پروژه‌ی اول درس برنامه‌سازی پیشرفته

رمزنگاری فایل (File Cryptography)

- **زمان تحویل از طریق بارگذاری در Moodle: دوشنبه ۹۶/۱/۱۴ ساعت ۲۳:۵۵ (عدم بارگذاری در زمان مقرر به منزله‌ی عدم تحویل پروژه بوده و نمره‌ی پروژه صفر در نظر گرفته می‌شود)**
- **تحویل حضوری در سایت دانشکده: چهارشنبه ۹۶/۱/۱۶ ساعت ۹ الی ۱۱**

در این پروژه می‌خواهیم برنامه‌ای بسازیم که بتوان به کمک آن بر روی فایل رمزگذاری (Encryption) انجام داد و همچنین فایلی که رمزگذاری کرده‌ایم را رمزگشایی (Decryption) نماییم. برای انجام رمزنگاری دو الگوریتم مختلف در نظر گرفته شده‌است که می‌بایست هر دوی آن‌ها را پیاده‌سازی نمایید. در ادامه، مراحل انجام کار به ترتیب شرح داده خواهد شد.

۱. فاز اول، فرآیندهای Encode و Decode فایل

۱-۱. فرآیند Encode

در فاز نخست قصد داریم یک فایل را خوانده و آن را به متن تبدیل نماییم. فایل مورد نظر از هر نوعی می‌تواند باشد. (عکس، فیلم، موسیقی، متنی و ...) در ابتدا برای این کار، فایل ورودی به صورت آرایه‌ای از بایت (byte[]) خوانده می‌شود. همان‌طور که در شکل زیر (شکل ۱) مشاهده می‌گردد، این بایت‌های کنار هم قرار گرفته را می‌توانیم به صورت جریانی از بیت‌های 0 و 1 در نظر بگیریم. سپس از ابتدا شروع کرده و هر شش بیت متوالی را به مقدار عددی آن در مبنای ۱۰ تبدیل می‌نماییم. واضح است که این عدد، مقداری بین ۰ تا ۶۳ خواهد داشت.

source bytes	77 (0x4d)								97 (0x61)								110 (0x6e)								...
bit pattern	0	1	0	0	1	1	0	1	0	1	1	0	0	0	0	1	0	1	1	0	1	1	1	0	...
Index	19								22								5								...
encoded-characters	T								W								F								...

شکل ۱

در ادامه، کاراکتر معادل با هر عدد به‌دست آمده را در جدول زیر (جدول ۱) یافته و با قرار گرفتن این کاراکترها در کنار یکدیگر، فایل اولیه به جریانی از کاراکترها تبدیل می‌شود.

Value	Char	Value	Char	Value	Char	Value	Char
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

جدول ۱

*** چالش:** هنگامی‌که ۶ بیت، ۶ بیت، داده را پردازش کرده و پیش می‌بریم، در بایت انتهایی ممکن است با مشکل مواجه شویم. اگر تعداد بایت‌های فایل ورودی مضربی از ۳ باشد، انتهای ۶ بیت آخر در پردازش با انتهای بایت آخر، هماهنگ و یکسان خواهد بود. (با توجه به شکل ۱: ۶×۴ = ۸×۳) در حالتی که تعداد بایت‌های فایل ورودی مضربی از ۳ نباشد، ۲ یا ۴ بیت اضافه می‌ماند و می‌بایست آن‌ها را برای حفظ صحت تبدیل آن‌ها را به گونه‌ای همانند داده‌های قبلی به کاراکتر تبدیل کنیم. چگونه می‌توانیم این مشکل را حل کنیم؟

۱-۲. فرآیند Decode:

فرآیند Decode معکوس فرآیند Encode خواهد بود. به این معنا که داده‌ی متنی را دریافت نموده و می‌خواهیم آن‌را کاراکتر به کاراکتر، مطابق با جدول ۱ به اعدادی بین ۰ تا ۶۳ تبدیل و سپس با کنار هم قرار دادن مقدار بیتی آن‌ها، این جریان بیتی را ۸ بیت، ۸ بیت به بایت تبدیل نماییم.

