Galaxy_TOC

Text-XOR (TXOR)

TXOR is an encryption algorithm that differs from the one-time pad cipher only by a fact that in stead of using only 2 different values, the "true" and the "false", 1 and 0, {0,1}, the number of values that the TXOR uses can be any whole, finite, number that is greater than or equal to 2.

Motivation

The main motivation for developing the TXOR has been to use an encryption algorithm that has the security properties of the one-time pad, but that is computationally efficient to use within text processing oriented scripting languages like PHP, JavaScript, Ruby.

The targeted use case of the TXOR is to use the maximum Unicode code value in stead of the classical "1" of the XOR based one-time pad.

As the range, **[0,max(Unicode codes)]**, contains numbers that have not been assigned to any existing character, cipher-texts can not be saved in text form without using some special encoding. For example, the cipher-text might be encoded as a comma-separated list of integers.

The Algorithm

TXOR(aa,bb,m)=((bb-aa+m) mod m)

Prerequisites:
$2 \leq m$
$0 \leq aa < m$
$0 \leq bb < m$

The aa,bb,m are whole numbers.

The **bb must always be in the role of a key** and
the **aa must always be** either
in the role of a **clear-text** or
in the role of a **cipher-text**.

An example:

ciphertext = TXOR(**cleartext**,**key**,m)
cleartext = TXOR(**ciphertext**,**key**,m)

Galaxy IOC

Various Checks

Property h1

0 ≦ (bb-aa+m)

According to prerequisites

$$2 \leqq m$$
$$0 \leqq aa < m$$
$$0 \leqq bb < m$$

Therefore the (bb-aa) has its smallest value when bb=0 and aa=(m-1).
0≦(bb-aa+m) because 0≦((0-(m-1))+m)=1

Property h2

0 ≦ ((bb-aa+m) mod m)
id est
0 ≦ TXOR(aa,bb,m)

According to the property h1 0 ≦ (bb-aa+m). According to the prerequisites (0 < 2 ≦ m).
A remainder of a positive dividend and a positive divisor is a positive number.

Property h3

((bb-aa+m) mod m) < m
id est
TXOR(aa,bb,m) < m

According to the property h2 and the prerequisites both, m and the ((bb-aa+m) mod m), are positive numbers.
((bb-aa+m) mod m) is a remainder of a division ((bb-aa+m) mod m)/m.
If the dividend and the divisor are both positive numbers and the divisor is greater than zero, then a remainder is always smaller than the divisor.

Property h4

0≦ ((bb- ((bb-aa+m) mod m) +m) mod m) < m
and
0≦ ((((bb-aa+m) mod m) -aa+m) mod m) < m
id est

Galaxy_TOC

0≦ TXOR(TXOR(aa,bb,m) ,bb,m) < m
and
0≦ TXOR(aa, TXOR(aa,bb,m) ,m) < m

According to properties h2 and h3 **0≦ TXOR(aa,bb,m) < m**
Therefore, the **TXOR(aa,bb,m)** is within the same bounds as the **aa** and the **bb**.
and the same contemplation that showed that **0≦ TXOR(aa,bb,m) < m** applies for showing that
0≦ TXOR(TXOR(aa,bb,m) ,bb,m) < m
and
0≦ TXOR(aa, TXOR(aa,bb,m) ,m) < m

Property #1aa

aa = TXOR(TXOR(aa,bb,m) ,bb,m)
id est
If **bb** is a key and **aa** is the cleartext, then the decryption result always matches with the encrypted cleartext.

ciphertext = TXOR(aa,bb,m) = ((bb-aa+m) mod m)

TXOR(TXOR(aa,bb,m),bb,m) = TXOR(ciphertext,bb,m) = ((bb-ciphertext+m) mod m) =

= ((bb-**((bb-aa+m) mod m)**+m) mod m) = g

If aa = g, then the Property #1aa holds.
If the formula of the **g**, the
g = ((bb-{ ((bb-aa+m) mod m) }+m) mod m)
transforms to aa, then **aa = g** can be transformed to **aa = aa**
and the condition **aa = g** is met.

bb-((bb-aa+m) mod m)+m-g=c1*m

-((bb-aa+m) mod m)=c1*m-m+g-bb
-((bb-aa+m) mod m)=(c1-1)*m+g-bb
-((bb-aa+m) mod m)=c2*m+g-bb

((bb-aa+m) mod m)=bb-g-c2*m

(bb-aa+m)-(bb-g-c2*m)=c3*m
bb-aa+m-bb+g+c2*m=c3*m
-aa+g+(c2+1)*m=c3*m
-aa+g=c3*m-(c2+1)*m
-aa+g=c3*m-c4*m
-aa+g=c5*m

aa=(g mod m)

According to the preconditions, **0 ≦ aa < m** and
according property h4 **0 ≦ g < m**.
Therefore, **g = aa**.

