باسمه تعالی

درس: آزمایشگاه امنیت شبکه

نام و نام خانوادگی: محمد هادی امینی

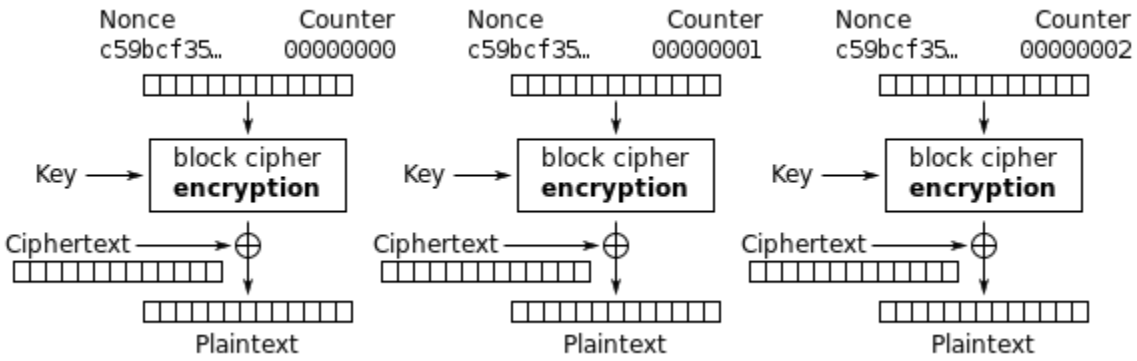
شماره دانشجویی: 9912762370

شماره تمرین : 02

مدیریت کلید (Key Management) در رمزنگاری اساسی است و برای امنیت اطلاعات بسیار حیاتی است. اگرچه معمولاً در تمرکز خود بر روی الگوریتم‌های رمزنگاری تمرکز کرده‌ایم، اما مدیریت صحیح کلیدها نیز بسیار مهم است و هر گونه نقص در این مرحله می‌تواند منجر به شکست امنیتی شود.

در این سناریو مهاجم با دستیابی به کلید رمزنگاری میتواند پیام ها را رمزگشایی کند.

در ابتدا سرور و کلاینت از طریق سوکت به هم دیگر متصل میشوند. سپس سرور یک عدد 16 بایتی را به عنوان کلید و عدد 8بایتی nonce را به کلاینت میفرستد.



تصویر1- توضیحات مربوط به nonce

کلاینت نیز کلید تصادفی خود را ارسال میکند و کلید مشترک از xor این دو کلید به دست می آید.

سپس پیام ها به وسیله الگوریتم AESرمزنگاری می‌شوند.

```
cipher = AES.new(shared_key, AES.MODE_CTR, nonce=nonce)
while True:
    message = input().encode()
    ciphertext = cipher.encrypt(message)
    conn.send(ciphertext)
```

تصویر2- کد رمزنگاری و ارسال

```
def receive_messages(sock, key, nonce):
    cipher = AES.new(key, AES.MODE_CTR, nonce=nonce)
    while True:
        data = sock.recv(1024)
        if not data:
            break
        message = cipher.decrypt(data)
        print("Received: ", message.decode())
```

تصویر3- کد دریافت و رمزگشایی

در هنگام تبادل کلید، مهاجم به وسیله یک اسکریپت پایتون (با استفاده از کتابخانه scapy)کلید را شنود می کند و سپس پیام ها را رمزگشایی میکند.

```
[*] Sniffing packets...
[+] received client key
[+] received server key
[+] received nonce
shared key:939568643323289817828083788867993245994
Server: hi
Client: how are you?
```

تصویر4- خروجی کد مهاجم

برای امن سازی ارتباط از الگوریتم Diffie-Hellman استفاده می کنیم. در این الگوریتم بجای تبادل مستقیم کلید، مقادیر p وg و کلید های عمومی مبادله میشوند. برای امنیت بالا باید کلید با طول زیاد انتخاب شود. P یک عدد اول بزرگ است. g که به آن generatorگفته میشود باید ریشه ابتدایی(primitive root) p باشد.

