kaspersky.academy

Лекция 5.

Шифр Вернама

Онлайн-курс по математике в информационной безопасности

kaspersky.academy

Лекция 5. Шифр Вернама

Всем большой привет!

Вы еще не устали от шифров, которые можно взломать? Что ж, у меня для вас отличные новости: прямо сейчас мы бодрым шагом двинемся к единственному и неповторимому **шифру Вернама,** который вообще невозможно взломать!

Наш план на эту лекцию:

Поехали!

О шифре Вернама

Как вы поняли, шифр Вернама называется именно так, потому что изобрел его **Гильберт Вернам.** Вернам был телеграфистом, и криптография его волновала потому, что он не хотел отправлять открытые данные **по телеграфному каналу связи.** Сигналы телеграфа – это точки и тире, но мы будем считать их **нулями и единицами**, потому что бинарная логика – она и в телеграфах бинарная. А нам с нашими компьютерами удобнее вообще все буквы и цифры перевести в бинарный код, чтобы не заморачиваться.

Рисунок 1. Гильберт Вернам, автор шифра Вернама

Вернаму не хотелось сначала записывать куда-то данные, потом шифровать, потом отправлять... И он думал: вот бы мне получить такой алгоритм, который шифровал бы данные на лету! И придумал.

Для каждого сообщения Вернам брал такую же по длине последовательность нулей и единиц – **гамму**, каждый ее бит складывал с соответствующим битом сообщения и отправлял адресату.

Гамма – это ключ в шифре Вернама. Но правильно говорить – **ключевая последовательность,** потому что гамма длинная (равна длине сообщения). Гамма может быть такой:

0010101110101010101110010111111001110100001010100000110...

А шифр Вернама еще называют **шифром гаммирования**. Или **шифром одноразового блокнота.** Одноразового – потому, что одна гамма, которую Вернам записал себе в блокнот, может применяться к какому-либо сообщению только один раз. После этого Вернам ее вычеркивал из блокнота.

Шифр гаммирования – поточный симметричный шифр.

Ключ для этого шифра – это гамма, или ключевая последовательность.

Важно помнить, что **гамма должна быть случайной последовательностью** чисел алфавита, в котором закодирован текст. Если мы находимся в бинарном алфавите, то и символы гаммы – это случайная последовательность нулей и единиц.

Зашифрование и расшифрование шифра Вернама

Еще одно желание Вернама – получить **легкий способ шифрования.** Алфавит Вернама – бинарный, $A = \{0,1\}$. Мощность этого алфавита равна двум, $|A| = 2$. И Вернам решил, что **зашифровывать** текст он будет, складывая символы этого текста с символами гаммы по модулю мощности алфавита – по модулю два.

Да и **расшифровывать** легко – просто сложим шифртекст с гаммой еще раз – и вот уже перед нами читаемый и понятный открытый текст! Вообще, логично было бы теперь вычесть символы гаммы из символов шифртектса, но магия бинарной логики состоит в том, что в **бинарном алфавите сложение и вычитание – это одно и то же.**

Попробуем на примере. Ниже представлен открытый текст и ключевая последовательность (гамма). **Чтобы зашифровать текст,** давайте применим операцию побитового исключающего ИЛИ (XOR) для каждого символа с соответствующим символом гаммы. Шифруем:

открытый текст	1	0	0	1	1	1	1	0	1	0	1	1	1	1	0	1	0	1	1	0	0	1	1	0	0
гамма	1	1	0	0	1	0	1	0	1	0	0	1	1	0	1	0	1	1	0	1	1	1	0	1	1
шифртекст	0	1	0	1	0	1	0	0	0	0	1	0	0	1	1	1	1	0	1	1	1	0	1	1	1

Таблица 1. Зашифровываем шифром Вернама

На этом шаге вы поняли, что шифр гаммирования – **это поточный шифр?** Вернам зря времени не теряет – одновременно и складывает, и отправляет! Да и ключ расшифрования будет тот же – очень удобно. А значит, **шифр еще и симметричный.**

Теперь давайте расшифровывать. Сложим шифртекст с гаммой еще раз.

