تکلیف اول

مجتبی ملائی
۴۰۱۳۱۳۸۳

رسول صالحی
۴۰۱۲۵۹۳۳

۱

محرمانگی (Confidentiality)

- اطمینان از اینکه اطلاعات فقط برای افراد مجاز قابل دسترسی هستند و افراد غیرمجاز نمی‌توانند به آن‌ها دسترسی داشته باشند. (برای مثال در سیستم های بانکی حفظ اطلاعات کاربران برای عدم سرقت مالی باید مورد اهمیت واقع شود)

یکپارچگی (Integrity)

- اطمینان از این که اطلاعات در طول زمان تغییر غیرمجاز نداشته‌اند و همچنان دقیق و کامل هستند. (برای مثال در یک سیستم رای گیری نباید با دستکاری و تغییر آرای ثبت شده باعث نامعتبر بودن نتیجه انتخابات شد)

دسترس‌پذیری (Availability)

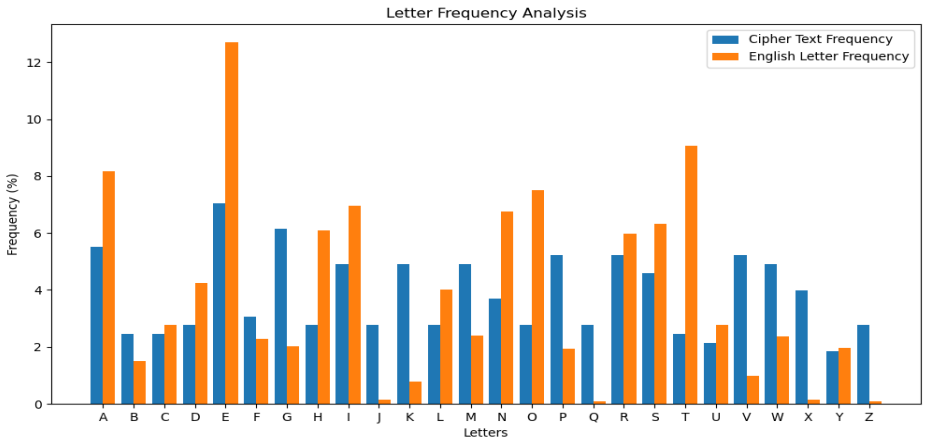
- تضمین اینکه اطلاعات و سیستم‌ها در زمان نیاز برای کاربران مجاز قابل استفاده باشند. (برای مثال در سیستم کنترل هوایی اگر یک پیام دیر و یا اصلا به مقصد نرسد میتواند باعث حوادث هوایی شود)

۲

۱. • تعداد تکرار حروف به ترتیب بیشترین:

E	G	A	P	R	V	M	I	W	K	S	X	N	F	H	O	Q	J	L	D	Z	T	C	B	U	Y
23	20	18	17	17	17	16	16	16	16	15	13	12	10	9	9	9	9	9	9	9	8	8	8	7	6

- بررسی توزیع آماری: خیر با توزیع زبان انگلیسی مشابه نیست.



- اگر چه این روش باعث شده است تا توزیع آماری متن اصلی با متن رمز شده متفاوت بنظر برسد و حملات آماری را تا حدودی دشوار کرده است، اما همچنان توزیع آماری کتاب(کلید) و متن اصلی تاثیر خود را روی توزیع متن رمز شده خواهند گذاشت.