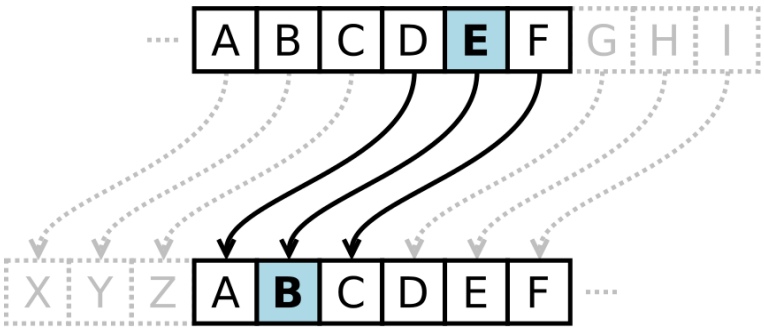
*** چالش:** لازم به ذکر است که راه‌کار درنظر گرفته شده برای انتهای فایل در انتهای فرآیند Encode می‌بایست در انتهای فرآیند Decode برگشت‌پذیر باشد تا هیچ بیتی از داده از دست نرود.

۲. فاز دوم، رمزنگاری

به‌طور کلی هر الگوریتم رمزنگاری از دو بخش رمزگذاری و رمزگشایی تشکیل می‌گردد. در فرآیند رمزگذاری داده‌ی ورودی به‌گونه‌ای تغییر شکل می‌دهد که نتوان آن‌را به راحتی به شکل اولیه بازگردانی نمود. هر چه این بازگردانی سخت‌تر انجام شود، امنیت داده‌ی رمزگذاری شده بیشتر خواهد بود. معمولا داده‌ای پنهان در الگوریتم‌های رمزنگاری وجود دارد که از اهمیت بالایی برخوردار است و آن کلید رمزنگاری است. این کلید که در برخی موارد آن‌را با نام گذرواژه (password) می‌شناسیم، عاملی است که تغییر شکل‌های متفاوت داده‌ی اولیه را به‌وجود می‌آورد و سبب می‌شود که بتوان آن را به شکل اولیه برگرداند.

۲-۱. رمزنگاری جابجایی ساده

این الگوریتم بر روی داده‌های متنی کار می‌کند و در آن قصد داریم در فرآیند رمزگذاری با جابجا نمودن حروف، متن اولیه را تغییر شکل دهیم و آن‌را به‌هم بریزیم. برای این کار در حالت ساده فرض کنیم که متنی با حروف A تا Z (حروف بزرگ) داریم. همه‌ی حروف موجود در متن را به اندازه‌ای ثابت (k) در ترتیب حروف حرکت می‌دهیم و حرف جدید را جایگزین آن می‌نماییم. در شکل ۲ این جابجایی را به سمت راست و به ازای k=3 می‌بینیم و توسط آن تمامی حروف D با A، تمامی حروف E با B و ... جایگزین می‌گردند.



شکل ۲

در فرآیند رمزگشایی، کافی است تا جابجایی حروف را در جهت عکس انجام داده تا متن تغییر شکل داده به متن اولیه تبدیل گردد. بدیهی است که اگر این جابجایی دقیقا به همان اندازه‌ی جابجایی در فرآیند رمزگذاری نباشد، این تبدیل به متن اولیه به درستی انجام نخواهد شد. بنابراین این میزان جابجایی (k) همان کلید رمزنگاری است که در هر دو فرآیند استفاده می‌گردد. (مفهوم رمزنگاری متقارن) در حالتی که متن تنها شامل حروف A تا Z باشد، بدیهی است که کلید مقداری بین ۱ تا ۲۵ می‌تواند داشته باشد.

در مثال زیر نمونه‌ای از فرآیند رمزگذاری به ازای کلید k=3 توسط این روش قابل مشاهده است:

Plain	A	T	T	A	C	K	F	R	O	M	S	O	U	T	H	E	A	S	T
Shift Count	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Cipher	X	Q	Q	X	Z	H	C	O	L	J	P	L	R	Q	E	B	X	P	Q

۲-۲. رمزنگاری جابجایی پیچیده

روش دومی که برای رمزنگاری در این پروژه در نظر گرفته شده، دارای کلید و الگوریتم پیچیده‌تری نسبت به آن‌چه که در روش جابجایی ساده گفته شد، می‌باشد. در این روش میزان جابجایی برای همه‌ی حروف یکسان نیست. کلید (k) در این روش رشته‌ای از حروف است و می‌تواند هر طولی داشته باشد.

