**Affine Cipher**

**Introduction**

The Affine cipher is a special case of the more general [monoalphabetic substitution](http://practicalcryptography.com/ciphers/monoalphabetic-substitution-category/) cipher.

The cipher is *less* secure than a [substitution cipher](http://practicalcryptography.com/ciphers/substitution-category/) as it is vulnerable to all of the attacks that work against substitution ciphers, in *addition* to other attacks. The cipher's primary weakness comes from the fact that if the cryptanalyst can discover (by means of frequency analysis, brute force, guessing or otherwise) the plaintext of two ciphertext characters, then the key can be obtained by solving a simultaneous equation [[1]](http://practicalcryptography.com/ciphers/affine-cipher/#references).

**The Algorithm**

The 'key' for the Affine cipher consists of 2 numbers, we'll call them *a* and *b*. The following discussion assumes the use of a 26 character alphabet (*m* = 26). *a* should be chosen to be relatively prime to *m* (i.e. *a*should have no factors in common with *m*). For example 15 and 26 have no factors in common, so 15 is an acceptable value for *a*, however 12 and 26 have factors in common (e.g. 2) so 12 cannot be used for a value of *a*. When encrypting, we first convert all the letters to numbers ('a'=0, 'b'=1, ..., 'z'=25). The ciphertext letter *c*, for any given letter *p* is (remember *p* is the number representing a letter):

http://practicalcryptography.com/media/latex/f202d3bbb5a0c3f972ed5a19f4f4cc2948ea7d01-11pt.png

The decryption function is:

http://practicalcryptography.com/media/latex/d5912326d7d36169b0b61de649b7769fb8c12a24-11pt.png

where *a*−1 is the multiplicative inverse of *a* in the group of integers modulo m.

To find a multiplicative inverse, we need to find a number *x* such that:

http://practicalcryptography.com/media/latex/85704a37321703dc18d617451ec8925aaf59b809-11pt.png

If we find the number *x* such that the equation is true, then *x* is the inverse of *a*, and we call it *a*−1. The easiest way to solve this equation is to search each of the numbers 1 to 25, and see which one satisfies the equation. If you want a more rigorous solution, you can use matlab to find *x*:

> [g,x,d] = gcd(a,m); % we can ignore g and d, we dont need them

> x = mod(x,m);

If you now multiply *x* and *a* and reduce the result (mod 26), you will get the answer 1. Remember, this is just the definition of an inverse i.e. if *a*\**x* = 1 (mod 26), then *x* is an inverse of *a* (and *a* is an inverse of *x*).

We now use the value of *x* we calculated as *a*-1. This allows us to perform the decryption step.

**Note:** As stated above, *m* does not have to be 26, it is simply the number of characters in the alphabet you choose to use. If upper case characters, lowercase characters and spaces are used, then m will be 53. Digits and punctuation could also be incorporated (which again would change the value of *m*).

Assume we discard all non alphabetical characters including spaces. Let the key be *a*=5 and *b*= 7. The encryption function is then (5\*p + 7)(mod 26). To encode:

'defend the east wall of the castle',

we would take the first letter, 'd', convert it to a number, 3 ('a'=0, 'b'=1, ..., 'z'=25) and plug it into the equation:

http://practicalcryptography.com/media/latex/65d02253cdd1654326cbf7219a11409facc7e0b3-11pt.png

since 'w' = 22, 'd' is transformed into 'w' using the values *a*=5 and *b*= 7. If we continue with all the other letters we would have:

'wbgbuwyqbbhtynhkkzgyqbrhtykb'

Now to decode, the inverse of 5 modulo 26 is 21, i.e. 5\*21 = 1 (mod 26). The decoding function is

http://practicalcryptography.com/media/latex/d5364509bd35df3c91fdd58785a4f340c1070330-11pt.png

so we have recovered d=3 as the first plaintext character.

'defendtheeastwallofthecastle'

Bottom of Form

**Other Implementations**

To encipher your own messages in python, you can use the [pycipher](http://www.practicalcryptography.com/miscellaneous/pycipher/installing-pycipher/) module. To install it, use pip install pycipher. To encipher messages with the Affine cipher (or another cipher, see [here for documentation](http://pycipher.readthedocs.org/en/latest/)):

>>>from pycipher import Affine

>>>Affine(a=5,b=9).encipher('defend the east wall of the castle')

'YDIDWYASDDJVAPJMMBIASDTJVAMD'

>>>Affine(a=5,b=9).decipher('YDIDWYASDDJVAPJMMBIASDTJVAMD')

'DEFENDTHEEASTWALLOFTHECASTLE'

**Cryptanalysis**

See [Cryptanalysis of the Affine Cipher](http://practicalcryptography.com/cryptanalysis/stochastic-searching/cryptanalysis-affine-cipher/) for a guide on how to break this cipher automatically.

The Affine cipher is a very insecure cipher, with the [Caesar cipher](http://practicalcryptography.com/ciphers/caesar-cipher/) possibly being the only easier cipher to crack. The Affine cipher is a monoalphabetic substitution cipher, so all the methods that are used to cryptanalyse substitution ciphers can be used for the affine cipher. Affine ciphers can also be cracked if any 2 characters are known.

As an example, imagine we have a ciphertext. If the 2 most common characters in the ciphertext are 'h' and 'q', then we can assume that these correspond to 'e' and 't' in the plaintext. We can set up a simultaneous equation ('h' -> 'e' and 'q' -> 't'), the following 2 equations are simply two instances of the affine cipher where we know (or assume we know) the values of the plaintext character and the corresponding ciphertext character for 2 cases, but do not know *a* or *b* (In the following equation we have converted letters to numbers, 'e'=4, 'h'=7, 'q'=16, 't'=19):

http://practicalcryptography.com/media/latex/cd896d4d8ad76d8ad5b705781ad33fef1c84bb45-11pt.png

For the following discussion we will refer to the more general set of equations:

http://practicalcryptography.com/media/latex/ae70ca4e82cd3b30b525afd9b1a28b442dc1050d-11pt.png

Solving systems of equations modulo 26 is slightly more difficult than solving them normally, but it is still quite easy. We know the values *p*, *q*, *r* and *s*, and we wish to find *a* and *b*. We must first find the number *D*= *p* - *q*, and *D*-1 (the inverse of *D*). *D*-1 is found by looping through the numbers between 1 and 25 until you find a number, *x*, such that *D*\**x* = 1 (mod 26). We can now find the value of *a* and *b*.

http://practicalcryptography.com/media/latex/cbdb3ee2567102128c27ddb47c8106e4aebd1481-11pt.png

Using the example we started with, p=4, r=7, q=19, s=16. *D* = p-q = -15 = 11 (mod 26). This means *D*-1= 19. So:

http://practicalcryptography.com/media/latex/cb93e69d85b2225eaf14967db046f0765c7e326a-11pt.png

From this we would conclude that the *a*, *b* pair used to encrypt the plaintext was 11 and 15 (this represents the key), respectively. If we decrypt the cipher text under this assumption, we can see if these are correct. If they are, that is the end, otherwise we could try other combinations of common cipher text letters instead of our guess of 'e' and 't'. This method is much easier to perform if you have a program that performs these steps automatically.

**References**

* [1] [Wikipedia](http://en.wikipedia.org/wiki/Affine_cipher) has a good description of the encryption/decryption process, history and cryptanalysis of this algorithm
* [2] A decent overview of the affine cipher:<http://www.math.sunysb.edu/~scott/Book331/Affine_enciphering.html>
* Singh, Simon (2000). The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography. ISBN 0-385-49532-3.