Весна 2021

Системное программное обеспечение

Онлайн-лекции

Лекция №12: Шифрование. UTF-8 . Звуковые файлы

Доцент, к.т.н. ГОЛЬЦОВ Александр Геннадьевич

Простейшие алгоритмы шифрования

- Замена
- Сложение с ключом
- Гаммирование
- Перестановка

- Часто шифрование/дешифрование является дополнительной ступенью доступа ко всем данным приложения, хранящимся в файлах, и требует эффективной реализации.

Замена символов - идея

- Для каждого символа в виде таблицы или аналитически (формулой) задается пара - заменяющий символ
- Соответствие "исходный-заменяющий" взаимно-однозначное (работает "в обе стороны")
- В ходе шифрования вместо исходных символов в выходную последовательность попадают заменяющие
- В ходе дешифрования замены проводятся в обратном направлении

Замена символов - обсуждение

- Это примитивный, легко вскрываемый шифр.
- Замену символа по таблице легко реализовать при помощи команды XLAT.
- Если символы входного сообщения имеют разную статистику встречаемости (например, это текст на естественном языке), то шифр вскрывается на основе частоты встречаемости символов.
- Шифр также вскрывается, если в распоряжение криптоаналитика попадет исходное и зашифрованное сообщение.
- Шифр может успешно применяться, если символы исходного сообщения встречаются с одинаковой частотой (например, шифруются данные, подвергнутые эффективному кодированию по Хаффману или Шеннону-Фано).
- Известный алгоритм шифрования DES в базовом варианте фактически тоже является шифром простой замены, только заменяются не символы, а блоки по 8 байт.

Замена символов - пример

- Пусть при шифровании берется символ с кодом на 3 большим, чем код исходного символа:

  'Иван Иваныч' ➔ 'Легр#Легрюъ'

- Очевидно, что при шифровании нужно прибавлять к каждому байту входной последовательности 3, а при дешифровании - вычитать 3.
- Если сумма вдруг превысит 255 - естественным образом в качестве результата сложения окажется младший байт суммы, что в этом случае и требуется

Сложение с ключом - идея

- Ключом шифрования является последовательность байтов ограниченной длины *K*, например, слово или фраза на естественном языке.
- Шифруемое сообщение - обычно более длинное, чем ключ.
- *K* первых кодов символов сообщения складываются с кодами ключевого слова, потом - следующие *K* и т.д., пока сообщение не закончится:

	м	а	м	а		м	ы	л	а		р	а	м	у
+	к	л	ю	ч	к	л	ю	ч	к	л	ю	ч	к	л
	6	11	B	G	K	7	5	L	10	Л	14	G	6	H

Операция XOR

Часто вместо сложения-вычитания используют операцию XOR, тогда шифрование и дешифрование не отличаются.

Шифрование: X xor Y ➔ Z

Дешифрование: Z xor Y ➔ X

X xor Y xor Y = X

Сложение с ключом - обсуждение

- Чем длиннее ключ, тем сложнее работать криптоаналитику.
- Если в руки криптоаналитика попадет исходное и зашифрованное сообщение - ключ очевиден.
- Хотелось бы сделать ключ бесконечно длинным - но как?

Гаммирование

- Под гаммированием понимают сложение с ключом большой длины (длина ключа обычно больше длины сообщения).
- Ключ не хранится / не вводится как последовательность байтов, а вычисляется в процессе шифрования.
- Один из способов вычисления гаммы - выходная последовательность классического генератора псевдослучайных чисел, например - мультипликативного датчика:

  $X_i = (A*X_{i-1} + B) \bmod M$
  key_i = младший байт X_i
  X_i - двухбайтовое или 4-байтовое

- Собственно ключом шифрования является X_0, с которого начинается генерация последовательности ПСЧ, или четыре числа (X_0, A, B, M).

Гаммирование - обсуждение

- Криптостойкость гаммы весьма высока по сравнению со сложностью алгоритма.
- Если криптоаналитику известен алгоритм генерации гаммы - шифр можно считать вскрытым.

- random(N) ➔ 0..N-1 (RandSeed) Randomize

Как вводить ключ гаммы?

- Пользователь может ввести для шифрования / дешифрования число-затравку. Это вполне приемлемо.
- Пользователь может ввести ключевое слово - любую последовательность любых символов. Тогда дополнительно по этому слову нужно сгенерировать затравку, например, циклически сдвигая X_0 и прибавляя очередной байт из введенного пользователем слова.
- При этом для одного и того же X_0 существует множество слов-ключей.