шифртекст	0	1	0	1	0	1	0	0	0	0	1	0	0	1	1	1	1	0	1	1	1	0	1	1	1
гамма	1	1	0	0	1	0	1	0	1	0	0	1	1	0	1	0	1	1	0	1	1	1	0	1	1
открытый текст	1	0	0	1	1	1	1	0	1	0	1	1	1	1	0	1	0	1	1	0	0	1	1	0	0

Таблица 2. Расшифровываем шифром Вернама

Как вы видите, буквально на лету мы с вами и Вернамом получили открытый текст!

Какой должна быть гамма?

Нет ничего сложнее, чем угадывать что-то случайное. Вот и гамму для шифрования мы тоже возьмем случайную.

Итак, гамма должна быть:

- случайной последовательностью
- равномерно распределенной последовательностью
- длина ее должна совпадать с длиной открытого текста.

Равномерно распределенная последовательность – это такая последовательность, у которой каждый ее символ появляется равновероятно. И если у нас есть монетка, которую мы подбрасываем, мы должны верить, что и орел, и решка выпадут с вероятностью.

Когда у математиков еще не было генераторов псевдослучайных последовательностей, существовала профессия **операциониста**. Девушки-операционистки сидели в кабинетах, подбрасывали монетку и записывали результат. И еще. И еще. И так целый рабочий день!

Так и вырабатывались гаммы для шифрования. А после этого все одноразовые блокноты с гаммами складывались в чемоданчик, и курьер, которому все очень доверяли, нес эти ключи на другой конец света – адресату, чтобы получатель смог расшифровать сообщения.

Сложно? Еще как!

А вот текст, написанный на естественном языке, – на русском, например, – гаммой быть никак не мог. Почему? Да потому что в таких текстах распределение символов не случайно. Вспомните, как мы ломали шифр подстановки – строили график для частоты встречаемости букв. Какая уж тут случайность, когда какие-то буквы выпадают чаще, а какие-то – реже.

Однобайтный XOR

Давайте рассмотрим еще одну историю, связанную с шифром гаммирования.

Теперь мы ненадолго представим себя вирусными аналитиками – это ребята, которые читают программный код разных вредоносных приложений. Все такие приложения тщательно маскируют свои следы, чтобы вирусные аналитики не догадались, какие папки они открывают и какие файлы создают. А для этого вирусописатели шифруют имена папок.

Но чтобы долго не заморачиваться, они берут коротенькую гамму и постоянно повторяют ее – стойким такой шифр не будет, но все-таки текст превратит в нечитаемый. Конечно, ребята-аналитики все равно расшифровывают всю их крипту, и даже очень сложную :) Но давайте на минутку представим себе, что происходит на стороне зла.

Как все вы прекрасно помните, любой символ в компьютере представляется в двоичном коде. Мы обратимся к кодировочной таблице ASCII.

Каждая буква кодируется одним байтом, или 8 битами. Например, буква A имеет шестнадцатеричный код 0x41 (сначала смотрим цифру в строке, затем в столбце). Однобайтная гамма в данном случае будет равна закодированному значению буквы.

Так как процесс получения шифртекста – это применение побитового исключающего ИЛИ, или XOR, к открытому тексту и ключевой последовательности, а длина кода одной буквы равна 1 байту, то такую гамму называют **однобайтным ксором.**

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	SOH	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Маленькая ремарка.

«Однобайтный ксор» – это суперсленговое название, которое используют программисты, вирусные аналитики, безопасники и другие ребята, работающие в IT. Конечно, можно сказать «применение побитового исключающего ИЛИ», но «ксорить» звучит короче и загадочнее :)

И снова пример! В качестве открытого текста пока возьмем только одну букву A. Ее однобайтный ксор – буква w (по таблице – 0x77). Однобайтный ксор применяется ровно к одной букве. Давайте переведем обе буквы в двоичный вид и поксорим!

Вот как это будет работать с нашими данными:

0x77 = 01110111 – двоичный вид w

0x41 = 01000001 – двоичный вид A

0x77	0	1	1	1	0	1	1	1
0x41	0	1	0	0	0	0	0	1
0x77 ⊕ 0x41	0	0	1	1	0	1	1	0

Возьмем получившийся шифртекст:

0011 0110 = 0x36 – этому коду соответствует цифра 6

То есть:

«A» ⊕ «w» = 0x41 ⊕ 0x77 = 0x36 = «6»

Символ буквы или цифры мы пишем в кавычках, а ее закодированное значение – в виде шестнадцатеричного числа.

Если же мы хотим **зашифровать целую строку,** то буква-гамма применяется к каждому символу отдельно. Просто повторите ее столько раз, сколько нужно для зашифрования всего текста.

