Tên: Nguyễn Việt Thanh Duy

MSSV: 19127378

Báo cáo phương pháp tấn công RSA

Thuật toán phân tích Fermat

1. **Tài liệu tham khảo:**
   Phương pháp được nêu trong bài báo khoa học: Performance Analysis of Fermat Factorization Algorithms (link).
2. **Cơ sở toán học:**
   - Trong thực tế, ta cần chọn p, q có cùng độ dài bit để tạo được 1 mã RSA mạnh, tuy nhiên nếu p, q quá gần nhau thì lại tạo ra lỗ hổng bảo mật khi mà attacker có thể dễ dàng factorize n.
   - Điều kiện sử dụng thuật toán: $p - q < \sqrt[4]{n}$
   - Thuật toán dựa trên hằng đẳng thức: $x^2 - y^2 = (x - y)(x + y)$
   - Ta có:

$$n = pq = \left(\frac{q-p}{2}\right)^2 - \left(\frac{p+q}{2}\right)^2 = x^2 - y^2$$

   - Vậy n có thể được phân tích thừa số nguyên tố như sau: n = (x-y)(x+y).
   - Như vậy nếu ta có thể tìm được (x,y) ta có thể suy ra dc (p,q):

$$p = x + y$$

$$q = x - y$$

   - Thuật toán phân tích Fermat có thể giúp ta làm được điều đó, mã giả:

Algorithm FF (Fermat's Factorization)
Input: n is a positive odd number.
Output: p and q are two prime numbers such that $n = p\,q$.
Begin
1. $x = \lfloor \sqrt{n} \rfloor + 1$
2. $y = x^2 - n$
3. While (y is not a perfect square) do
4. $\quad x = x + 1$
5. $\quad y = x^2 - n$
6. End while
7. $p = x + \sqrt{y}$
8. $q = x - \sqrt{y}$

End.

3. Thực nghiệm mô phỏng tấn công:

❖ Chương trình mô phỏng(được viết bằng ngôn ngữ Python) bao gồm các phần:

- Xây dựng mã RSA với |p-q| nhỏ để dễ dàng mô phỏng tấn công.
- Tấn công RSA bằng phương pháp phân tích Fermat.

❖ Kết quả một số lần chạy thử chương trình:

- TN1: p,q là số nguyên tố 32 bit

```
PS C:\Users\BIN\Desktop\Cryp\lab05> python -u "c:\Users\BIN\Desktop\Cryp\Bonus\Source code.py"
Cho m = 19127378
Ma hoa m bang RSA:
e = 4737242479
d = 3145199619658514847
n = 20897929801601982211
c = 3724650044171855717
Tan cong RSA:
p = 4698568169
q = 4447723019
d = 3145199619658514847
m = 19127378
```

- TN2: p,q là số nguyên tố 64 bit

```
PS C:\Users\BIN\Desktop\Cryp\lab05> python -u "c:\Users\BIN\Desktop\Cryp\Bonus\Source code.py"
Cho m = 19127378
Ma hoa m bang RSA:
e = 25676790046854447517
d = 44510716936365599944604086061709900985 3
n = 48724094791153264135549248741259943479 1
c = 35808461391468727886057658899387182124 7
Tan cong RSA:
p = 22073535011672521901
q = 22073535011672521891
d = 44510716936365599944604086061709900985 3
m = 19127378
```

- TN3: p,q là số nguyên tố 128 bit

```
PS C:\Users\BIN\Desktop\Cryp\lab05> python -u "c:\Users\BIN\Desktop\Cryp\Bonus\Source code.py"
Cho m = 19127378
Ma hoa m bang RSA:
e = 41050406213656285085745498324661128124 1
d = 24170687050291375107547376640345081680929549109938222131434422302499798270804 1
n = 41215881555376575938973898331314270339060454377535462947141130107032977763540 7
c = 35129053540439257259060722045218735825829434111243662846696880566056350107179
Tan cong RSA:
p = 64199596225659064450930317881301806839 7
q = 64199596225659064450930317881301806833 1
d = 24170687050291375107547376640345081680929549109938222131434422302499798270804 1
m = 19127378
```

- TN4: p,q là số nguyên tố 512 bit

- TN5: p,q là số nguyên tố 1024 bit

❖ Kết luận:

- Chỉ cần |p – q| đủ nhỏ thì thuật toán chạy rất tốt, kể cả với số lớn.
- Để tránh cho RSA không bị tấn công phân tích Fermat thì trước khi sử dụng 2 hai số nguyên tố p và q (được sinh ngẫu nhiên), ta nên kiểm tra |p - q| không được quá nhỏ.