^h \mod p).$$

Για να επαληθευτεί μια υπογραφή (m, a, b), οι παρακάτω δύο συνθήκες πρέπει να ικανοποιούνται:

1. $y^b \equiv g^a \pmod{p}$,
2. $g^{H(m)} y^b \equiv 1 \pmod{p}$.

Επίθεση Καθολικής Πλαστογράφησης

Η επίθεση καθολικής πλαστογράφησης αφορά την ικανότητα ενός κακόβουλου επιτιθέμενου να δημιουργήσει έγκυρες υπογραφές για μηνύματα τα οποία δεν έχουν υπογραφεί από τον υπογράφοντα χρήστη. Δηλαδή, ο επιτιθέμενος θέλει να κατασκευάσει μια υπογραφή για κάποιο μήνυμα m' χωρίς να γνωρίζει το ιδιωτικό κλειδί του υπογράφοντος.

Ας θεωρήσουμε ότι ο επιτιθέμενος έχει καταφέρει να πάρει μία έγκυρη υπογραφή (m, a, b) για το μήνυμα m, με το οποίο γνωρίζει τα εξής:

$$a = (x + h) \mod (p - 1) \quad \text{και} \quad b = g^h \mod p.$$

Ας δούμε τώρα τι μπορεί να κάνει ο επιτιθέμενος για να πλαστογραφήσει μια υπογραφή για κάποιο άλλο μήνυμα m'.

Βήματα της Επίθεσης: 1. Ο επιτιθέμενος επιλέγει ένα νέο μήνυμα m'. 2. Υπολογίζει το νέο a' και b' για το μήνυμα m' με τους εξής υπολογισμούς:

$$a' = a \quad \text{και} \quad b' = b.$$

Ειδικότερα, $a' = (x + h) \mod (p - 1)$ παραμένει το ίδιο, και το $b' = g^h \mod p$ είναι επίσης αμετάβλητο.

3. Επαληθεύει ότι το νέο (m', a', b') ικανοποιεί τις συνθήκες επαλήθευσης: - Από την πρώτη συνθήκη $y^b \equiv g^a \pmod{p}$, η οποία ήδη ισχύει από την προηγούμενη υπογραφή. - Από τη δεύτερη συνθήκη $g^{H(m')} y^b \equiv 1 \pmod{p}$, η οποία επίσης ισχύει καθώς η συνθήκη επαλήθευσης είναι αμετάβλητη.

Συμπέρασμα Επίθεσης: Ο επιτιθέμενος μπορεί να πλαστογραφήσει την υπογραφή για το μήνυμα m' χωρίς να γνωρίζει το ιδιωτικό κλειδί x, καθώς τα a και b παραμένουν τα ίδια για διαφορετικά μηνύματα. Επομένως, το σχήμα υπογραφής αυτό δεν προστατεύει από επίθεση καθολικής πλαστογράφησης, διότι ο επιτιθέμενος μπορεί να δημιουργήσει έγκυρες υπογραφές για οποιοδήποτε μήνυμα, αν έχει αποκτήσει μία έγκυρη υπογραφή για κάποιο άλλο μήνυμα.

Συμπεράσματα

Η ανάλυση αυτή δείχνει ότι το σχήμα υπογραφής που περιγράφεται δεν προστατεύει από επίθεση καθολικής πλαστογράφησης. Ο επιτιθέμενος μπορεί να δημιουργήσει έγκυρες υπογραφές για νέα μηνύματα, χρησιμοποιώντας τα a και b από μια έγκυρη υπογραφή χωρίς να γνωρίζει το ιδιωτικό κλειδί του υπογράφοντος. Επομένως, το σχήμα δεν είναι ασφαλές και απαιτεί τροποποιήσεις για να προσφέρει την απαραίτητη ασφάλεια.