The Vernam Cipher

Carol Pignato & Hareem Bokhari

History

The Vernam Cipher is named after Gilbert Sandford Vernam (1890-1960) who, in 1917, invented the stream cipher and later co-invented the One Time Pad. His patent US 1,310,719 was filed in 1918 and is, according to the NSA, perhaps the most important one in the history of cryptography[1] .

The Vernam Cipher uses the One-Time Pad, an encryption technique in which each character of the plaintext is combined with a character from a random key stream. When used correctly, the OTP provides a truly unbreakable cipher. It is named after the sheets of paper (pads) on which the key stream was usually printed, as it consists of a stack of small very thin pages, each with a series of random numbers on them. Each page was destroyed immediately after use. It was used by agents of the former Soviet Union (USSR) during the 1960s[2].

However, the Vernam Cipher uses ASCII binary values and the XOR logic gate, rather than the usual alphabet numbering.

1. https://www.cryptomuseum.com/crypto/vernam.htm
2. https://www.cryptomuseum.com/crypto/otp/index.htm

About

- The Vernam Cipher is known to have perfect security and is unbreakable.
- It depends on the fact that the key must be truly random.
- The Vernam Cipher is based on the idea that each plaintext character from a message is mixed with one character from a key stream.
- It is a symmetric cipher - The key used to encrypt a plaintext message must be the same key used to decrypt that message.

XOR Logic Gate

The XOR operator outputs a 1 whenever the <u>inputs do not</u> <u>match</u>, which occurs when one of the two inputs is **exclusively true**.

Here is the truth table:

Input		Output
A	B	A XOR B
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**0**

Procedure

Encrypting

1. Get your plaintext and its ASCII values and convert it to binary
2. Generate a key that is truly random and is as long as the plaintext. Convert that to binary too.
3. Produce the ciphertext by applying bitwise XOR on the plaintext and the key

Decrypting

1. Take your cipher text (in binary).
2. Use the secret shared key given to you by the other person (in binary).
3. Reproduce the plaintext by applying the bitwise XOR on the ciphertext and key.

Example using the ASCII table
plain text: math
random key: DEAD

remember key must be same or greater length as plain text

Binary Plain Text m	Binary key D	Bitwise XOR
1	1	0
1	0	1
0	0	0
1	0	1
1	1	0
0	0	0
1	0	1

Dec	Hex	Oct	Binary	Char	Dec	Hex	Oct	Binary	Char
64	40	100	1000000	@	96	60	140	1100000	`
65	41	101	1000001	A	97	61	141	1100001	a
66	42	102	1000010	B	98	62	142	1100010	b
67	43	103	1000011	C	99	63	143	1100011	c
68	44	104	1000100	D	100	64	144	1100100	d
69	45	105	1000101	E	101	65	145	1100101	e
70	46	106	1000110	F	102	66	146	1100110	f
71	47	107	1000111	G	103	67	147	1100111	g
72	48	110	1001000	H	104	68	150	1101000	h
73	49	111	1001001	I	105	69	151	1101001	i
74	4A	112	1001010	J	106	6A	152	1101010	j
75	4B	113	1001011	K	107	6B	153	1101011	k
76	4C	114	1001100	L	108	6C	154	1101100	l
77	4D	115	1001101	M	109	6D	155	1101101	m

The bitwise XOR: 0101001 is the right parenthesis

)

Binary Plain Text a	Binary key E	Bitwise XOR
1	1	0
1	0	1
0	0	0
0	0	0
0	1	1
0	0	0
1	1	0

Dec	Hex	Oct	Binary	Char	Dec	Hex	Oct	Binary	Char
64	40	100	1000000	@	96	60	140	1100000	`
65	41	101	1000001	A	97	61	141	1100001	a
66	42	102	1000010	B	98	62	142	1100010	b
67	43	103	1000011	C	99	63	143	1100011	c
68	44	104	1000100	D	100	64	144	1100100	d
69	45	105	1000101	E	101	65	145	1100101	e

The bitwise XOR: 0100101 is the dollar sign

$

Binary t	Binary A	Bitwise XOR
1	1	0
1	0	1
1	0	1
0	0	0
1	0	1
0	0	0
0	1	1

The bitwise XOR: 0110101 is the digit

5

Binary	Char	Dec	Hex	Oct	Binary	Char
1000000	@	96	60	140	1100000	`
1000001	A	97	61	141	1100001	a
1000010	B	98	62	142	1100010	b
1000011	C	99	63	143	1100011	c
1000100	D	100	64	144	1100100	d
1000101	E	101	65	145	1100101	e
1000110	F	102	66	146	1100110	f
1000111	G	103	67	147	1100111	g
1001000	H	104	68	150	1101000	h
1001001	I	105	69	151	1101001	i
1001010	J	106	6A	152	1101010	j
1001011	K	107	6B	153	1101011	k
1001100	L	108	6C	154	1101100	l
1001101	M	109	6D	155	1101101	m
1001110	N	110	6E	156	1101110	n
1001111	O	111	6F	157	1101111	o
1010000	P	112	70	160	1110000	p
1010001	Q	113	71	161	1110001	q
1010010	R	114	72	162	1110010	r
1010011	S	115	73	163	1110011	s
1010100	T	116	74	164	1110100	t

Binary h	Binary D	Bitwise XOR
1	1	0
1	0	1
0	0	0
1	0	1
0	1	1
0	0	0
0	0	0

The bitwise XOR: 0101100 is a comma

,

THE ENCODED TEXT IS

)$5,

plain text: math

random key: DEAD

Cipher text:)$5,

Now we could decode the cipher using our same key

Binary Cipher Text)	Binary key D	Bitwise XOR
0	1	1
1	0	1
0	0	0
1	0	1
0	1	1
0	0	0
1	0	1

0100100	$	68	44	104	1000100	D
0100101	%	69	45	105	1000101	E
0100110	&	70	46	106	1000110	F
0100111	'	71	47	107	1000111	G
0101000	(	72	48	110	1001000	H
0101001	)	73	49	111	1001001	I

The bitwise XOR: 1101101 is the lowercase letter

m

Binary $	Binary E	Bitwise XOR
0	1	1
1	0	1
0	0	0
0	0	0
1	1	0
0	0	0
0	1	1

The bitwise XOR: 1100001 is the lowercase letter a

Binary 5	Binary A	Bitwise XOR
0	1	1
1	0	1
1	0	1
0	0	0
1	0	1
0	0	0
1	1	0

The bitwise XOR: 1110100 is the lowercase letter t

Binary ,	Binary D	Bitwise XOR
0	1	1
1	0	1
0	0	0
1	0	1
1	1	0
0	0	0
0	0	0

The bitwise XOR: 1101000 is the lowercase letter h

)$5, is now decrypted to math

Program

Let's go through the code together

https://repl.it/repls/WobblyMonstrousNetwork

Conclusion

Pros

- Because it is truly random, there is no way to break the cipher and frequency analysis cannot be used

Cons

- Key must be truly random, without any detection of a pattern
- Key length must be greater than or equal to the length of the message
- The key must be destroyed immediately after use, and you thus cannot reuse the same key.
- Difficulty with mass distribution as it could be easier to gain access to the key