.۲

- برای پیدا کردن کلید از تابع زیر استفاده می‌کنیم:

```
key = ""
for i , j in zip(cipher_text,plain_text):
  if i == ' ':
    continue
  key+= chr((ord(i) - ord(j))%26 + ord('a'))
print(key)
```

که خروجی آن برابر است با: expectopatronumharryyelledtryingtoblotthescreamingfromhisearsexp
با فاصله گذاری مناسب خواهیم داشت:
expecto patronum harry yelled trying to blot the screaming from his ears exp

- متن داده شده مربوط به کتاب Harry Potter and the Prisoner of Azkaban است.
- متن اصلی:

the one time pad otp is an unbreakable encryption method when used correctly it relies on a truly random key that is as long as the message itself and is never reused since each character in the plaintext is combined with a completely unpredictable key character statistical analysis becomes impossible this guarantees perfect secrecy as long as the key remains unknown and is only used once

.۳

- همچنان توزیع کلمات در کتاب های مختلف یکسان است و ترکیب آنها نیز توزیع یکسانی خواهد داشت. پس در برابر حملات آماری تاثیری نخواهد داشت. اما این کار تا حد کمی میتواند امنیت را در برابر مهاجمی که بخواهد با جست و جوی کتاب ها کلید را پیدا کند، افزایش می‌دهد.
- مشکل روش های قبلی غیر تصادفی بودن کلید بود که باعث میشد دارای توزیع های آماری خاصی باشند و در برابر حمله های اماری آسیب پذیر باشند. بنابراین برای جلوگیری از حملات آماری باید کلید کاملا تصادفی باشد.
- از OTP استفاده کنیم. از یک کلید یکبار مصرف کاملا تصادفی استفاده کنیم. همچنین بجای جمع از XOR باید استفاده کنیم.

۳

Step	b0	b1	b2	b3	b4	b5	b6	b7	Output (b0)	Feedback (b7 ⊕ b3)
0	1	0	1	0	1	0	1	0	1	0⊕0=0
1	0	1	0	1	0	1	0	1	0	1⊕1=0
2	0	0	1	0	1	0	1	0	0	0⊕0=0
3	0	0	0	1	0	1	0	1	0	1⊕1=0
4	0	0	0	0	1	0	1	0	0	0⊕0=0
5	0	0	0	0	0	1	0	1	0	1⊕0=1
6	1	0	0	0	0	0	1	0	1	0⊕0=0
7	0	1	0	0	0	0	0	1	0	1⊕0=1
8	1	0	1	0	0	0	0	0	1	0⊕0=0
9	0	1	0	1	0	0	0	0	0	0⊕1=1
10	1	0	1	0	1	0	0	0	1	0⊕0=0
11	0	1	0	1	0	1	0	0	0	0⊕1=1
12	1	0	1	0	1	0	1	0	1	0⊕0=0
13	0	1	0	1	0	1	0	1	0	1⊕1=0
14	0	0	1	0	1	0	1	0	0	0⊕0=0
15	0	0	0	1	0	1	0	1	0	1⊕1=0

محاسبه ۱۶ بیت اول خروجی:

1000001010101000

دوره تناوب:

- بله الگوی خروجی در حال تکرار است که با توجه به خروجی بعد از بیت ۱۲ دوباره تکرار شده پس دوره تناوب آن ۱۲ است.

دلیل ناامن شدن رمز:

این اشتباه باعث ناامن شدن رمز می‌شود زیرا:

- **دوره تناوب کوتاه:** دوره تناوب LFSR با فیدبک اشتباه $b3 \oplus b7$ بسیار کوتاه است (۱۲ بیت). این باعث می‌شود الگوی خروجی به سرعت تکرار شود و مهاجم بتواند به راحتی الگو را تشخیص دهد.
- **ضعف در رندوم سازی خروجی:** LFSR با فیدبک صحیح $b1 \oplus b7$ دوره تناوب طولانی‌تری دارد و الگوی خروجی آن تصادفی‌تر به نظر می‌رسد. اما با فیدبک اشتباه، الگوی خروجی قابل پیش‌بینی می‌شود.
- **ملاحظات امنیتی:** در طراحی LFSR، باید از فیدبک‌هایی استفاده شود که دوره تناوب را به حداکثر برسانند و الگوی خروجی غیرقابل پیش‌بینی باشد. در این حالت، فیدبک $b3 \oplus b7$ به دلیل دوره تناوب کوتاه، امنیت سیستم را کاهش می‌دهد.