Преобразование ключевого слова в затравку

```
.286
.model small
.stack 100h

.data
KeyLen equ 7
Key db 'keyword'
X0  dw 0
.code
        mov     ax,@data
        mov     ds,ax

        mov     cx, keylen
        xor     bx,bx

m1:     mov     al,key[bx]
        add     byte ptr X0, al
        rol     X0,8
        inc     bx
        loop    m1

        mov     ax,4C00h
        int 21h
        end
```

Перестановка - идея

- Входная последовательность разбивается на блоки равной длины.
- Байты внутри каждого блока переставляются по определенному закону, задаваемому таблично или аналитически (формулой).
- Поскольку последний блок может оказаться не полным, есть варианты: вообще не переставлять байты последнего блока или дополнить сообщение до количества байт, кратного размеру блока, но тогда придется как-то хранить количество лишних байт для отбрасывания при расшифровке.

Перестановка - обсуждение

- Символы исходного сообщения сохраняются.
- Чем больше блок, тем сложнее криптоаналитику.
- Если исходное и зашифрованное сообщение попадут в руки криптоаналитика - шифр будет вскрыт.
- Как задавать ключ? Поручить конечному пользователю указать последовательность номеров в блоке кого куда переставлять - плохая идея.

Перестановка - пример

Зашифрование

```
.model small
.stack 100h

.data
BlSize = 8
Key db 2,4,3,6,7,1,0,5
msg db 'corridor'
res db BlSize dup(?)
.code
        mov     ax,@data
        mov     ds,ax

        xor     ax,ax
        xor     bx,bx
        mov     cx, BlSize

m1:     mov     al,key[bx]
        mov     di,ax
        mov     al,msg[bx]
        mov     res[di],al
        inc     bx
        loop    m1

        mov     ax,4C00h
        int 21h
        end
```

Расшифрование

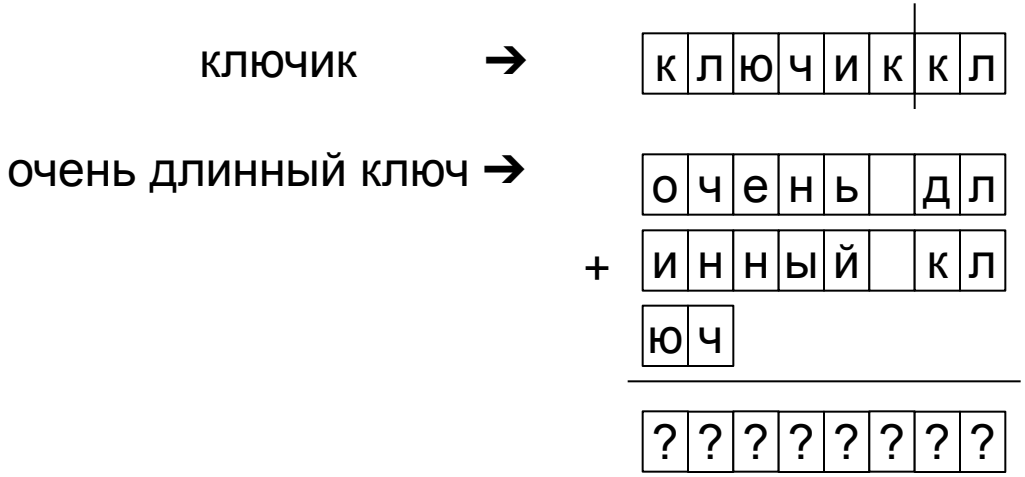
```
.model small
.stack 100h

.data
BlSize = 8
Key db 2,4,3,6,7,1,0,5
msg db 'odcrorri'
res db BlSize dup(?)
.code
        mov     ax,@data
        mov     ds,ax

        xor     ax,ax
        xor     bx,bx
        mov     cx, BlSize

m1:     mov     al,key[bx]
        mov     di,ax
        mov     al,msg[di]
        mov     res[bx],al
        inc     bx
        loop    m1

        mov     ax,4C00h
        int 21h
        end
```

Ключ для перестановки

- Переменная Key в примере - это массив чисел от 0 до BufSize-1, собственно определяющий перестановку байтов в блоке.
- Поручить пользователю формировать такой ключ неправильно - он может ошибиться.
- Идея - пользователь вводит слово (последовательность символов) произвольной длины, а программа на его основе формирует ключ перестановки. Как и с гаммой, для одной таблицы может быть множество ключевых слов.

