به نام خدا

دانشکده مهندسی کامپیوتر

تمرین دوم درس مبانی امنیت اطلاعات

فصل دوم رمزنگاری متقارن

استاد درس: دکتر شهریاری

فهرست مطالب

تمرین تشریحی

سوال 1 (5 نمره)

الگوریتم DES چگونه عمل می‌کند؟ مراحل رمزنگاری در هر دور از DES را با جزئیات و ترسیم نمودار مربوطه توضیح دهید.

سوال 2 (5 نمره)

نشان دهید که ویژگی مکمل پذیری[1] در DES برقرار است. در مورد DES خاصیت مکمل پذیری به شکل زیر تعریف می شود[2]:

$$C = DES(P, K) \Leftrightarrow \overline{C} = DES(\overline{P}, \overline{k})$$

[1] complementarity

[2] برای راهنمایی بیشتر میتوانید از مطالب این صفحه کمک بگیرید.

سوال 3 (5 نمره)

عملکرد و ویژگی‌های مدهای کاری **ECB**، **CBC** ، **CFB**، **OFB** و **CTR** را بررسی و به سوالات زیر پاسخ دهید.

1. مد کاری **CBC** چگونه کار می‌کند و چرا نسبت به **ECB** امنیت بیشتری دارد؟
2. مد کاری **ECB** چرا استفاده از آن توصیه نمی‌شود و در چه شرایطی ممکن است خطرناک باشد؟
3. مد کاری **CFB** چه تأثیری در انتقال خطا دارد و چگونه داده‌ها را رمزگذاری می‌کند؟
4. مد کاری **OFB** چه تفاوتی با **CFB** در نحوه انتقال خطا دارد و چگونه با خطاهای انتقال مقابله می‌کند؟
5. مد کاری **CTR** چرا این مد مناسب برای سیستم‌هایی است که نیاز به سرعت بالا دارند و چه ویژگی خاصی دارد؟
6. نمودارهای مدهای کاری **OFB**، **CTR** و **CFB** را رسم کنید.

تمرین عملی

در این قسمت الگوریتم DES[3] که با آن در اسلایدهای درس تا حدی آشنا شدید را پیاده‌سازی می‌کنیم. برای راهنمایی و همچنین توضیحات دقیق‌تر می‌توانید به این لینک مراجعه کنید. برای سادگی در پیاده‌سازی از Jupyter Notebook استفاده خواهیم کرد که بتوانیم مرحله به مرحله پیاده‌سازی الگوریتم را کامل کنیم. این notebook در این لینک قابل دسترسی است.

به صورت کلی، ترتیب مراحل به شکل زیر است:

1. Initial Permutation
2. Key Generation
    a. Key initial permutation
    b. Key splitting & shifting
    c. Key compression
    d. Sub-key generation
3. Rounds Of Encryption
    a. Expansion/Permutation (EP)
    b. S-box
    c. P-box
    d. Special Function (xor)
4. Final Permutation

که هر کدام از قسمت ها را به صورت خلاصه توضیح خواهیم داد[4].

نکته‌ی بسیار مهم: برای ساده سازی؛ فرض میکنیم ورودی ما صرفا یک بلاک ۶۴ بیتی است. لازم نیست حالتی را پیاده‌سازی کنید که ورودی ما بیشتر از ۶۴ بیت است.

[3] Data encryption standard

[4] برای توضیحات بیشتر و پیدا کردن جدول های لازم برای پیاده سازی حتما به لینک داده شده مراجعه فرمایید.

Initial permutation

در این قسمت ورودی، یک درهم ریزی[5] اولیه خواهد داشت. این درهم ریزی بر اساس جدول از پیش تعیین شده ای انجام می شود.

توجه: این قسمت **تنها یک بار** اجرا می شود و به ازای هر مرحله **نیست**.

Key Generation

در این قسمت به سراغ تولید کلید می‌رویم. به صورت کلی ورودی الگوریتم یک کلید ۶۴ بیتی است که تمام بیت‌هایی که اندیس آن‌ها مضربی از ۸ است را نادیده گرفته و یک کلید ۵۶ بیتی درست خواهیم کرد.

توجه: این قسمت هم **تنها یک بار** اجرا می‌شود و به ازای هر مرحله **نیست**.

Key splitting & shifting

در مرحله‌ی بعد کلید ۵۶ بیتی اولیه را به دو قسمت ۲۸ بیتی تبدیل کرده. هر تکه‌ی ایجاد شده یک یا دوبار، مطابق با جدول از پیش تعیین شده ای به ازای هر راند، شیفت داده می شود.