فرض کنیم همانند مثال روش قبلی برای سادگی، فقط از حروف بزرگ استفاده نماییم و کلید در نظر گرفته شده برای رمزنگاری کلمه‌ی POINT باشد. جایگاه حروف این کلمه در ترتیب حروف به صورت زیر خواهد بود:

Character	P	O	I	N	T
Index	16	15	9	14	20

در این‌صورت جابجایی حروف برای به‌هم ریختن متن ورودی در فرآیند رمزگذاری همانند مثال زیر انجام خواهد شد:

Plain	A	T	T	A	C	K	F	R	O	M	S	O	U	T	H	E	A	S	T
Shift Count	16	15	9	14	20	16	15	9	14	20	16	15	9	14	20	16	15	9	14
Cipher	I	E	K	M	I	U	Q	I	A	S	C	Z	L	F	N	O	L	J	F

بدیهی است که در فرآیند رمزگشایی، می‌بایست جابجایی‌ها در جهت معکوس انجام گردد.

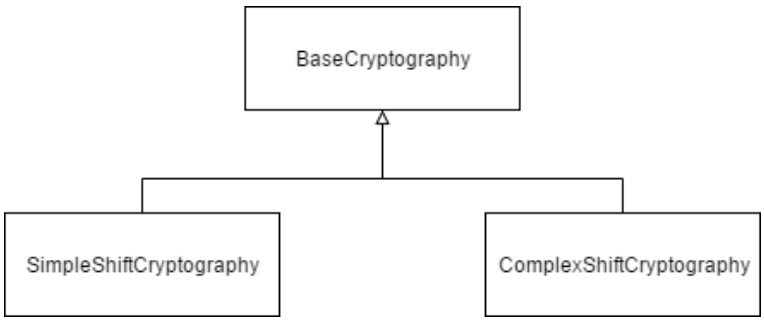
۳. پیاده‌سازی

۳-۱. معماری کلاس‌ها

برای عملیات‌های Encode و Decode می‌بایست یک کلاس با نام Coding بنویسید که وظیفه‌ی آن انجام این دو عملیات است. همچنین می‌بایست یک کلاس با نام InputFileReader برای خواندن از فایل و یک کلاس با نام OutputFileWriter برای نوشتن در فایل ایجاد نمایید.

برای هر کدام از روش‌های رمزنگاری جابجایی ساده و پیچیده نیز می‌بایست کلاسی جدا تعریف کنید که از یک کلاس پایه (Abstract) ارث‌بری می‌نمایند. تمامی متغیرها و متدهای مشترک می‌بایست در کلاس پایه قرار گیرند. کلاس پایه (پدر) می‌بایست حتما شامل دو متد abstract با امضای زیر باشد که در کلاس‌های فرزند متناسب با روش رمزنگاری پیاده‌سازی می‌گردند:

```
public abstract String encrypt(String plainText);
public abstract String decrypt(String cipherText);
```



۳-۲. روال منطقی اجرای برنامه

روال منطقی فرآیندهای رمزگذاری و رمزگشایی به‌صورت زیر خواهد بود: (به عنوان مثال بر روی فایلی با نام IMG0012.jpg)

Encryption Procedure:
Input File (IMG0012.jpg) → Read File → Plain Bytes → Encode → Plain Text → Encrypt → Cipher Text → Decode → Cipher Bytes → Write File → Output File (IMG0012.jpg.pbe)

Decryption Procedure:
Input File (IMG0012.jpg.pbe) → Read File → Cipher Bytes → Encode → Cipher Text → Decrypt → Plain Text → Decode → Plain Bytes → Write File → Output File (IMG0012.jpg)

* (pbe = Protected By Encryption)

۳-۳. نحوه‌ی اجرای برنامه

ورودی‌های اجرای برنامه از طریق آرگومان‌های متد main فرستاده می‌شود. ((public static void main (String[ ] args) برای این منظور، کاربر با وارد نمودن یک دستور (Command) که چند از قسمت تشکیل شده‌است، می‌تواند عملیات مورد نظر خود را به اجرا برساند. این دستور دارای ساختار مشخصی است که در جدول زیر (جدول ۲) بخش‌های مختلف آن بررسی شده‌اند:

Switch	Argument Count	Argument	Presence	Description
-es	1	Integer	Mandatory	Specifies simple encryption operation and captures the key as an argument.
-ds	1	Integer	Mandatory	Specifies simple decryption operation and captures the key as an argument.
-ec	1	String	Mandatory	Specifies complex encryption operation and captures the key as an argument.
-dc	1	String	Mandatory	Specifies complex decryption operation and captures the key as an argument.
-i	1	String	Mandatory	Input file or directory path
-o	1	String	Optional	Output directory path
-r	0	none	Optional	It means that input files must be removed after completing the operation.

