**MATHS IA**

**Mathematical cryptanalysis of modern and ancient crypt algorithms:**

Introduction:

In the last year there were many news items that demonstrated hackers attacking websites, stealing and changing personal information by breaking passcodes, and installing dangerous malware on computer systems. In recent times there’s been a viral, state-sponsored ransomware, leaks of spy tools from US intelligence agencies.

Every year a new story about hacker’s attacks and latest viruses emerges. It was very unpleasant for me to find that my email account was hacked. This left me thinking that I should investigate the strength of modern password, and understand the mathematics behind the modern cryptographic algorithms and learn the ways these algorithms can be attacked.

**Cryptanalysis of modern passwords:**

Everyone protects their information using passwords on websites, home computers, at offices etc. Every year hackers break into these passwords and there needs to be a way to protect these passcodes by encrypting them.

Method for cryptanalysis:

Out of 4 characters, need to make 2 char password.

a,b,c,d

ab ac ad ba bc bd ca cb cd da db dc

without repetition it is 12 combinations. We can find the number of combinations by doing 4C2 \* 2! = 12

ab ac ad ba bc bd ca cb cd da db dc aa bb cc dd

With repetition we will get 16 combinations. To find these number of combinations we do 2^4= 16

For selecting practical passwords following is the general criteria:

* numbers (10 different ones: 0-9)
* letters (52 different ones: A-Z and a-z)
* special characters (33 different ones)

In total unique characters = 95

If repetitions are allowed the total combinations of 8-character password that are possible are

Lets take another example: out of 8 characters we need to make a 4 char password. It is not that efficient to find each and every combination as it will take a long time since there are so many combinations. So we’re going to use the same technique as used in the previous example:

Without repetition: 8C4 \* 4! = 70 \* 24= 1680

With repetion: 4^8= 65536

**Brute force attack**:

 A trial and error method used by application programs to decode encrypted data such as passwords.

Here is a table showing the number of possible combinations (with repetition of characters) of various password sizes out of 90 character set. In this character set there are following groupings:

* 26 lowercase characters [a-z]
* 26 uppercase characters [A-Z]
* 10 number digits [0-9]
* 28 special characters[!$#@&\*…]

For finding total number ofcharacter passwords we can arrive at following formula from the permutation and combinations that includes repetitions

|  |  |  |  |
| --- | --- | --- | --- |
| Password length(n) | Number of combinations of passwords | Brute force attack avge time /s | Time taken for 100 machines/s |
| 2 |  |  |  |
| 4 |  |  |  |
| 6 |  |  |  |
| 8 |  |  |  |
| 10 |  |  |  |
| 12 |  |  |  |

For accurately finding out the number of combinations of the passwords with restrictions to choose 8 character password with minimum one lower case, one upper case, one digit and one special character could be empirically determined by the following calculation:

All 8 character string :

Then remove all combinations with no uppercase , all passwords with no lowercase , all possible combinations with no digits , all passwords with no special characters

But then you removed some passwords twice. You must add back all passwords with:

* no lower case and no uppercase :
* no lowercase and no digit:
* no lowercase and no special character :
* no uppercase and no digit:
* no uppercase and no special character:
* no digit and no special character:

But then you added back a few passwords too many times. For instance, an all-digit password was removed three times in the first step, then put back three times in the second step, so it must be removed again:

* only uppercase
* only lowercase
* only digit
* only special character

Total =  **possible combinations**

However is not a completely accurate value. This is because if for example the punctuation special character first appears in the fifth position in the password with the first lower case letter, first upper case letter, and first digit appearing in the first 3 positions of the password then there are only characters that can appear in the fourth position of the password.

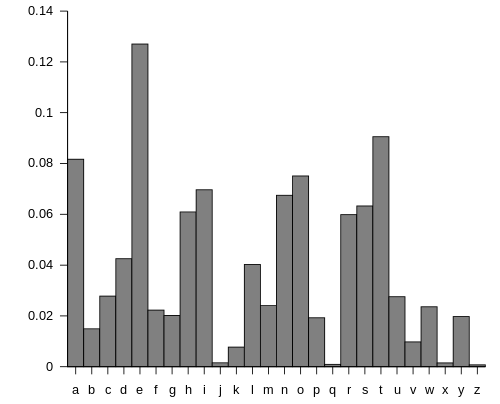
Dictionary attack: a method used of trying to determine a cipher’s/authentication’s decryption key or password by trying each and every word in the dictionary.