Формирование промежуточного ключа

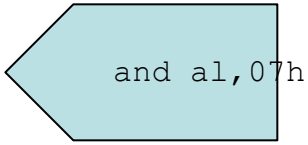
- Нужна ключевая последовательность длиной BufSize:
  - ее может ввести пользователь нужной длины
  - ее может сгенерировать датчик ПСЧ, проинициализированный как предыдущем примере с гаммой
  - нужную длину можно сформировать из строки пользователя любой длины, продолжая строку или складывая ее саму с собой:



Формирование ключа перестановки

```
.model small
.stack 100h

.data
BlSize = 8
MidKey db 'keywordk'
Key db BlSize dup (-1)

.code
        mov     ax,@data
        mov     ds,ax

        mov     dl, BlSize
        xor     bx,bx
        mov     di,bx
        mov     cx, BlSize
```

```
m1:     mov     al,MidKey[bx]
        xor     ah,ah
        div     dl
        xor     al,al
        xchg    al,ah
        mov     di,ax
Check:  test    Key[di],80h
        jnz     Empty
        inc     di
        cmp     di,BlSize
        jne     Check
        xor     di,di
        jmp     Check
Empty:  mov     Key[di],bl
        inc     bx
        loop    m1

        mov     ax,4C00h
        int 21h
        end
```



Итоговые замечания

- Считывать и записывать информацию из/в шифруемый файл необходимо блоками.
- Ключ желательно задавать в удобном конечному пользователю формате.
- Целесообразно делать программу с интерфейсом командной строки.
- Делать одну или разные программы для зашифрования и расшифрования - не очень важно.

Кодировка текстовых файлов

Кодировка текстовых файлов

- Набор ASCII - 128 стандартных символов (7 бит):
  - управляющие 0..31
  - знаки препинания и математических действий
  - цифры
  - большие и малые латинские буквы
- Однобайтовая кодировка: ASCII + 128 символов с кодами от 128 до 255. Коды в старших 128 позициях соответствуют буквам кириллицы или расширенной латиницы с черточками-точками-хвостиками.
- Unicode - 2 (или 4) байта на символ, этого достаточно для символов мировых языков, включая иероглифы, нот, математических, астрологических и прочих знаков.
- UTF (Unicode Transformation Format) - способ записи последовательности символов Unicode кодами переменной (от 1 до 4 байт) длины.

Кодовая страница 1251

0		16		32		48	0	64	@	80	P	96	`	112	p
1		17		33	!	49	1	65	A	81	Q	97	a	113	q
2		18		34	"	50	2	66	B	82	R	98	b	114	r
3		19		35	#	51	3	67	C	83	S	99	c	115	s
4		20		36	$	52	4	68	D	84	T	100	d	116	t
5		21		37	%	53	5	69	E	85	U	101	e	117	u
6		22		38	&	54	6	70	F	86	V	102	f	118	v
7		23		39	'	55	7	71	G	87	W	103	g	119	w
8		24		40	(	56	8	72	H	88	X	104	h	120	x
9		25		41	)	57	9	73	I	89	Y	105	i	121	y
10		26		42	*	58	:	74	J	90	Z	106	j	122	z
11		27		43	+	59	;	75	K	91	[	107	k	123	{
12		28		44	,	60	<	76	L	92	\	108	l	124	|
13		29		45	-	61	=	77	M	93	]	109	m	125	}
14		30		46	.	62	>	78	N	94	^	110	n	126	~
15		31		47	/	63	?	79	O	95	_	111	o	127	•

