RSA Crypto Algorithm

Key Generation (Receiver Side)

1.

$$
\begin{gathered}
p = 3 \\
q = 11
\end{gathered}
$$

2.

$$
\begin{gathered}
n = p \cdot q \\
n = 3 \cdot 11 \\
n = 33
\end{gathered}
$$

3.

$$
\begin{gathered}
\lambda(n) = lcm(p-1, q-1) \\
\lambda(33) = lcm(2, 10) \\
\lambda(33) = 10
\end{gathered}
$$

4.

$$
\begin{gathered}
(\gcd(10, e) = 1) \ \& \ \& (1 < e < 10) \\
e = 3
\end{gathered}
$$

5.

$$
\begin{gathered}
d = e^{-1} \mod \lambda(n) \\
d = 3^{-1} \mod 10
\end{gathered}
$$

- Extended Euclidean Algorithm:

q	a	b	r	t_1	t_2	$t = t_1 - t_2 \cdot q$
3	10	3	1	0	1	-3
3	3	1	0	1	-3	6

$$r = 0 \Rightarrow d = t_2 = -3 \mod 10 = 7$$

6.

در الگوریتم rsa هر فرد دو کلید دارد که یک کلید عمومی است و آنرا در اختیار همه می گذارد، کلید دیگر خصوصی است و باید پیش خود شخص مخفی بماند. افراد دیگر با استفاده از کلید عمومی که دارند می توانند پیام خود را رمز کنند و به شخص مورد نظر بفرستند و آن شخص با استفاده از کلید خصوصی خود می تواند پیام را رمزگشایی کند و هیچ کس به جز او نمی تواند این پیام را رمزگشایی کند چون کلید باز کردن را فقط خود شخص دارد.

$$
\begin{gathered}
\textbf{privateKey} = \{d, n\} = \{7, 10\} \\
\textbf{publicKey} = \{e, n\} = \{3, 10\}
\end{gathered}
$$

با عوض کردن جای کلید عمومی و خصوصی الگوریتم همچنان کار می کند:

$$m = (m^e)^d = (m^d)^e \mod n$$

اما، در این الگوریتم کلید خصوصی باید بزرگ باشد تا قابل حدس یا حمله نباشد و از طرفی کلید عمومی باید کوچک باشد تا هزینه رمزکردن کم باشد. بنابرین الگوریتم با عوض کردن جای کلید ها همچنان کار می کند اما از لحاظ امنیتی مشکل دارد و کاربردی نیست.

Encryption (Sender Side)

Message: $m = 13$
Ciphertext: $c = ?$

$$c = m^e \mod n$$
$$c = 13^3 \mod 33$$
$$c = (13^1 \cdot 13^2) \mod 33$$
$$c = (13 \cdot 4) \mod 33$$
$$c = 52 \mod 33$$
$$c = 19$$

Decryption (Receiver Side)

Ciphertext: $c = 19$
Message: $m = ?$

$$m = c^d \mod 33$$
$$m = 19^7 \mod 33$$
$$m = (19^1 \cdot 19^2 \cdot 19^4) \mod 33$$
$$19^2 \mod 33 = -2$$
$$19^4 \mod 33 = -2 \cdot -2 = 4 \mod 33$$
$$m = (19 \cdot -8) \mod 33$$
$$m = -152 \mod 33$$
$$m = 13$$

Diffie-Helman

1.

$$p = 23$$

2.

با استفاده از کد زیر مولد را پیدا می کنیم:

```python
p = 23
for i in range(1, p): # [1, p-1]
    visited = [False] * p
    trace = []
    k = 1
    counter = 0
    while True:
        if counter == p-1:
            break
        if visited[k] == True:
            break
        visited[k] = True
        k = (k * i) % p
        if k == 0:
            break
        counter += 1
        trace.append(k)
    if counter == p-1:
        print('generator: ', i)
        print(trace)
        break
```

```
generator:  5
[5, 2, 10, 4, 20, 8, 17, 16, 11, 9, 22, 18, 21, 13, 19, 3, 15, 6, 7, 12, 14, 1]
```

3.

در الگوریتم Diff-Hellman باید سعی کنیم احتمال وجود تمام اعداد را در متن رمز شده باقی بگذاریم تا بیشترین میزان امنیت را داشته باشیم اگر مقدار p مولد نباشد آنگاه حمله کننده می تواند آزمون های بسیار کمتری نسبت به حالت مولد بودن آن انجام دهد و رمز ما را بشکند.

4.

$$\alpha = 29$$
$$\beta = 31$$

5.

$$g^{\alpha} \mod p = 5^{29} \mod 23$$
$$= 5^{(11101)_2} \mod 23$$
$$5 \mod 23 = 5$$
$$5^2 \mod 23 = 2$$
$$5^4 \mod 23 = 4$$
$$5^8 \mod 23 = 16$$
$$5^16 \mod 23 = 3$$
$$5^{29} \mod 23 = 5 \cdot 4 \cdot 16 \cdot 3 = 17$$

$$g^{\beta} \mod p = 5^{31} \mod 23$$
$$= 5^{(11111)_2} \mod 23$$

$$
\begin{gathered}
5 \mod 23 = 5 \\
5^2 \mod 23 = 2 \\
5^4 \mod 23 = 4 \\
5^8 \mod 23 = 16 \\
5^{16} \mod 23 = 3 \\
5^{31} \mod 23 = 5 \cdot 2 \cdot 4 \cdot 16 \cdot 3 = 11
\end{gathered}
$$

6.

$$
\begin{gathered}
(g^\alpha)^\beta = 17^{31} \mod 23 \\
= 17^{(11111)_2} \mod 23 \\
17 \mod 23 = -6 \\
17^2 \mod 23 = 13 \\
17^4 \mod 23 = 8 \\
17^8 \mod 23 = 18 \mod 23 = -5 \\
17^{16} \mod 23 = 2 \\
17^{(11111)_2} \mod 23 = -6 \cdot 13 \cdot 8 \cdot -5 \cdot 2 = 7
\end{gathered}
$$

$$
\begin{gathered}
(g^\beta)^\alpha = 11^{29} \mod 23 \\
11 \mod 23 = 11 \\
11^2 \mod 23 = 6 \\
11^4 \mod 23 = 13 \\
11^8 \mod 23 = 8 \\
11^{16} \mod 23 = 18 = -5 \\
11^{29} \mod 23 = 11^{16} \cdot 11^8 \cdot 11^4 \cdot 11^1 \mod 23 \\
= -5 \cdot 8 \cdot 13 \cdot 11 \mod 23 \\
= 7
\end{gathered}
$$

$$7 = 7 :)$$