**Cryptanalysis of ancient crypto algorithms**.

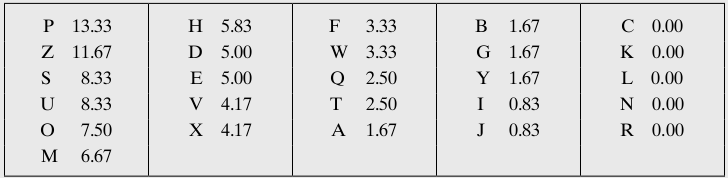
There are various types of Ancient crypto algorithms .

Frequency analysis of letters in the english language:

 Frequency analysis is the study of the [frequency of letters](https://en.wikipedia.org/wiki/Letter_frequencies" \o "Letter frequencies) or groups of letters in a [ciphertext](https://en.wikipedia.org/wiki/Ciphertext" \o "Ciphertext). Frequency analysis is based on the fact that, in any given length of written language, certain letters and combinations of letters occur more than once.



This diagram shows the relative frequency of letters in Plaintext.

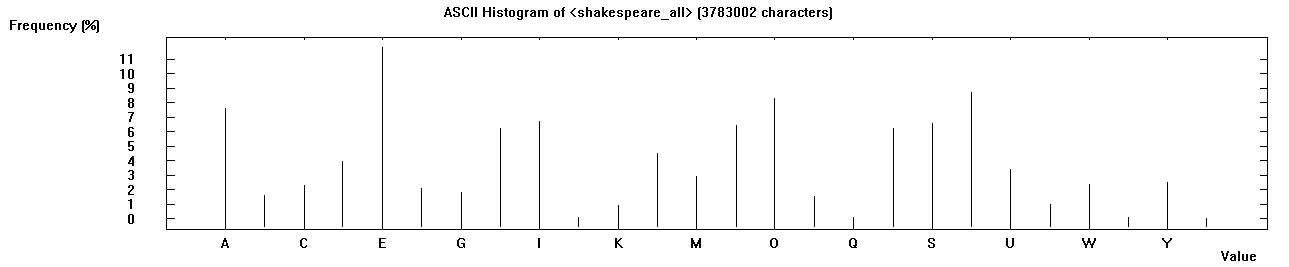


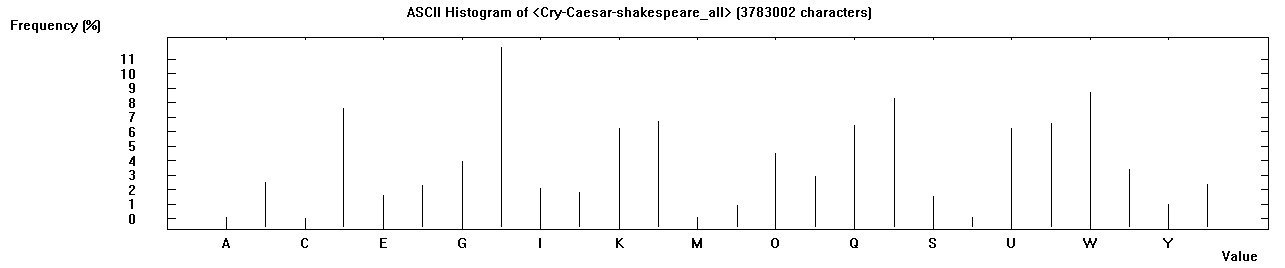
This table shows the relative frequency of letters in Cipher text.

From the 2 diagrams we can see that the cipher letters P and Z are equal to the plain text letters e and t as they have the highest frequency. The ciphertext letters S,U,O,M,H probably correspond the plaintext letters of a,h,n,o,r,s as they are all of relatively high frequencies. The ciphertext letters with lowest frequency A,B,G,Y,I,J probably correspond to the plaintext letters of b,j,k,q,v,x,z

Some examples of substitution ciphers:

Cryptanalysis of entire Shakespeare’s plays encrypted with Caesar cipher





If we compare the two graphs we can see that most frequently found letter has shifted to H from E, and Second most frequently found letter T has shifted to W. Another most frequently found letter O has shifted to R. This give us a clear understanding that the encryption used has shifted the letters by 3 to the right.