128	Ъ	144	ђ	160		176	°	192	А	208	Р	224	а	240	р
129	Ѓ	145	‘	161	Ў	177	±	193	Б	209	С	225	б	241	с
130	‚	146	’	162	ў	178	І	194	В	210	Т	226	в	242	т
131	ѓ	147	“	163	Ј	179	і	195	Г	211	У	227	г	243	у
132	„	148	”	164	¤	180	ґ	196	Д	212	Ф	228	д	244	ф
133	…	149	•	165	Ґ	181	µ	197	Е	213	Х	229	е	245	х
134	†	150	–	166	¦	182	¶	198	Ж	214	Ц	230	ж	246	ц
135	‡	151	—	167	§	183	·	199	З	215	Ч	231	з	247	ч
136	€	152	?	168	Ё	184	ё	200	И	216	Ш	232	и	248	ш
137	‰	153	™	169	©	185	№	201	Й	217	Щ	233	й	249	щ
138	Љ	154	љ	170	Є	186	є	202	К	218	Ъ	234	к	250	ъ
139	‹	155	›	171	«	187	»	203	Л	219	Ы	235	л	251	ы
140	Њ	156	њ	172	¬	188	ј	204	М	220	Ь	236	м	252	ь
141	Ќ	157	ќ	173		189	S	205	Н	221	Э	237	н	253	э
142	Ћ	158	ћ	174	®	190	s	206	О	222	Ю	238	о	254	ю
143	Џ	159	џ	175	Ї	191	ї	207	П	223	Я	239	п	255	я

Русская кодировка DOS (CP-866)

А 128	Б 129	В 130	Г 131	Д 132	Е 133	Ж 134	З 135	И 136	Й 137	К 138	Л 139	М 140	Н 141	О 142	П 143
Р 144	С 145	Т 146	У 147	Ф 148	Х 149	Ц 150	Ч 151	Ш 152	Щ 153	Ъ 154	Ы 155	Ь 156	Э 157	Ю 158	Я 159
а 160	б 161	в 162	г 163	д 164	е 165	ж 166	з 167	и 168	й 169	к 170	л 171	м 172	н 173	о 174	п 175
▒ 176	▒ 177	▓ 178	│ 179	┤ 180	╡ 181	╢ 182	╖ 183	╕ 184	╣ 185	║ 186	╗ 187	╝ 188	╜ 189	╛ 190	┐ 191
└ 192	┴ 193	┬ 194	├ 195	─ 196	┼ 197	╞ 198	╟ 199	╚ 200	╔ 201	╩ 202	╦ 203	╠ 204	═ 205	╬ 206	╧ 207
╨ 208	╤ 209	╥ 210	╙ 211	╘ 212	╒ 213	╓ 214	╫ 215	╪ 216	┘ 217	┌ 218	▉ 219	▄ 220	▌ 221	▐ 222	▀ 223
р 224	с 225	т 226	у 227	ф 228	х 229	ц 230	ч 231	ш 232	щ 233	ъ 234	ы 235	ь 236	э 237	ю 238	я 239
Ё 240	ё 241	Є 242	є 243	Ї 244	ї 245	Ў 246	ў 247	° 248	∙ 249	· 250	√ 251	№ 252	¤ 253	■ 254	nbsp 255

Коды между **п** и **р** разорваны псевдографикой.

"Текстовый фильтр-транслитератор"

- Символы табуляции 9 заменять на количество пробелов, помещающих следующий символ в позицию от начала строки, кратную 8.
- Символы псевдографики заменять на подходящие по начертанию обычные: - = + |
- Символы конца строки (13, 10) и конца страницы 12 - наверное, пропускать.
- Остальные управляющие символы с кодами меньше 32 - не пропускать.

Кириллица в Unicode

Кодировка Windows-1251															
Ђ 402	Ѓ 403	‚ 201A	ѓ 453	„ 201E	… 2026	† 2020	‡ 2021	€ 20AC	‰ 2030	Љ 409	‹ 2039	Њ 40A	Ќ 40C	Ћ 40B	Џ 40F
ђ 452	‘ 2018	’ 2019	“ 201C	” 201D	• 2022	– 2013	— 2014		™ 2122	љ 459	› 203A	њ 45A	ќ 45C	ћ 45B	џ 45F
NBSP A0	Ў 40E	ў 45E	Ј 408	¤ A4	Ґ 490	¦ A6	§ A7	Ё 401	© A9	Є 404	« AB	¬ AC	SHY AD	® AE	Ї 407
° B0	± B1	І 406	і 456	ґ 491	µ B5	¶ B6	· B7	ё 451	№ 2116	є 454	» BB	ј 458	Ѕ 405	ѕ 455	ї 457
А 410	Б 411	В 412	Г 413	Д 414	Е 415	Ж 416	З 417	И 418	Й 419	К 41A	Л 41B	М 41C	Н 41D	О 41E	П 41F
Р 420	С 421	Т 422	У 423	Ф 424	Х 425	Ц 426	Ч 427	Ш 428	Щ 429	Ъ 42A	Ы 42B	Ь 42C	Э 42D	Ю 42E	Я 42F
а 430	б 431	в 432	г 433	д 434	е 435	ж 436	з 437	и 438	й 439	к 43A	л 43B	м 43C	н 43D	о 43E	п 43F
р 440	с 441	т 442	у 443	ф 444	х 445	ц 446	ч 447	ш 448	щ 449	ъ 44A	ы 44B	ь 44C	э 44D	ю 44E	я 44F