۴

پیش فرض های مسئله

- $p = 3$
- $q = 11$
- $n = p \times q = 3 \times 11 = 33$
- $e = 3$ (کلید عمومی)
- $d = 7$ (کلید خصوصی)

۱. رمزنگاری پیام‌های $m = 2$ و $m = 3$:
فرمول رمزنگاری در RSA به این صورت است:

$$c = m^e \mod n \quad (۱)$$

برای $m = 2$:

$$c = 2^3 \mod 33 = 8 \quad (۲)$$

برای $m = 3$:

$$c = 3^3 \mod 33 = 27 \quad (۳)$$

پس مقادیر رمزنگاری‌شده به این صورت هستند:

- $m = 2 \rightarrow c = 8$
- $m = 3 \rightarrow c = 27$

همچنین برای امضای آنها داریم:
فرمول امضا در RSA به این صورت است:

$$s = m^d \mod n \quad (۴)$$

برای $m = 2$:

$$s = 2^7 \mod 33 = 128 \mod 33 = 29 \quad (۵)$$

برای $m = 3$:

$$s = 3^7 \mod 33 = 2187 \mod 33 = 9 \quad (۶)$$

پس امضاها به این صورت هستند:

- $m = 2 \rightarrow s = 29$
- $m = 3 \rightarrow s = 9$

بخش دوم سوال:
قسمت اول:

$$(2^7) \times (3^7) \mod 20 = 16 = S_f \quad (۷)$$

قسمت دوم:

$$m_6 = 6^7 \mod 20 = 16 == S_f \quad (۸)$$

قسمت سوم:

$$(x^7) \times (y^7) \mod 20 = (x \times y)^7 \mod 20 \quad (۹)$$

در سمت چپ چون پایه‌ها با هم برابرند، در نتیجه با سمت راست یکی شده و باقی‌مانده آن‌ها بر ۲۰ نیز یکسان خواهد بود.
قسمت چهارم:
حداکثر عددی که کمتر از ۳۳ باشد و از ۲ و ۳ تشکیل شده باشد عدد ۲۷ است، در نتیجه ماکسیمم ضرر:

$$3 - 27 = 24 \quad (۱۰)$$

چون ما فقط با داشتن امضای s_3 توانستیم به این برسیم.
قسمت پنجم:
برای جلوگیری از آن می‌توان یک عدد رندوم جنریت کرد و به انتهای پیام اصلی اضافه و کل پیام را امضا کنیم که به این عدد رندوم nonce می‌گویند و همچنین مقدار n را بزرگتر کنیم.

۵

- **پایگاه داده شماره‌های ملی:** به علت داده‌هایی با شماره‌های اغلب مشابه **ECB** مناسب نمی‌باشد و همچنین چون نیاز به حفظ ترتیب دارد **CTR** نیز مناسب نیست و غیر بهینه است، پس بهترین حالت **CBC** است که با یک **IV** باعث محرمانگی دیتا می‌شود.
- **استریم زنده ویدیو:** چون نیاز به عدم تأخیر داریم پس از **CBC** نمی‌توان استفاده کرد چون وابسته به بلوک قبلی خود می‌باشد و از طرفی **ECB** باعث می‌شود دیتاهای مشابه در تصویر را لو دهد، پس بهترین گزینه برای انتخاب **CTR** است که سریع است و امکان رمز کردن موازی را دارد.
- **تصویر بیت‌مپ سیاه و سفید:** به علت داده‌هایی با نواحی بزرگ یکنواخت، **ECB** مناسب نمی‌باشد زیرا الگوهای تصویر را مخفی نمی‌کند و همچنین چون نیاز به حفظ ترتیب دارد، **CTR** نیز مناسب نیست و غیر بهینه است، پس بهترین حالت **CBC** است که با یک **IV** باعث محرمانگی دیتا می‌شود.
- **برنامه پیام‌رسانی با رمزگذاری سرتاسری:** در این حالت **CBC** نامناسب است زیرا رمزگشایی هر پیام به قبلی‌های آن وابسته می‌شود و همچنین اگر پیام‌ها شامل متن‌های تکراری باشند، **ECB** برای آن ناامن می‌شود. پس بهتر است از **CTR** که مستقلاً رمز و رمزگشایی می‌کند و سریع است به علت موازی بودنش و نیازی هم به **padding** ندارد، استفاده کنیم.
- **ویرایش فایل رمز شده:** می‌توان از **ECB** یا **CTR** برای این کار استفاده کرد، چون به نسبت **CBC** هر دو سریع‌تر هستند و نیازی به رمزگشایی کل فایل برای ویرایش قسمتی از آن نیست. ولی از بین دو موردی که گفتیم، برای فایل‌های بزرگ باز هم **CTR** گزینه مناسب‌تر و امن‌تری نسبت به **ECB** است.