Q.E.D.

Property #1bb

Galaxy_TOC

Generally **bb = TXOR(aa, TXOR(aa,bb,m) ,m)** does not hold.
id est
If **aa** is a key and **bb** is the cleartext, then the decryption result might not match with the encrypted cleartext.

bb = TXOR(aa, TXOR(aa,bb,m) ,m) can be shown to be false by finding one example, where it is false.

The example values are:

aa = key = 2

bb = clear-text = 6

m = 10

ciphertext = TXOR(aa,bb,m) = ((bb-aa+m) mod m) = ((6-2+10) mod 10) = 14 mod 10 = 4

TXOR(aa, TXOR(aa,bb,m) ,m) = TXOR(aa, ciphertext, m) = decryption-result =

= ((ciphertext-aa+m) mod m) = ((4-2+10) mod 10) = (12 mod 10) = 2

2 != 4

Q.E.D.

Property #2com

In respect to aa and bb the TXOR(aa,bb,m) is not commutative.
id est
There exist cases, where TXOR(x,y,m)!=TXOR(y,x,m)

Sample values:

x=2, y=6, m=10

TXOR(x,y,m) = (y-x+m) mod m) = ((6-2+10) mod 10) = 4
TXOR(y,x,m) = (x-y+m) mod m) = ((2-6+10) mod 10) = 6
4 != 6

Q.E.D.

That is to say that

XOR(x,y) = XOR(y,x)
but
TXOR(x,y,m) != TXOR(y,x,m)

Property #3stat

If cleartext is a constant, then ciphertexts of different keys never collide.

GaLaxy_Toe
id est
If bb1 != bb2, then TXOR(aa,bb1,m) != TXOR(aa,bb2,m)

TXOR(aa,bb**1**,m) = ((bb1-aa+m) mod m) = ((bb1+c1) mod m)
TXOR(aa,bb**2**,m) = ((bb2-aa+m) mod m) = ((bb2+c1) mod m)

Due to the "mod m" part of the equations the
((bb1+c1) mod m) equals with the ((bb2+c1) mod m) only,
if bb1=bb2+c2*m. Due to one of the prerequisites, the
(0 ≦ bb < m),
the c2 = 0, because otherwise m ≦ bb2, which would violate the prerequisite.

If the TXOR(aa,bb1,m) = TXOR(aa,bb2,m) only, when the bb1 = bb2, then
TXOR(aa,bb1,m) != TXOR(aa,bb2,m), whenever bb1 != bb2.

Q.E.D.

Consequently, for any clear-text value there exists as many possible ciphertext values
as there are values in the key-space,
(0 ≦ bb < m), number of possible keys is (m-1),
and the same kind of unbreakability_proof applies as is used
for the one-time pad.

Historical Notes

I martin.vahi@softf1.com, first published the TXOR idea as a comment at minut.ee
sometime before the year 2013.

This, relatively cleanly written, HTML version of the TXOR specification has been
written in August 2013. The presented TXOR algorithm is probably nothing new or
original, but I figured it out myself by extending the classical one-time pad by
utilising the classical modulo arithmetic.

As free advertisement is always beneficial to freelancers like me and it had happened
to me before that a simplistic, self-figured-out, algorithm turned out to be totally
unique, I link a signed (read: timestamped) version of this document into itself.

2021_12_31 comment: In 2013_08 I received some very nice feedback on
LWN.net*(2021_12_31_archival_copy)*. Part of the criticism seems to be that with
symmetric key crypto-algorithms it is so difficult to exchange keys that symmetric key
crypto-algorithms are seen as impractical for web applications. Another part of the
criticism seems to be that the conversion between character codes, whole numbers, and
character encoding bitstreams seems to be so difficult to avoid that an encryption
algorithm that is specifically designed to work with whole numbers does not offer
enough benefits from speed/efficiency point of view. It might be that I misunderstood
the feedback that I received.

Thank You for reading this HTML-page. :-)