ریشه ابتدایی یک عدد صحیح مثبت، اولین عددی است که با توان‌هایش، تمامی اعداد مثبت کوچک‌تر از (یک عدد ثابت) را به دست می‌آورد. به عبارت دیگر، اگر gیک عدد صحیح مثبت باشد و ما با توان‌هایش اعدادی متفاوت را به دست بیاوریم، این g را ریشه ابتدایی می‌نامیم.

مثال:

3 یک ریشه اولیه برای 7 است زیرا:

$$
\begin{aligned}
3^1 &= 3^0 \times 3 \equiv 1 \times 3 = 3 \equiv 3 \pmod 7 \\
3^2 &= 3^1 \times 3 \equiv 3 \times 3 = 9 \equiv 2 \pmod 7 \\
3^3 &= 3^2 \times 3 \equiv 2 \times 3 = 6 \equiv 6 \pmod 7 \\
3^4 &= 3^3 \times 3 \equiv 6 \times 3 = 18 \equiv 4 \pmod 7 \\
3^5 &= 3^4 \times 3 \equiv 4 \times 3 = 12 \equiv 5 \pmod 7 \\
3^6 &= 3^5 \times 3 \equiv 5 \times 3 = 15 \equiv 1 \pmod 7
\end{aligned}
$$

برای پیدا کردن عدد اول p با توجه به بزرگ بودن عدد و زمانبر بودن بررسی تمام اعداد، از تست Miller–Rabin primality استفاده می کنیم.

```
Input #1: n > 2, an odd integer to be tested for primality
Input #2: k, the number of rounds of testing to perform
Output: "composite" if n is found to be composite, "probably prime" otherwise
```

```
let s > 0 and d odd > 0 such that n − 1 = 2^s d  # by factoring out powers of 2 from n − 1
repeat k times:
    a ← random(2, n − 2)  # n is always a probable prime to base 1 and n − 1
    x ← a^d mod n
    repeat s times:
        y ← x^2 mod n
        if y = 1 and x ≠ 1 and x ≠ n − 1 then # nontrivial square root of 1 modulo n
            return "composite"
        x ← y
    if y ≠ 1 then
        return "composite"
return "probably prime"
```