۶

۱. چون از AES در مود **ECB** استفاده می‌کنیم می‌دانیم که رمز شده هر بلاک جدا از بقیه بلاک ها است. همچنین ساختار HMAC گفته شده به **attack extension Length** آسیب پذیر است. بنابراین بدون دانستن کلید می‌توانیم یک هش معتبر جدید برای وقتی که متنی به پیام اضافه می‌کنیم بسازیم. حال می‌توانیم رمز شده reciever اول یا مقدار amount یا sender را به انتهای پیام اضافه کنیم(به عنوان یک .(receiver سپس هش آن را با استفاده از تنکیک گفته شده افزایش دهیم. بدین صورت می توانیم اطلاعات دریافت کنندگان را تغییر دهیم. همچنین اگر پیام دومی با همان کلید رمز و هش شود، می توانیم گیرندگان پیام اول را به گیرندگان پیام دوم اضافه کنیم، زیرا نشان دادیم که میتوانیم یک بلاک رمز شده به پیام ها اضافه کنیم و یک هش معتبر برای آن بسازیم.

۲. حمله طول-افزایی **Length Extension (Attack)** زمانی ممکن است که از یک تابع هش مانند SHA-256 با ساختار نامناسب **HMAC = Hash(Key || Message)** استفاده شود. در این روش، مهاجم می‌تواند بدون دانستن کلید، مقدار جدیدی به پیام اصلی اضافه کند و هش معتبری برای پیام تغییریافته تولید کند. دلیل این موضوع به نحوه‌ی کارکرد توابع هش مانند SHA-256 برمی‌گردد، که در آن مقدار هش نهایی نه تنها به ورودی بلکه به حالت داخلی **(Internal State)** تابع هش نیز وابسته است. وقتی مهاجم مقدار هش پیام اولیه را داشته باشد، می‌تواند از این مقدار به عنوان وضعیت میانی تابع هش استفاده کند و یک ادامه‌ی جدید به پیام اضافه کند، سپس هش جدیدی تولید کند که معتبر خواهد بود. این به مهاجم امکان می‌دهد که اطلاعات جدیدی مانند گیرندگان اضافی را بدون داشتن کلید مخفی به پیام اصلی اضافه کند و همچنان یک **HMAC** معتبر ارائه دهد.

۳. یک حمله دیگر این است که می‌توانیم در این اپلیکیشن عضو شویم و چند تراکنش انجام دهیم. چون به رمز شده پیام ها دسترسی داریم میتوانیم مقدار رمز شده گیرنده ها و amount و sender دسترسی داشته باشیم. بدین ترتیب وقتی تراکنشی انجام بگیرد که گیرنده یا ارسال کننده اش یکی از گیرنده هایی باشد که ما مقدار رمز شده آن را داریم، میتوانیم آن تراکنش را تشخیص دهیم. همچنین مقدار amount های خاص مثلا $10 را میتوانیم بدست آوریم و با خواندن پیام های رمز شده آنها را تشخیص بدهیم. به ترتیب اصل محرمانگی داده ها برای این پروتکل دیگر برقرار نیست زیرا می‌توان تراکنش ها را تشخیص داد.

