XOR and more

Notes on August 20 reading

Jeffrey Goldberg
jeffrey@goldmark.org

August 20, 2021 (Revised May 8, 2024)

AgileBits, Inc

Adding stuff

Adding stuff

Adding letters and numbers

Adding 3

- 'A' $+ 3 =$ 'D'
- 'N' $+ 3 =$ 'Q'
- 'Y' $+ 3 =$ 'B'

Subtracting 3

- 'N' $-$ 3 $=$ 'K'
- 'Y' $-$ 3 $=$ 'V'
- 'A' $-$ 3 $=$ 'X'

Subtracting by adding

Subtracting 3 is the same as adding $26 - 3$.

- 'N' $+ 23 =$ 'K'
- 'Y' $+ 23 =$ 'V'
- 'A' $+ 23 =$ 'X'

Nqqvat (be ebgngvat ol) 13 vf gur fnzr nf fhogenpgvat 13.

Adding stuff

Adding letters

Adding ‘J’

- 'A' + 'J' = 'J'
- 'N' + 'J' = 'W'
- 'Y' + 'J' = 'H'

From letters to numbers

$$\text{'A'} = 0$$
$$\text{'B'} = 1$$
$$\dots$$
$$\text{'Y'} = 24$$
$$\text{'Z'} = 25$$

$$c = p + k \pmod{26}$$

Don't leave spaces in Vigenère

Gen. E. Kirby Smith, comdg. Trans-Miss. Dept., Gen.: - Vvq ecilmympm rvcog ui lhomnides kfch kdf wasptf us tfcfsto abxc bjx azjkhmgjsiimivbceq qb ndel ueisu ht kfg auhd egh opcm mfs uvajwh xrymcoci yu dddxtmpt iu icjqkpxt es vvjau mvrr twhtc abxc iu eoieg o rdcgx en ucr pv ntiptyxec rqvariyyb rgzq rspz rksjcph ptax rsp ekez raecdstrzpt mzmseb acgg nsfqvvf mc kfg smhe ftrf wh mvv kkge pyh fefm ckfrlisytyxl xj jtbbx rq htxd wbhz awvv fd acgg avxwzvv yciag oe nzy fet lgxa scuh.

I am most respectfully your obdt. servt.,
(Signed) R.E. Lee.

And don't partially encrypt

Montgomery, 30th.

To Gen'l E. K. Smith,

Shreveport, La., via Wi.

What are you doing to execute the instructions sent you, to HCDLVW XMWQIG KM GOEI DMWl JN VAS DGUGUHDMITD. If success will be more certain, you can substitute EJTFKMPG OPGEEVT KQFARKF TAG HEEPZZN BBWYPHDN OMOMNQQG. By which you may effect O TPQGEXYK above that part HJ OPG KWMCT patrolled by the ZMGRIK GGIUL CW EWBNDLXL.

Jeffn. Davis

Hints

- Both messages use the same key
- The two instances of ‘kfg’ in the first are encodings of the same word.
- The distance between them will be a multiple of the length of the key
- What 11 letter word might be mentioned in an order to a general at Shreveport?
- It shouldn’t be hard to guess the encrypted part of “above that part HJ OPG KWMCT patrolled by”.
- The key is not random, but is a phrase that one might expect from those traitors making war on the United States.

Adding stuff

Adding bits

Is logic exclusive?

lor "You must be 17 or older *or* accompanied by an adult."

xor "The answer is 42 *or* it isn't."

	Inclusive	Exclusive
Short name	lor	xor
Logic	$x \vee y$	$(x \vee y) \wedge \neg(x \wedge y)$
Math/Crypto	$x + y$	$x \oplus y$
Prog. logic	`x || y`	`(x || y) && !(x && y)`
Prog. bits	`x	y`	`x ^ y`

Table 1: Or: inclusive and exclusive notation

For a bit

a	b	$a \oplus b$
0	0	0
1	0	1
0	1	1
1	1	0

Table 2: Single bit xor

For letters we did addition modulo 26. For bits, we are doing addition modulo 2.

Bit strings

When we xor sequences of bits (of the same length) we xor bit by bit, eg

$$1010 \oplus 1100 = 0110$$

Bit facts

Notation: '0^n' means a string of n bits, all of which are zero. Eg, $0^5 = 00000$.

Property	Definition	Example
Commutative	$a \oplus b = b \oplus a$	$01 \oplus 11 = 11 \oplus 01 = 10$
Associative	$(a \oplus b) \oplus c = a \oplus (b \oplus c)$	
Zero identity	$a \oplus 0^n = a$	$0110 \oplus 0000 = 0110$
Own inverse	$a \oplus a = 0^n$	$10 \oplus 10 = 00$

Table 3: Properties of xor: For all a, b, and c bit strings of length n.