جدول ۲

در پیاده‌سازی تفسیر دستور ورودی به نکات زیر توجه نمایید:

- اقلام دستور (سوییچ‌ها و آرگومان‌ها) با space از هم تفکیک می‌گردند.
- در استفاده از سوییچ‌ها در یک دستور ترتیب وجود ندارد.
- چنانچه یک سوییچ آرگومان داشته باشد، آرگومان بلافاصله پس از سوییچ مربوطه قرار می‌گیرد.
- در دستور وارد شده حتما باید یکی از سوییچ‌های مربوط به فرآیند (es- ، ds- ، ec- و dc-) حضور داشته باشد.
- چنانچه آرگومان سوییچ i- آدرس مربوط به یک دایرکتوری باشد، عمل مشخص شده توسط سوییچ مربوط به فرآیند می‌بایست بر روی تمامی فایل‌های داخل این دایرکتوری به صورت جداگانه صورت گیرد.
- چنانچه در دستور از سوییچ o- استفاده شده باشد، فایل‌های خروجی می‌بایست در مسیری که توسط آرگومان این سوییچ مشخص شده است، ذخیره گردند.
- چنانچه در دستور از سوییچ o- استفاده نشده باشد، فایل‌های خروجی می‌بایست در دایرکتوری فعلی آن‌ها ذخیره گردند.

- چنانچه آدرس مشخص شده در دستور برای خروجی (آرگومان سوییچ o-) وجود نداشت، می‌بایست دایرکتوری مورد نظر در مسیر داده شده ایجاد گردد.

نمونه‌ای از دستور وارد شده توسط کاربر به صورت زیر خواهد بود:

-ec "mypassword" -r -i "C:\Users\Amin\Desktop\IMG0012.jpg"

با اجرای دستور فوق می‌بایست فایل IMG0012.jpg توسط روش جابجایی پیچیده با کلید mypassword رمزگذاری شده و فایل خروجی IMG0012.jpg.pbe در همان پوشه‌ی Desktop ذخیره گردد، همچنین فایل ورودی نیز باید حذف شود.

*** توجه:** لازم به ذکر است در تفسیر دستور وارد شده توسط کاربر، چنانچه هر گونه مشکلی در آن وجود داشت (مثلا پس از سوییچ i- آدرس فایل نامعتبر باشد، یا آرگومان برای این سوییچ ارسال نشده باشد و ...) می‌بایست عبارت خطای مناسب برای کاربر در Console چاپ گردد.

۳-۴. راهنمایی بهبود عملکرد (امتیازی)

به‌جای آن‌که کل یک فایل را به صورت یک byte[ ] بخوانید و آن‌را Decode و سپس رمزنگاری نمایید، می‌توانید این کار را به صورت تکه تکه انجام دهید. به عنوان مثال می‌توانید ۳ کیلوبایت از فایل را خوانده و عملیات مورد نظر را بر روی آن انجام دهید و در فایل مقصد بنویسید و سپس ۳ کیلوبایت بعدی و الی آخر. با این کار میزان مصرف حافظه مدیریت شده و کاهش خواهد یافت. همچنین می‌توان در پیاده‌سازی محاسبات، بهینگی را در نظر داشت تا سرعت اجرا تا حد امکان بیش‌تر گردد.

۴. جزییات تصحیح پروژه

عنوان	نمره
تفسیر دستور ورودی	۱۰
ENCODE و DECODE فایل	۲۰
رمزنگاری جابجایی ساده و پیچیده	۳۰
رعایت نکات پیاده‌سازی	۲۰
صحت عملکرد	۲۰
کارایی عملکرد (امتیاز اضافه)	۱۰
مجموع	۱۰۰ (+۱۰)