'A' = u-0410, 'я' = u-044F

UTF-8

- Если файл состоит только из символов ASCII, то файл UTF идентичен файлу в однобайтовой кодировке.
- Для каждого символа его 16-битный код Unicode кодируется 1, 2, 3 или 4 байтами, чем меньше код Unicode, тем меньше байтов требуется:
  - 7 значащих битов ➔ 1 байт
  - 11 значащих битов ➔ 2 байта
  - 16 значащих битов ➔ 3 байта
  (- 21 значащий бит ➔ 4 байта)
- Файл (или поток передаваемых символов в кодировке UTF-8 в линии связи) принято начинать маркером UTF-8 - кодами EF BB BF

Алгоритм кодирования UTF-8

- Единственный байт: 0xxxxxxx
- 2 байта: 110xxxxx 10xxxxxx
- 3 байта: 1110xxxx 10xxxxxx 10xxxxxx

- Например:
  код '$'=24h = 100100 ➔ 00100100
  код '«'=ABh = 10101011 ➔ 11000010 10101011
  код 'Б'=411h = 100 00010001 ➔ 11010000 10010001

- Дамп файла в UTF-8, содержащего три символа ББББ и конец строки:
  EF BB BF D0 91 D0 91 D0 91 0D 0A
- То же в Unicode:
  FF FE 11 04 11 04 11 04 0D 00 0A 00

Ошибки в UTF-8

- Последовательность кодов может быть нарушена. Преобразователь из UTF-8 в другую кодировку должен быть устойчив к таким событиям.
- Если встретился "не первый" байт без первого - он игнорируется.
- Если встретился "первый" байт там, где ожидался "продолжающий" - начинается новая цепочка, искаженная игнорируется.
- Декодирование должно продолжаться, несмотря на ошибки.

Файлы WAV

Файлы WAV

- Это формат-контейнер, в файле могут быть как не сжатые данные (с такими файлами нужно работать в РЗ), так и, например, звук, сжатый при помощи MPEG1 layer 3 (MP3)
- Файл имеет заголовок (формат есть в архиве к РЗ) и собственно данные.
- Несжатые данные могут быть организованы в несколько дорожек (моно, стерео, квадро, 5.1 и т.п.)
- Частота дискретизации 8000, 11025, 22050, 44100, 48000 Гц
- 1, 2, 3 и 4 байта на отсчет.

Отсчеты в несжатом Wav-файле

- Каждый период частоты дискретизации при создании аудиофайла АЦП измеряет уровень сигнала. В каждой аудиодорожке (от каждого микрофона).
- Уровень сигнала кодируется по-разному в зависимости от параметра "бит на отсчет".
- 8-битные данные: от 0 до 255 (без знака!), среднее значение (полная тишина) соответствует 128.
- 16-битные данные: от -32768 до 32767, среднее значение соответствует 0.
- 24- и 32-битные данные - формат с плавающей запятой, кодируется число от -1.0 до 1.0.

"Склейка" WAV

- Формат (частота дискретизации, количество дорожек, бит на отсчет) должны совпадать.
- Если они не совпадают - можно отказаться выполнять склейку или нужно преобразовывать файлы к единому формату.
- Проще всего преобразовать моно➔стерео дублированием дорожек или понизить частоту дискретизации в 2 раза выбрасыванием отсчетов через один.
- В обратную сторону чуть сложнее - нужно усреднять стерео-отсчет в моно или пытаться интерполировать отсчеты с удвоенной частотой.

Склейка Wav

- В итоге у нового файла должен быть заголовок, в котором прописана правильная длина файла.
- Аудиоданные файлов должны быть скопированы последовательно, при желании - преобразование.
- Доступ к файлам нужно осуществлять крупными блоками.

Вывод информации о Wav

- Почти вся информация берется из заголовка, НО -
- Максимум громкости нужно посчитать, пробежавшись по всему массиву аудиоднанных.

Спасибо за внимание.