Task

Convince yourself that

$$(a \oplus b) \oplus (c \oplus a) = c \oplus b$$

for any bit sequences a, b, and c of the same length.

You can do this either algebraically using the properties of xor, or you can do it by constructing a few examples and working through them.

Do both.

Task: Algebraic solution

$$
\begin{aligned}
(a \oplus b) \oplus (c \oplus a) &= (a \oplus b) \oplus (c \oplus a) && \text{Start} \\
&= a \oplus (b \oplus (c \oplus a)) && \text{Associative} \\
&= (b \oplus c) \oplus a) \oplus a && \text{Commutative} \\
&= (b \oplus c) \oplus (a \oplus a) && \text{Associative} \\
&= (b \oplus c) \oplus 0^n && \text{Own inverse} \\
&= b \oplus c && \text{0 identity} \\
&= c \oplus b && \text{Commutative}
\end{aligned}
$$

That was more than a bit pedantic in stepping through all of the commuting and re-associating. It is the identity and inverse properties that do the interesting stuff once you have re-arranged things.

Manipulating a pad: Dusk or dawn

k	‘FUBZETDBZOJJ’	$\mathcal{A}$ does not know
m	‘Attack at dawn’	$\mathcal{A}$ knows
c	‘FNUZGD DU COFW’	$\mathcal{A}$ knows
'DUSK' − 'DAWN'	‘AUWX’	$\mathcal{A}$ computes
'COFW' + 'AUWX'	‘CIBT’	$\mathcal{A}$ computes
c'	‘FNUZGDDU CUJZ’	$\mathcal{A}$ computes
m'	‘ATTACKATDUSK’	$\mathcal{A}$ knows

Table 4: Chosen ciphertext & malleability. $\mathcal{A}$ is the adversary; k is the key; m is the plaintext message; c is the ciphertext. m' and c' are alternative message and ciphertext.

Hex intro

Hexadecimal is just a more compact what of writing about bit sequences.

bits	*hex*	*bits*	*hex*	*bits*	*hex*	*bits*	*hex*
`0000`	`0x0`	`0100`	`0x4`	`1000`	`0x8`	`1100`	`0xC`
`0001`	`0x1`	`0101`	`0x5`	`1001`	`0x9`	`1101`	`0xD`
`0010`	`0x2`	`0110`	`0x6`	`1010`	`0xA`	`1110`	`0xE`
`0011`	`0x3`	`0111`	`0x7`	`1011`	`0xB`	`1111`	`0xF`

Table 5: Hexadecimal representation of four bits. Binary sequences are sometimes prefixed with '`0b`'. Hex is frequently prefixed with '`0x`'. Often context is your only clue.

Manipulating a pad: Which account

k	`0x64bdf13da0095afa`	key
m	`0x234142433132332e`	“#ABC123.”
c	`0x47fcb37e913b69d4`	$m \oplus k$
d	`0x00191b1905070500`	'#ABC123.' $\oplus$ '#XYZ456.'
c'	`0x47e5a867943c6cd4`	$c \oplus d$
m'	`0x2358595a3435362e`	“#XYZ456.”

Table 6: Chosen ciphertext & malleability using xor. k is the key; m is the plaintext message; c is the ciphertext; d is the difference (XOR) between m and what $\mathcal{A}$ wants to change it to. m' and c' are alternative message and ciphertext.

Computing the difference

The message is the ASCII *text* “Transfer 1,000,000.00USD to account #ABC123.” We want to change that to end with “#XYZ456.” by manipulating cipher text only.

m_i (char)	m_i (bits)	m'_i (char)	m'_i *(bits)*	$m_i \oplus m'_i$	hex
‘#’	`00100011`	‘#’	`00100011`	`00000000`	`0x00`
‘A’	`01000001`	‘X’	`01011000`	`00011001`	`0x19`
‘B’	`01000010`	‘Y’	`01011001`	`00011011`	`0x1B`
‘C’	`01000011`	‘Z’	`01011010`	`00011001`	`0x19`
‘1’	`00110001`	‘4’	`00110100`	`00000101`	`0x05`
‘2’	`00110010`	‘5’	`00110101`	`00000111`	`0x07`
‘3’	`00110011`	‘6’	`00110110`	`00000101`	`0x05`
‘.’	`00101110`	‘.’	`00101110`	`00000000`	`0x00`