۴.
- از **AES-GCM** استفاده کنیم.
- از HMAC زیر استفاده کنیم:

$$\text{HMAC(Key, Message)} = \text{Hash}((\text{Key} \oplus \text{opad}) \,||\, \text{Hash}((\text{Key} \oplus \text{ipad}) \,||\, \text{Message}))$$

۷

۱. نشان دهید که حمله (MitM) می‌تواند برای شکستن رمزگذاری دوگانه استفاده شود و امنیت AES را به میزان قابل توجهی کاهش دهد.

- مهاجم ابتدا به یک متن رمز شده و همچنین خود متن رمز نشده آن دست می‌یابد.
- سپس مهاجم متن رمز نشده را با تمامی حالات K_1 رمز می‌کند.
- و همچنین متن رمز شده را با تمامی حالات K_2 رمزگشایی می‌کند.
- مهاجم به محض یافتن متن مشابه در قسمت‌های ۲ و ۳، کلیدهایی که با آن‌ها به این متن مشترک رسیده را به عنوان کلیدهای K_1 و K_2 شناسایی می‌کند.

امنیت رمزگذاری دوگانه به طور قابل توجهی به همان امنیت یگانه رمزگذاری کاهش می‌یابد زیرا مهاجم می‌تواند به طور مستقل به هر لایه حمله کند. پیچیدگی زمانی حمله 2^n است (که در آن n اندازه کلید است)، که بسیار کمتر از 2^{2n} مورد انتظار برای رمزگذاری است.

۲. توضیح دهید که AES Triple چگونه کار می‌کند و چرا در برابر حملات مرد میانی ایمن‌تر است.

AES Triple (3-AES):

- در این روش، داده‌ها سه بار با سه کلید متفاوت رمزگذاری می‌شوند.
- این ساختار به عنوان **Encrypt-Decrypt-Encrypt (EDE)** شناخته می‌شود.

ایمنی در برابر حملات MitM:

- در برابر حملات MitM ایمن‌تر است، زیرا:
  - برای شکستن AES، Triple مهاجم باید هر سه لایه رمزگذاری را به طور همزمان بشکند.

- این کار به دلیل افزایش پیچیدگی محاسباتی، عملاً غیرممکن است.
- همچنین، AES Triple از طول کلید مؤثر بیشتری استفاده می‌کند (معمولاً ۱۶۸ بیت)، که امنیت را به طور قابل توجهی افزایش می‌دهد.

۳. از نظر امنیت و کارایی، کدام روش ترجیح داده می‌شود؟ AES-256 را با AES Triple مقایسه کنید.

AES-256:

- از یک کلید ۲۵۶ بیتی استفاده می‌کند.
- امنیت آن در برابر حملات کلاسیک بسیار بالا است.
- در برابر حملات کوانتومی، امنیت آن به حدود ۱۲۸ بیت کاهش می‌یابد (به دلیل الگوریتم‌های کوانتومی مانند Grover’s Algorithm).

Triple AES:

- از سه کلید مستقل استفاده می‌کند (معمولاً ۱۶۸ بیت).
- امنیت آن در برابر حملات کلاسیک و کوانتومی بسیار بالا است.
- در برابر حملات کوانتومی، امنیت آن به حدود ۸۴ بیت کاهش می‌یابد (به دلیل Grover’s Algorithm).

مقایسه امنیت و کارایی:

- **امنیت:**
  - از لحاظ امنیت، AES Triple امنیت بیشتری نسبت به AES-256 دارد، زیرا از سه لایه رمزگذاری استفاده می‌کند.
  - اما در برابر حملات کوانتومی، امنیت AES Triple کمی کمتر از AES-256 است (۸۴ بیت در مقابل ۱۲۸ بیت).
- **کارایی:**
  - از نظر کارایی AES-256 بهتر است، زیرا فقط یک لایه رمزگذاری دارد.
  - ولی AES Triple به دلیل سه لایه رمزگذاری، کندتر است و منابع بیشتری مصرف می‌کند.