**Безкровна Дар’я**

**ІПС-31**

**Кр основи криптології**

**Алгоритм Ель-Гамаль**

Алгоритм Ель-Гамаля - це криптографічний алгоритм, що використовується для шифрування та підпису повідомлень. Базується даний алгоритм на складних математичних задачах, пов'язаних з теорією чисел.

**Опис кроків алгоритму Ель-Гамаля для шифрування:**

* Вибір параметрів:

Обираються два великих простих числ p I g. Число p є простим числом, а g - першим коренем за модулем p.

* Генерація ключів:

Закритий ключ x вибирається випадковим чином з інтервалу 1 < x < p−1.

Відкритий ключ y = g^x modp.

* Шифрування повідомлення:

Нехай M - це повідомлення, яке потрібно зашифрувати.

Вибирається випадкове ціле число k таке, що 1<k<p−1 і НСД(k,p−1)=1.

Обчислюються два числа: a=g ^k mod p і b=M\*y ^k mod p.

* Дешифрування повідомлення:

Для дешифрування використовується закритий ключ x.

Спочатку обчислюється обернене k до k за модулем p−1.

Потім повідомлення M вилучається як M=b⋅(a ^x)^ −1 mod p.

Алгоритм Ель-Гамаля також може використовуватися для створення електронного цифрового підпису. У цьому випадку підписувальна сторона створює підпис на основі свого закритого ключа, а інші сторони можуть перевірити підпис за допомогою відкритого ключа підписавшої сторони.

**Приклад** алгоритму Ель-Гамаля для шифрування та дешифрування повідомлення.

1. Вибір параметрів:
   * Оберемо просте число p=23.
   * За модулем p перший корінь g - будь-яке число від 2 до 22. Давайте візьмемо g=5.
2. Генерація ключів:
   * Закритий ключ x - випадкове число, наприклад, x=6.
   * Відкритий ключ y обчислюється як y =g^x mod p=5^6 mod 23= 8.

Тепер, нехай у нас є повідомлення, яке ми хочемо зашифрувати: M=15.

1. Шифрування повідомлення:
   * Оберемо випадкове k, наприклад, k=3.
   * Обчислюємо a=g^k mod p=5^3 mod  23=10.
   * Обчислюємо b=M\*y^k mod p=15⋅8^3 mod 23=15\*512 mod 23=12.

Тепер пара шифртексту буде (a,b)=(10,12).

1. Дешифрування повідомлення:
   * Ми використовуємо закритий ключ x=6.
   * Обчислюємо обернене k за модулем p−1: k^(−1)=3^(−1) mod22=15.
   * За допомогою цього оберненого k ми дешифруємо M = b\*(a^x)^−1 mod p:

M=12\*(10^6)^15 mod23=12\*16^15 mod23=15.

Таким чином, ми успішно дешифрували повідомлення M=15.

Начало формы

**Атака Вінера в RSA**

У деяких програмах криптосистем RSA необхідно прискорити процеси розшифрування. Тому має значення вибір невеликого експонента, що розшифровує, d. Зрозуміло, що при цьому має велике значення відкриті експоненти Е.

Якщо розшифровуюча експонента d<(1/3)N1/4, можна запропонувати ефективний алгоритм (атака Вінера) знаходження розшифровуючої експоненти d.

Атака Вінера виходить з теорії безперервних дробів.

Розглянемо поняття безперервного дробу.

Нехай дано деяке речове число a, тоді визначимо наступну послідовність

d0=a, p0=q0=1, p1=a0\*a1+1, q1=a1,

ai=[di], d(i+1)=1/(di-ai),

pi = ai \* p (i-1) + p (i-2), при i> = 2

qi=ai\*q(i-1)+q(i-2), при i>=2

Цілі числа a0, a1, a2, ... називаються безперервним дробом, що представляє a.

Раціональні числа pi/qi називаються відповідними дробами. Кожна відповідна дріб нескоротна, а швидкість зростання їх знаменників можна порівняти з показовою.

Одним із важливих результатів теорії безперервних дробів є те, що якщо нескоротний дріб p/q задовольняє нерівності

|a-p/q|<=1/(2\*q2), (1)

то p/q є одним із відповідних дробів у розкладанні a в безперервний дріб.