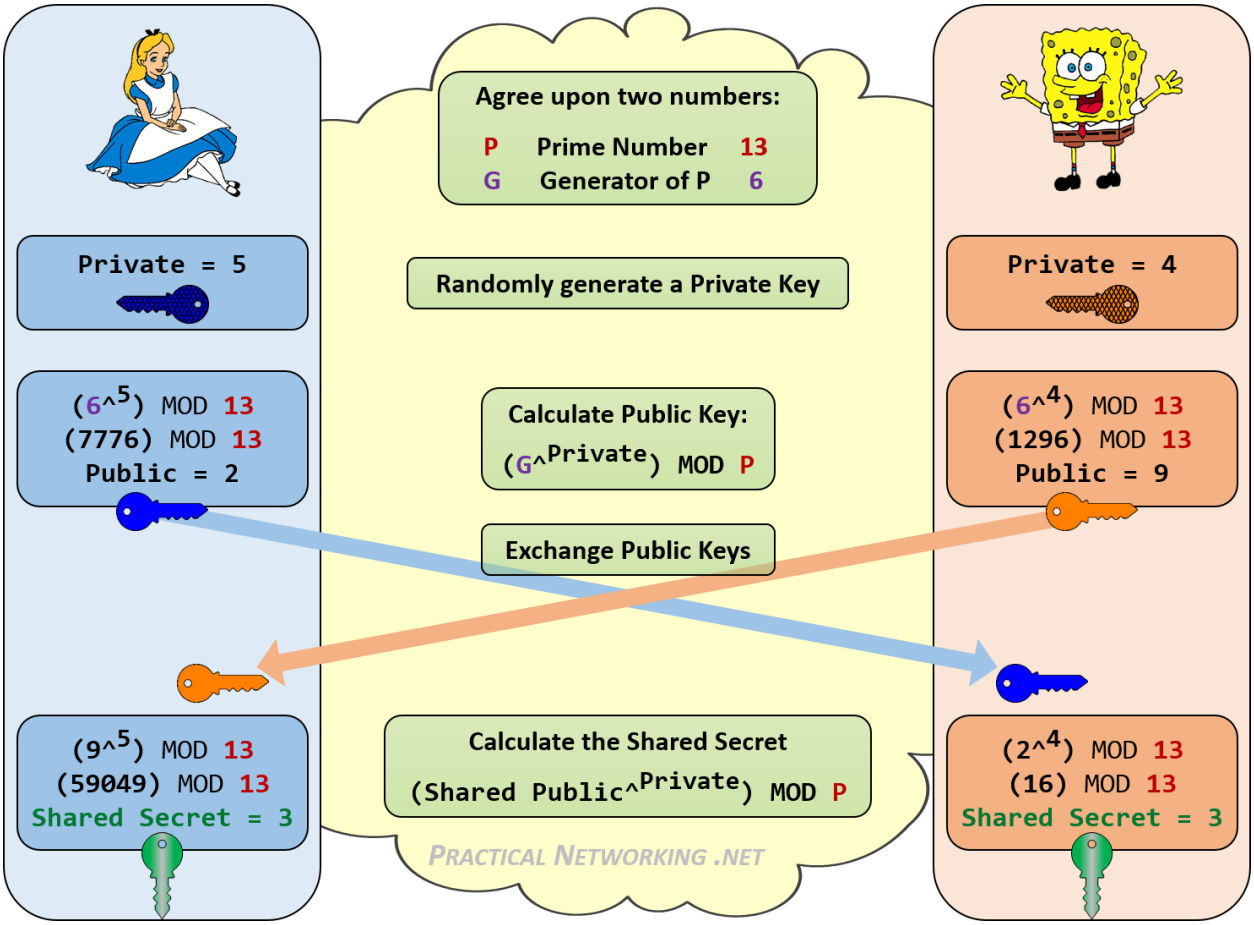
تصویر5- شبه کد الگوریتم میلر-رابین

در صورت موفقیت آمیز بودن این تست، به احتمال زیاد عدد اول است (اگر k تعداد تکرار باشد، خطای آن به اندازه 4^{-k} است) با افزایش تعداد تکرار می‌توان به دقت بسیار بالایی رسید.

مرتبه زمانی این الگوریتم $O(k \log^3 n)$ میباشد.

بعد از توافق p و g ، هرکدام یک کلید خصوصی تصادفی انتخاب می کنند.

سپس مقدار $g^{private} \ mod \ p$ را به طرف مقابل ارسال میکند. اگر A مقدار دریافت شده باشد کلید مشترک از رابطه $A^{private} \ mod \ p$ به دست می آید.



تصویر6 – الگوریتم Diffie-Hellman

امنیت این پروتکل مبتنی بر دشواری حل مسئله لگاریتم گسسته است.

مسئله لگاریتم گسسته همان حل کردن معادله $a^x \equiv a \ mod \ b$ برای مجهول x است و به دلیل کاربردی که در ریاضیات و به خصوص در رمزنگاری پیدا کرده‌است به صورت یک اصطلاح درآمده‌است.

حل کردن مسئله لگاریتم گسسته (محاسبه لگاریتم گسسته) از دیدگاه ریاضی معادل با حل کردن مسئله تجزیه اعداد صحیح در نظر گرفته می‌شود و وجوه اشتراکی بین آن دو وجود دارد:

- هر دو مسئله جزو مسائل دشوار ریاضی هستند، به این معنی که روش سریعی برای حل کردن آن‌ها پیدا نشده است.

- هر الگوریتم مرتبط با یکی از این دو مسئله، قابل تبدیل به الگوریتم مشابهی در ارتباط با مسئله دیگر می‌باشد.
- از دشوار بودن حل هر دو مسئله، برای طراحی و ایجاد سیستم‌های رمزنگاری مختلفی استفاده شده‌است.