به طور مثال اگر فرض کنید جدول shift به این شکل است:

$$\text{shift\_schedule} = [1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1]$$

اگر در راند اول باشیم یک بار شیفت به راست می‌دهیم یا اگر در راند ۱۵ باشیم ۲ بار شیفت به راست می‌دهیم.

Key compression

حال دو قسمت شیفت خورده‌ی کلید را به هم چسبانده و بعد برای هر راند و مطابق با جدول از پیش تعیین شده ای یک subkey تولید خواهیم کرد که در عملیات رمزنگاری از آن استفاده خواهیم کرد.

[5] permutation

*** **توجه داشته باشید که** مراحل key splitting & shifting و key compression به ازای هر راند انجام خواهند شد.

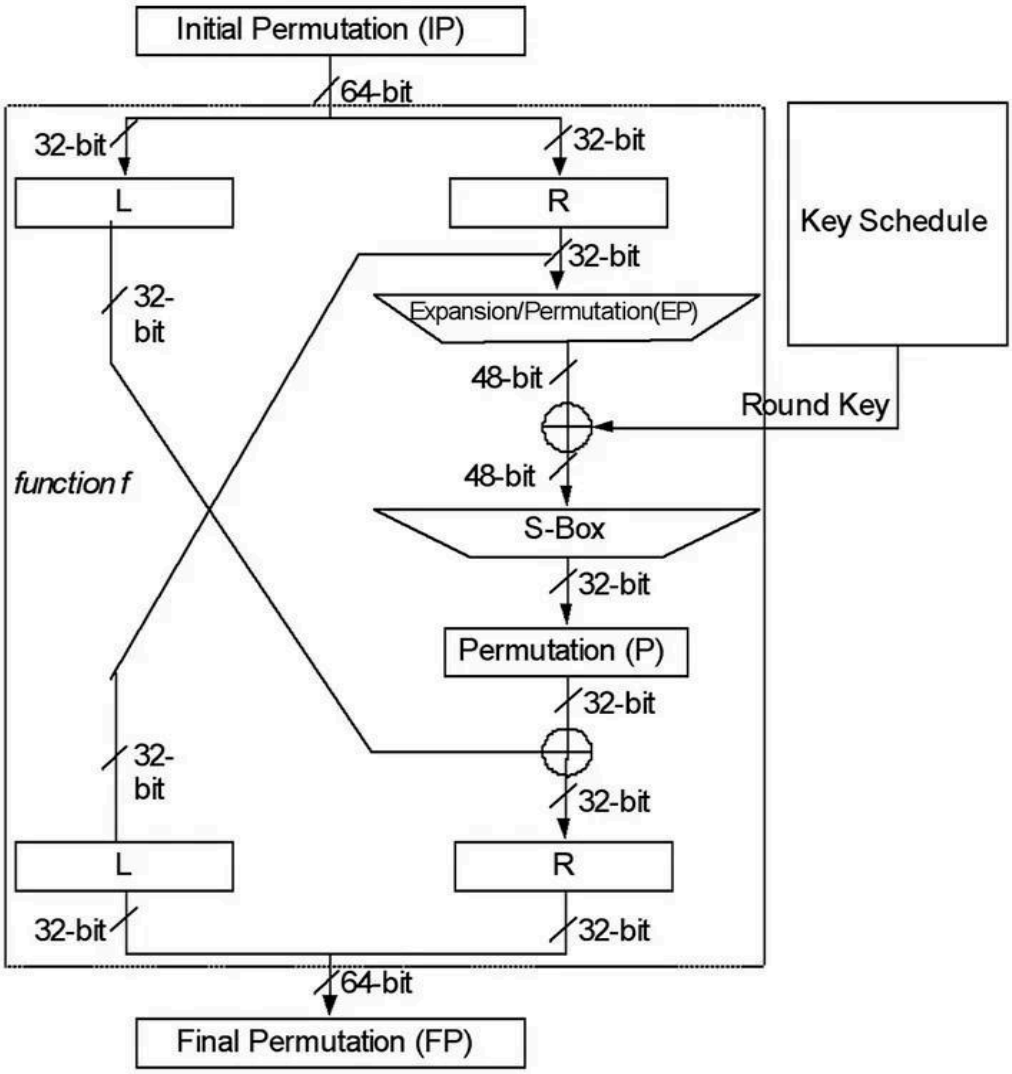
حال که subkey خود را بدست آورده‌ایم؛ می‌توانیم به قسمت اصلی رمزنگاری برویم.