**Mathematics behind modern crypto algorithms:**

**RSA Algorithm:**

**Mathematical Background:**

* Natural Number – a number from the set of {1,2,3,.., }
* Integer – a number in the set {,L,2,1,0,1,2,L, }
* Composite Number – a natural number that is not prime.
* Prime Number – a natural number that is only divisible by one and itself.
* Greatest Common Divisor – the largest common factor of a set of numbers.
* Coprime – a set of numbers is coprime if their greatest common divisor is one
* Relatively Prime – the same thing as coprime.

**Totative** – a number, *m*  *n* , is a totative of *n* if gcd(*m*, *n*)  1, where *n* is a natural number.

* Gives a count of how many number in the set {1,2,3, 4,5,..,..,n} share no common factors with n that are not greater than 1 for example :
* For example, the totatives of n = 10 are {1, 3, 4, 7,8 and 9}

All of these numbers are co-prime to 10.

* For example, n=7 the totatives are {1,2,3,4,5,6} are all co-primes of prime number 7.

**RSA modern crypto algorithm:**

RSA makes use of an expression with exponentials. Plaintext is encrypted in blocks, with each block having a binary value less than some number n. That is, the block size must be less than or equal to

Encryption and decryption are of the following form, for a plaintext block M and Ciphertext block C.

Both the sender and reciever should know the value of N. Only the reciever knows the value of *d* and the sender knows the value of *e* .

Some requirements need to be done:

1. It is possible to find values of e, d, and n such that for all M< N.
2. It is relatively easy to calculate and
3. It is infeasible to determine d given e and n.

We need to find a relationship of the form :

Euler totient function:

The number of integers which are coprime to a positive integer *n*, between 1 and *n*, is given by [Euler's totient function](https://en.wikipedia.org/wiki/Euler%27s_totient_function" \o "Euler's totient function) or Euler's phi function

RSA use of totient function:

Euler’s generalization of Fermat’s little theorem:

M and *n* are co-prime. Since the L.H.S. is congruent to one, we can raise it to any power, *x* without changing the value:

At last we multiply both LHS and RHS by M. The multiplication by M removes the co-prime requirement between M and *n*.

The modulator exponentiation to a power of is an identity operation.

RSA method is a way of splitting this identity operation into two steps: encryption and decryption.

To ensure that any message that is encrypted and then decrypted is the original message, we have:

Equate the earlier identity due to Euler:

And the encryption/decryption requirement

We get:

For RSA, by putting in the specific value for the totient, we can find for the exponent relation:

This equation shows how one needs the prime factors of n to find d. Which is done through the RSA algorithm.

Relationship between *e* and *d* can be written as:

Step by step RSA algorithm:

1. Select 2 prime numbers, p=15 and q=19
2. Calculate N = 15 = 285
3. Calculate
4. Select *e* such that *e* is relatively prime to and less than ; we choose *e* = 5
5. Determine

(252 2) +1 = 5*d*

Encryption or decryption of messages using RSA key pair:

Once you have generated the RSA key, you can encrypt and decrypt messages.

For example, the **encryption of the message ATTACK AT DAWN.**

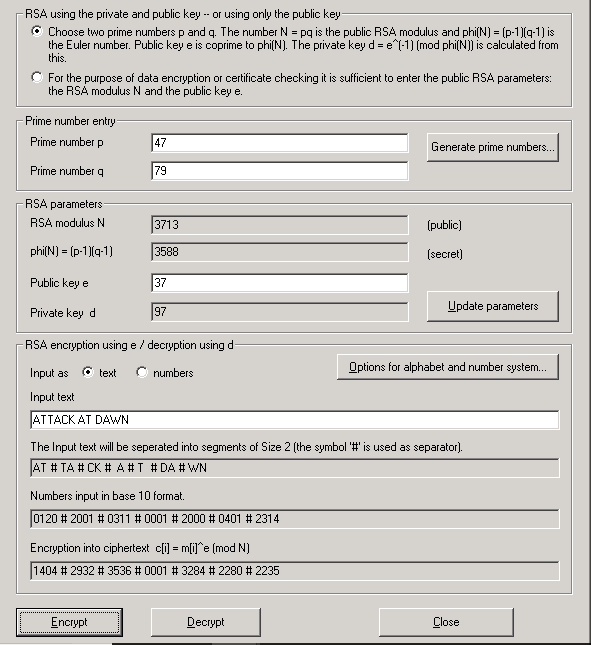
First of all generate the RSA key using the following:

p=47, q=79 e=37.

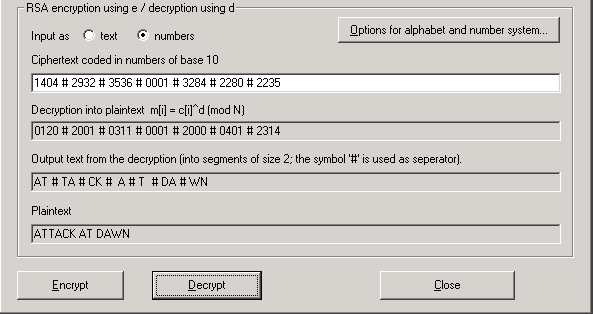
parameters N = p\*q = 3713, phi(N) = 3588 and d = 97 are calculated.

The encrypted version of this is the number sequence

1404 # 2932 # 3536 # 0001 # 3284 # 2280 # 2235



The number “#”, serves here to visually split up the individual numbers. If you insert these numbers into the input line and then choose **Decrypt**, the original plaintext will be restored.



Attack on RSA Algorithm:

As well as the ciphertext, the attacker also has access to the public key e and the RSA modulus N.

The following examples describe attacks for the factorization of public RSA modulus N. If the factorization is successful, then all the secret parameters of the RSA cryptosystem can be calculated, and these can then be used to crack the ciphertext challenge

**Cryptanalysis of RSA algorithm:**

The public key is the pair {e,n}and the private key is the pair {d,n}. The security stems from the linkage between d and e being through the prime factors of n.

If n is made up as the product of two very large prime numbers, then the difficulty of factoring n is what stops one from finding d when given e and n.

The RSA method exploits the fact that factoring a large number into smaller components is more difficult than forming a large number by multiplying together smaller numbers.

If the large number is a product of two near equal length primes, then this computational disparity can be quite extreme. This is an example of a one way trap door function. To ensure the one wayness of this function, RSA type cryptosystems use numbers with thousands of digits.

The crypt analysis is performed in two stages:

1. The prime factorization of the RSA modulus is calculated.
2. In the second stage the secret key for encryption of the message is determined. After this, the ciphertext can be decrypted with the cracked secret key.

Factorization of the RSA modulus with the aid of prime factorization:

1. The natural number N must be broken down to its prime factors p and q. For example *N* = 31313 into its prime factors p and q (173 & 181)
2. Calculate the secret key d from the prime factorization of N and the public key e: With knowledge of the prime factors p = 173 and q = 181 and the public key *e* = 4913, you are now in a position to decrypt the ciphertext.
3. The Euler phi function phi(N) = (p-1)\*(q-1) is calculated from the numbers p and q, and from this and public key e the secret RSA key d is determined.

Prime factor p = 173

Prime factor q = 181

RSA modulus N = 31313

phi(N) = (p-1)\*(q-1) = 30960

Public key e = 4913

Private key d = 6497

Different algorithms for factorization and time taken for each:

RSA modulus N = 3849371651547409827462738464856873573625

|  |  |
| --- | --- |
| Algorithm Type | Time taken to find the factors |
| Brute-force method | 0.012 seconds |
| Brent Algorithm | 0.055 seconds |
| Pollard method | 0.120 seconds |
| Williams method | 0.272 seconds |
| Lenstra Algorithm | 0.384 seconds |
| Quadratic Sieve method | 0.660 seconds |

Brute force takes the least amount of time to find the factors. Therefore, it is most efficient.

I wanted to explore why the number of prime numbers reduces as the value becomes larger.