Например, если бы мы наложили букву w на строку AAAA, мы получили бы значение 6666:

AAAA ⊕ 0x77 = 6666

Что же делать с русским алфавитом?

Если вы уже успели огорчиться, что мы переезжаем жить в мир нулей и единиц, то не расстраивайтесь – русские буквы тоже будут! Давайте для начала их пронумеруем:

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я

Здесь 32 буквы – от «ё» мы в этот раз откажемся. А значит, когда мы будем складывать код буквы с кодом гаммы, мы будем брать модуль 32 – это **мощность русского алфавита.**

Ну что, давайте шифровать вот такую фразу и гамму к ней:

Ч	Е	М	Ш	И	Ф	Р	У	Ю	Т	Р	У	С	С	К	И	Й	А	Л	Ф	А	В	И	Т
Т	О	Ч	Н	О	Н	Е	О	Д	Н	О	Б	А	Й	Т	Н	Ы	М	К	С	О	Р	О	М

Давайте вместе **посчитаем код для первой буквы шифртекста**, а дальше вы сами. Код обозначим переменной *c* от слова code:

$$c(Ч) + c(Т) = (23 + 18) \bmod 32 = 9.$$

Код 9 – это буква Й. Пишем!

Ч	Е	М	Ш	И	Ф	Р	У	Ю	Т	Р	У	С	С	К	И	Й	А	Л	Ф	А	В	И	Т
Т	О	Ч	Н	О	Н	Е	О	Д	Н	О	Б	А	Й	Т	Н	Ы	М	К	С	О	Р	О	М
Й	У	Г	Е	Ц	Б	Х	Б	В	Я	Ю	Ф	С	Ъ	Ь	Х	Д	М	Х	Е	О	Т	Ц	Ю

А что насчет **расшифровки**? Здесь нам не так везет, как в двоичном алфавите, в котором сложение и вычитание – это одна и та же операция. Так что будем теперь вычитать по-честному. И про модуль 32 не забывайте!

Й	У	Г	Е	Ц	Б	Х	Б	В	Я	Ю	Ф	С	Ъ	Ь	Х	Д	М	Х	Е	О	Т	Ц	Ю
Т	О	Ч	Н	О	Н	Е	О	Д	Н	О	Б	А	Й	Т	Н	Ы	М	К	С	О	Р	О	М

Подсказка для первой буквы:

$$c(Й) - c(Т) = (9 - 18) \bmod 32 = -9 \bmod 32 = 32 - 9 = 23$$

Получаем букву Ч. Общий результат:

Й	У	Г	Е	Ц	Б	Х	Б	В	Я	Ю	Ф	С	Ъ	Ь	Х	Д	М	Х	Е	О	Т	Ц	Ю
Т	О	Ч	Н	О	Н	Е	О	Д	Н	О	Б	А	Й	Т	Н	Ы	М	К	С	О	Р	О	М
Ч	Е	М	Ш	И	Ф	Р	У	Ю	Т	Р	У	С	С	К	И	Й	А	Л	Ф	А	В	И	Т

Абсолютно стойкие и достаточно стойкие шифры

Пришло время рассмотреть абсолютно стойкий шифр.

Абсолютно стойкий шифр – это такой шифр, который нельзя взломать ни теоретически, ни практически, даже если у злоумышленника есть бесконечно большие вычислительные ресурсы.

Он может применять брутфорс-атаку, но даже если он переберет все варианты ключа, он все равно не поймет, взломал он сообщение или нет.

Представьте себе, что перед вами сундук и тысяча ключей. И вы пытаетесь каждым ключом открыть сундук. На подходе уже последний, тысячный ключ – вы применяете его и… Ничего. Совсем. Вообще. Ключей больше не осталось, а сундук все еще закрыт.

Давайте смотреть примеры. Пускай по каналу связи передаются координаты одного очень важного объекта.

$$(55^{\circ}50^{0}11.6;37^{\circ}28^{0}46.1)$$

Разделим точкой градусы, минуты и секунды координат:

$$55.50.11.6;\ 37.28.46.1$$

И запишем их в бинарном виде:

00110111.00110010.00001011.00000110; 00100101.00011100.00101110.00000001

Что перехватывает злоумышленник?

Он получает последовательность нулей и единиц. Он может подставлять любые варианты 0 и 1 на каждую из позиций и перебирать все возможные варианты, но это никак не приблизит его к разгадке.