Розглянемо використання цього результату в атаці Вінера.

Нехай є модуль N = pq причому

q < р < 2q. Припустимо, що експонента, що розшифровує, задовольняє нерівності: d<(1/3)N1/4

Крім того, є шифруюча експонента Е володіє властивістю

Ed = 1 (mod φ),

де φ(N)=(p-1)(q-1). Будемо також вважати, що

E<φ. Тоді отримуємо

Ed-kφ=1 (k деяке натуральне число) (2)

Отже, розділивши рівність на dφ отримаємо

|E/φ-k/d|=1/dφ

Оскільки φ=N-p-q+1≈N, то отримуємо

N-φ|=|p+q-1|<|2q+q-1|=|3q-1|<|3q|=3|q|<3\*sqrt(N)

Звідси можна отримати

|E/N-k/d|=|(Ed-kN)/dN|=|(Ed-kφ+kφ-kN)/dN|=

=|(1+k(φ-N))/dN|<=|(3k\*sqrt(N)-1)/dN|<

<|3k\*sqrt(N)/dN|=3k/(d\*sqrt(N))

Оскільки E<φ, то k<d (як видно з рівності (2)).

Крім того, за припущенням d<(1/3)N1/4 тоді d2<(1/9)N1/2

3k/(d\*sqrt(N))<3k/(9d\*d2)<3d/(9d\*d2)=1/(3d2)

Тобто отримуємо

|E/N-k/d|<1/(3d2)

Тоді згідно (1) k/d – відповідний дріб у розкладанні E/N

Таким чином, розкладаючи число E/N в безперервний дріб, можна дізнатися експоненту, що розшифровує, по черзі підставляючи знаменники відповідних дробів у вираз:

(mE)d = m (mod N)

де m випадково вибране число.

Розглянемо приклад. Нехай N має вигляд

N = 9449868410449.

Нехай відкритий ключ криптосистеми заданий як

E = 6792605 526025

а секретний ключ задовольняє нерівності

d<(1/3)N1/4 ≈584

Розкладемо число а=E/N у безперервний дріб і перевіримо знаменник.

Відповідні дроби розкладання a

1, 2/3,3/4,5/7,18/25, 23/32, 409/569,….

Почергово перевіряючи знаменники, можна переконатися, що d = 569 — секретний ключ, що шукається.

**Приклад**

Нехай *N* - це публічний ключ RSA, який ми хочемо зламати, і *e* - це відкрита експонента (зазвичай вона має значення 65537).

Атака Вінера ґрунтується на тому, що якщо приватний ключ *d* надто малий у порівнянні з *N*, то можливо знайти його за допомогою методу континуант.

Отже, нехай *d* - приватний ключ RSA, який нам потрібно знайти, і нехай *e*⋅*d*−1 ділиться на *λ*(*N*), де *λ*(*N*) - кармайклова функція Ейлера.

Тепер, якщо *d* надто маленький, можливо знайти його за допомогою апроксимації.

Ось простий приклад:

1. Нехай *N*=91 і *e*=5. Ми хочемо знайти *d*.
2. Обчислимо *λ*(*N*)=НСД(*p*−1,*q*−1), де *p* і *q* - прості множники *N*.
   * *p*=7 і *q*=13, отже, *λ*(*N*)=НСД(6,12)=6.
3. Тепер ми знаємо, що *e*⋅*d*−1=*k*⋅*λ*(*N*), де *k* - якийсь цілий додатній коефіцієнт.
   * Тут 5⋅*d*−1=6*k*.
4. Ми можемо знайти найменший цілий додатній *k*, який задовольняє це рівняння.
   * Давайте випробуємо *k*=1, тоді *d*=(6⋅1+1)/5​=5/7​=1.4, що не є цілим числом.
   * Спробуємо *k*=2, отримаємо *d*=(6⋅2+1)/5=13/5​=2.6, що також не є цілим числом.
   * Продовжуємо і таким чином, поки не знайдемо ціле число *d*. Якщо *d* дійсно надто маленький, це може статися досить швидко.

Отже, у цьому прикладі *d*=7. Тепер ми можемо використати *d* для розшифрування повідомлень, зашифрованих за допомогою відповідного публічного ключа.