Rounds of encryption

در اینجا قصد داریم برای **۱۶ راند** عملیات زیر را انجام دهیم:

- ورودی را به ۲ تکه‌ی ۳۲ بیتی تبدیل کرده می‌کنیم.
- تکه‌ی راست را با استفاده از EP تبدیل به ۴۸ بیت می‌کنیم.
- خروجی قسمت EP را به تکه های ۶ بیتی تبدیل کرده و هر تکه را به s-box[6] ورودی می‌دهیم.
- خروجی s-box را با استفاده از p-box بیشتر درهم می‌ریزیم.
- خروجی p-box را با ۳۲ بیت سمت چپ ورودی xor کرده و در قسمت راست ورودی راند بعد قرار می‌دهیم.
- ۳۲ بیت سمت راست را در نیمه‌ی سمت چپ ورودی راند بعد، بدون هیچ تغییری، کپی می‌کنیم.

[6] توجه داشته باشید که جدول sbox در لینک داده شده وجود ندارد؛ میتوانید از این لینک استفاده کنید.



*** **توجه داشته باشید که باید** ۱۶ مرحله باید این روند را تکرار کنید و شکل داده شده صرفا یک مرحله را نشان می‌دهد.

Final Permutation

بعد از راند شانزدهم جای نصفه‌ی راست و نصفه‌ی چپ یکبار عوض شده و در نهایت مطابق جدول از قبل تعیین شده‌ی دیگری درهم ریزی آخر انجام شده و خروجی الگوریتم آماده می شود.

توضیحاتی در مورد s-box

در مرحله‌ی Round Of Encryption یکی از مراحل نیازمند استفاده کردن از جدول s-box است که توضیحات آن در این لینک آمده است. برای راهنمایی بیشتر به این جدول توجه کنید:

S_5		Middle 4 bits of input															
		0000	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**
Outer bits	**00**	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

ورودی s-box تعدادی بسته‌ی ۶ بیتی است. s-box برای هر ورودی با توجه به این جدول یک خروجی ۴ بیتی تولید کرده و در نهایت تمام خروجی‌ها را بهم می‌چسبانیم که خروجی ۳۲ بیتی s-box تولید شود.

نحوه‌ي کار جدول هم به شکل زیر است: ابتدا بیت MSB[7] و LSB[8] هر ورودی را برداشت کرده و یک 2 بیتی از آن می سازیم. در مرحله‌ی بعد با توجه به ۴ بیت میانی و دوبیتی جدید ساخته شده در جدول خروجی را پیدا می‌کنیم.

برای مثال فرض کنید ورودی به جدول رشته‌ی "011011" است. داریم که:

Most significant bit: 0

Least significant bit: 1

=> Outer bits: 01 (I)

4 middle bits: 1101 (II)

(I), (II) => output: 1001

راهنمایی: برای پیاده‌سازی جدول‌ها در کد می‌توانید از روش های زیر استفاده کنید:

[7] Most significant bit
[8] Least significant bit

- If statement
- Switch Case
- Dictionaries

استفاده از هر کدام از این روش‌ها اختیاری است ولی استفاده از dictionaryها برای سادگی کد پیشنهاد می‌شود.

سوالات امتیازی

سوال اول

شاید تا بحال متوجه شده باشید که در بسیاری از الگوریتم های رمزنگاری (مانند AES، DES و...) از تابع XOR استفاده زیادی می شود. چه دلایلی برای این موضوع به نظر شما وجود دارد.

سوال دوم

از ریاضیات دوران دبیرستان به یاد داریم، شرط کافی و لازم برای برگشت پذیری یک تابع یک به یک بودن آن است. نشان دهید که DES یک تابع یک به یک است.[9]

سوال سوم

فرض کنید قصد داریم که از بین یکی از مدهای کاری DES یکی را بجز ECB برای پیاده‌سازی به صورت موازی انتخاب کنیم. به نظر شما کدام یک از این مدهای کاری speedup خوبی نسبت به پیاده سازی سریال خود خواهند داشت؟

[9] برای این سوال کافی است که مرحله به مرحله هر یک از عملیات های DES را بررسی کنید و یک به یک بودن هر مرحله را استدلال کنید و در نهایت استدلال کنید که کل تابع یک به یک است.

موفق باشید.