Возможно, первое число координат – это 55. Может, 67, 11, 43, 65... **Вариантов бесконечно много, и даже перебрав все из них, злоумышленник не получит ответ.**

Почему? Да у него просто не будет критерия, по которому он сможет остановиться и решить, что нашел правильный ответ. Все ключи перебрал, а сундук все еще закрыт!

Совсем другое дело с **достаточно стойкими шифрами.**

Достаточно стойки шифр – это шифр, который можно взломать только брутфорс-атакой или полным перебором.

Они уже не являются теоретически невзламываемыми, но брутфорсить или перебирать все возможные ключи долго и тяжело. А никаких других вариантов, которые работали бы быстрее, мы не знаем.

С достаточно стойкими шифрами мы с вами еще не сталкивались, и в первый раз их увидим в шифре **RSA.**

А все шифры, которые мы рассматривали до шифра гаммирования, – **вообще не стойкие,** потому что мы умеем их ломать. Вспомните шифр Цезаря. Что там будет являться ключом? Шаг сдвига! И когда мы переберем все возможные шаги сдвига, какой-то точно нам даст правильный ответ: после какого-то шага текст станет осмысленным.

Если длина бинарного сообщения также равна N, то количество переборных вариантов будет 2^N.

Каждые 8 бит сообщения – это код некоторой буквы. Таким образом, мы будем перебирать ключевые последовательности до тех пор, пока не получим искомую последовательность.

А вот **шифр Вернама является абсолютно стойким,** потому что единственное, что мы про него знаем, – это длина шифртекста. А сколько существует фраз одинаковой длины? Навскидку можно привести слова одинаковой длины:

СЛОН
СЛОТ
ДИСК
ПИСК
ДАМП
...

Какое слово верное? Непонятно.

Криптоанализ шифра Вернама: как ломать невзламываемые шифры?

То, что шифр Вернама абсолютно стойкий, – чистая правда. Но не зря же мы так долго говорили про гамму и про то, какой она должна быть. **Любая гамма не подойдет.** Нужна случайная.

А еще мы говорили, **что одну гамму нельзя использовать дважды**, иначе вся стойкость шифра рассыплется.

Что ж, представим, что кто-то поленился и все-таки использовал одну гамму два раза. Назовем гамму буквой K – ключ. И зашифруем два разных сообщения – A и B – этой гаммой. Получим два шифртекста C_1 и C_2,

$$A \oplus K = C_1$$

$$B \oplus K = C_2$$

А если теперь сложить по модулю получившиеся шифртексты, то значение гаммы уничтожится: K складывается сама с собой, и каждый его разряд будет равен 0: $0 \oplus 0 = 0, 1 \oplus 1 = 0$. Не верите? Проверим:

A	0	1	1	1
K	0	1	1	0
B	1	1	0	1
K	0	1	1	0
$A \oplus K \oplus B \oplus K = A \oplus B$	1	0	1	0

Теперь, когда мы получили выражение $A \oplus B$, можно применить атаку брутфорса. Мы будем подбирать A перебором. Если угадаем правильно, настоящее A и наше подобранное A^0 взаимоуничтожатся и мы увидим открытое сообщение B. Критерий того, что догадка верная – **получение осмысленного сообщения B.**

Достоинства и недостатки шифра Вернама

Давайте теперь кратко подытожим все, что мы знаем. **Чем же шифр Вернама хорош?**

- сам процесс шифрования довольно легкий – нужно просто уметь складывать по модулю числа.
- если длина гаммы равна длине текста, шифр является невзламываемым.

И это действительно делает шифр Вернама очень крутым! Но не так все просто, и сложности тоже есть.

Во-первых, как сгенерировать случайную последовательность большой длины? Можно, конечно, нанять отдел с людьми, побрасывающими монетки, или же отдел математиков и программистов, которые будут разрабатывать генераторы псевдослучайных последовательностей. Но задача эта сложная.

Во-вторых, как передавать ключевую последовательность? У нас же нет защищенного канала связи. А если бы был, то и шифровать сообщения незачем.

Наконец, что делать с ошибками при передаче сообщения? Что если при передаче сотрется какой-то бит числа? И вместо слова «Криптография» мы получим слово «Табуретка»?

Итак, используйте шифр Вернама и помните: здесь все дело в гамме!