To understand the distribution for prime numbers between 2^62 and 2^64,

Gauss and Legendre conjectured the rate of growth of prime numbers.

Prime numbers become less common as numbers become larger.

prime-counting function is the [function](https://en.wikipedia.org/wiki/Function_(mathematics)) counting the number of [prime numbers](https://en.wikipedia.org/wiki/Prime_number) less than or equal to some [real number](https://en.wikipedia.org/wiki/Real_number) x.

Let π(x) be the number of primes less than x.

π(x) =

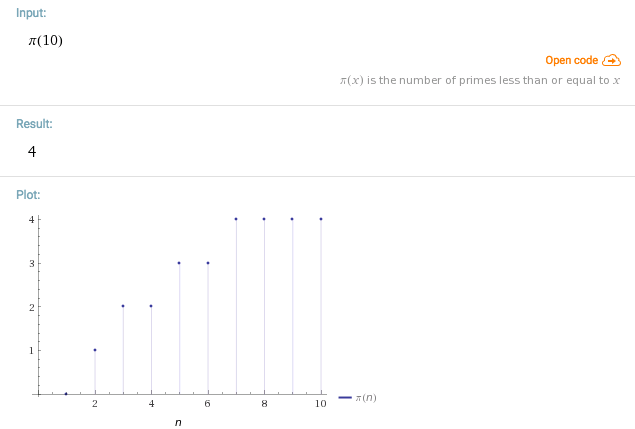
OR

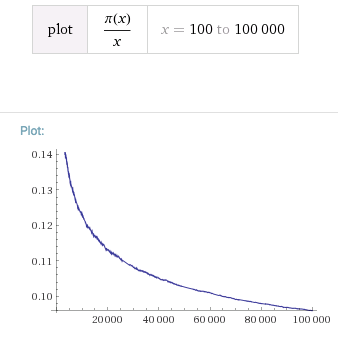
Asymptotic law of distribution of prime numbers.

π(x)=

π(10)= ≈ 4

because there are four prime numbers (2, 3, 5 and 7) less than or equal to 10.

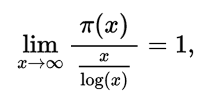




As x keeps increasing the number of prime numbers π(x) keeps reducing . Longer the number less is the probability of finding a prime number.That is why encyrption keys consist of many characters and therefore difficult to guess.

The prime number theorem then states that

 is a good approximation to π(x), in the sense that the [limit](https://en.wikipedia.org/wiki/Limit_of_a_function) of the quotient of the two functions π(x) and  as x increases without bound is 1:



In the following table I used different values of x and calculated the actual value of to prove that its not equal to 1. Then I calculated the percentage error from 1. We can see that as the value of x exponentially increases by the percentage error keeps reducing.

|  |  |  |  |
| --- | --- | --- | --- |
| Values of x |  |  | Percentage error |
|  | 168 | 1.1605 | 16.05% |
|  | 78498 | 1.0844 | 8.44% |
|  | 50847534 | 1.05372 | 5.37% |
|  | 37607912018 | 1.0391 | 3.91% |
|  | 29844570422669 | 1.0308 | 3.08% |
|  | 24739954287740860 | 1.0254 | 2.54% |

Conclusion:

The RSA cryptosystem is the de facto standard for public-key encryption and signature worldwide. It is implemented in the most popular security products and protocols in use today, and can be seen as one of the basis for secure communication in the Internet.

Till to date a  computationally efficient  method has not yet been found to factorize the products of large prime numbers. In practice when people use 1024-bit RSA modulus, the private exponent d should be at least 300 bits in length. If the public exponent e is chosen to be 65,537 (the most commonly used value), and we calculate d as de º 1 mod •(n), then we are guaranteed to have d nearly as long as n, and for this size attack should not pose a threat. Longer the numbers the more difficult it is to break them.

Its underlying function and properties have been extensively studied by mathematicians and security professionals for more than a quarter of a century.

However, many applications are still protected by the passwords of small length. Hackers have continued to exploit this weakness in the internet applications. Ancient ciphers are no longer in use and cryptanalysis has revealed various ways of breaking it easily.

Mathematics has clearly made such a huge influence and impact over the modern e-commerce that it would be completely insecure without it.