XOR cipher

In cryptography, the **simple XOR cipher** is a type of *additive cipher*,[1] an encryption algorithm that operates according to the principles:

$A \oplus 0 = A$,

$A \oplus A = 0$,

$(A \oplus B) \oplus C = A \oplus (B \oplus C)$,

$(B \oplus A) \oplus A = B \oplus 0 = B$,

where $\oplus$ denotes the exclusive disjunction (XOR) operation. This operation is sometimes called modulus 2 addition (or subtraction, which is identical).[2] With this logic, a string of text can be encrypted by applying the bitwise XOR operator to every character using a given key. To decrypt the output, merely reapplying the XOR function with the key will remove the cipher.

Contents

Example

For example, the string "Wiki" (`01010111 01101001 01101011 01101001` in 8-bit ASCII) can be encrypted with the repeating key `11110011` as follows:

```
  01010111 01101001 01101011 01101001
⊕ 11110011 11110011 11110011 11110011
= 10100100 10011010 10011000 10011010
```

And conversely, for decryption:

```
  10100100 10011010 10011000 10011010
⊕ 11110011 11110011 11110011 11110011
= 01010111 01101001 01101011 01101001
```

The XOR operator is extremely common as a component in more complex ciphers. By itself, using a constant repeating key, a simple XOR cipher can trivially be broken using frequency analysis. If the content of any message can be guessed or otherwise known then the key can be revealed. Its primary merit is that it is simple to implement,

and that the XOR operation is computationally inexpensive. A simple repeating XOR (i.e. using the same key for xor operation on the whole data) cipher is therefore sometimes used for hiding information in cases where no particular security is required.

If the key is random and is at least as long as the message, the XOR cipher is much more secure than when there is key repetition within a message.[3] When the keystream is generated by a pseudo-random number generator, the result is a stream cipher. With a key that is truly random, the result is a one-time pad, which is unbreakable even in theory.

In any of these ciphers, the XOR operator is vulnerable to a known-plaintext attack, since *plaintext* ⊕ *ciphertext* = *key*. It is also trivial to flip arbitrary bits in the decrypted plaintext by manipulating the ciphertext. This is called malleability.

Example implementation

Example using the Python programming language.[4]

```
from __future__ import print_function, unicode_literals
from os import urandom


def genkey(length):
    """Generate key"""
    return urandom(length)


def xor_strings(s, t):
    """xor two strings together"""
    if isinstance(s, str):
        # Text strings contain single characters
        return "".join(chr(ord(a) ^ ord(b)) for a, b in zip(s, t))
    else:
        # Python 3 bytes objects contain integer values in the range 0-255
        return bytes([a ^ b for a, b in zip(s, t)])


message = 'This is a secret message'
print('message:', message)

key = genkey(len(message))
print('key:', key)

cipherText = xor_strings(message.encode('utf8'), key)
print ('cipherText:', cipherText)
print ('decrypted:', xor_strings(cipherText, key).decode('utf8'))

# verify
if xor_strings(cipherText, key).decode('utf8') == message:
    print('Unit test passed')
else:
    print('Unit test failed')
```

See also

- Vernam cipher
- Vigenère cipher

References

1. Tutte 1998, p. 3
2. Churchhouse 2002, p. 11
3. Churchhouse 2002, p. 68
4. This was inspired by Richter, Wolfgang (August 3, 2012), "Unbreakable Cryptography in 5 Minutes" (http://xrds.acm.org/blog/2012/08/unbreakable-cryptography-in-5-minutes/), *Crossroads The ACM Magazine for Students*, Association for Computing Machinery

Bibliography

- Churchhouse, Robert (2002), *Codes and Ciphers: Julius Caesar, the Enigma and the Internet*, Cambridge: Cambridge University Press, ISBN 978-0-521-00890-7
- Tutte, W. T. (19 June 1998), *Fish and I* (http://cryptocellar.web.cern.ch/cryptocellar/tutte.pdf) (PDF), retrieved 7 October 2010 Transcript of a lecture given by Prof. Tutte at the University of Waterloo

External links

- XOR encryption for text files on windows with source code (http://sourceforge.net/projects/xorencrypt2/)
- Solving the Basic XOR Cipher (https://web.archive.org/web/20130413230658/http://chris.dod.net/xor/xor.php)