On the Usage and Implementation of Normal Irrational Numbers in Symmetric-Key Cryptography

Michael Jan

Great Neck North High School

June 8, 2016

Jan, Michael

On the Usage and Implementation of Normal Irrational Numbers in Symmetric-Key Cryptography

Abstract

Cryptography, the study of various techniques to securely communicate between two parties in the presence of a third party, has presumably been practiced as early as the 1900s B.C. Ever since then, a cryptographic arms race has been raging and still is to this day. Every time a new cipher is invented, inevitable weaknesses are discovered by exploiters and the cipher is rendered useless. Countless other ciphers have been invented, some more secure than others. Secure modern ciphers used all over the world today include the asymmetric-key RSA, and the symmetric-key AES. However, it is predicted that in approximately 20 to 30 years, quantum computing advancements may be able to break these ciphers in a matter of seconds, which is why new cryptosystems must be developed, in order to keep data secure. Thus, this study proposes implementations and performs basic cryptanalysis on a type of a fairly new and little-researched symmetric-key stream cipher based off of the binary expansion of irrational numbers. During this study, time complexity (running time) was examined for encryption and decryption. Also, bit randomness was analyzed through specialized algorithms provided by National Institute of Standards and Technology (NIST), which determined that the bits of the tested irrationals were sufficiently random and thus fit for encryption. Lastly, the brute-force attack method was analyzed, and it was found that a hypothetical irrational number cipher with both key parts being 128-bit would take roughly 10^{54} centuries to break.

On the Usage and Implementation of Normal Irrational Numbers in Symmetric-Key Cryptography

Introduction

People have been attempting to send secret messages to one another in the presence of third parties presumably as early as the 1900s BC. The study of this type of secure communication is known as cryptography, which is still used in the modern era for a multitude of different reasons. However, many basic ciphers such as the Caesar cipher and the Vigenère cipher are easily breakable. The problem lies in the short and rudimentary key, which induces patterns in the outputted plaintext. More advanced and modern cryptosystems, such as the asymmetric-key RSA and the symmetric-key AES, rely on what is known as the computational hardness assumption, which asserts that the mathematical problem behind the cipher is easy to perform one way, but extremely hard to perform backwards, unless some additional information is known about the problem: the key. While computers today may not be able to crack these ciphers in a feasible amount of time, estimations have shown that quantum computers may be able to brute force RSA in as little as 20 to 30 years (Moses, 2009), which means that newer and stronger cryptosystems are needed in order to keep information secure. While much research has been done on different cryptosystems, the use of irrational numbers in this field has been somewhat untouched. Although the use of (somewhat related) continued fractions in cryptography has been proposed (Kane, 2013), not much research has been done regarding the use of the actual binary expansions of irrational numbers as a cryptographic key (which could then be combined with the plaintext/ciphertext using XOR to generate a ciphertext). Thus, the purpose of this study is to provide implementation techniques and basic cryptanalysis/statistical testing for key-generation based off of algebraic irrational numbers.

Symmetric and Asymmetric Key Ciphers

There are two main types of ciphers today: symmetric-key ciphers, also known as private-key ciphers; and asymmetric-key ciphers, also known as public-key ciphers. Each has its own advantages and drawbacks. Symmetric-key ciphers require that both the sending and receiving parties have knowledge of the same or similar key. Any ciphertext (encrypted text) encrypted with a symmetric-key cipher can be decrypted through the same cipher. Thus, to keep third parties from viewing the plaintext (unencrypted text), this key must be kept a secret (Agrawal & Mishra, 2012)[1]. On the other hand, asymmetric-key ciphers involve a private key and a public key. The public key is used to encrypt messages, and the private key is used to decrypt them. Many modern banks utilize a type of asymmetric-key cipher known as RSA in their online banking features. A user's sensitive personal information is first encrypted through a public key, which is available to everyone. Then, the bank uses its private key, which only the bank knows, to decrypt the sensitive information. Unless the private key can be found by a third party, the data is safe between the user and the bank (Agrawal & Mishra, 2012)[2].

One disadvantage to symmetric-key cryptography is that both users must have the same key. Oftentimes, the sender and the receiver are physically far from each other. The key in its unencrypted form cannot be sent via the Internet because third parties would be able to see the key and decrypt subsequent messages. The key also cannot be encrypted using a symmetric-key system because that symmetric-key system would need its own key. However, when using a secure algorithm, it is more secure, and even faster, than an asymmetric-key cipher (Lander, n.d.). Also, it is more than enough for a user to encrypt his/her own sensitive data on a personal hard drive. Because the user only needs to memorize one key, it is also the more convenient solution to encrypting the user's own data.

An asymmetric-key cipher also has its own pros and cons. A bank would most likely use an asymmetric-key cipher to encrypt data sent from online bankers because there are a multitude of users sending information to the bank. Since the private decrypting key is never released, the bank can send everyone their public encrypting key to encrypt their messages and sent it back (Kazmeyer, n.d.). To be clearer, the publicly released key can only encrypt messages, not decrypt them. That job is one that only the private key can do. If the bank used a symmetric-key cipher, it would need a different key for every single user who banked online, which would get extremely confusing and become a waste of server storage. As mentioned before, however, asymmetric-key ciphers are often slower than their symmetric counterparts, and are also less convenient for single person uses (due to the need of having a public and private key) in many cases involving personal protection of information.

Despite the fact that the proposed algebraic irrational number cipher is a symmetric-key cipher, this paper will frequently refer back to the asymmetric-key RSA to make various comparisons. Although comparing it to modern symmetric-key ciphers such as AES may seem more reasonable, AES is a multi-step matrix based cryptosystem, whose sheer complexity would make it unsuitable for meaningful and understandable direct comparisons. Although the actual RSA encryption scheme is also fairly convoluted, its strength is based only off of two very large prime numbers, which makes for easy strength comparisons.

Stream Ciphers

A stream cipher generates a long pseudo-random key – usually as long as the plaintext itself – and then employ a method to combine the key and the plaintext together to output a ciphertext. The strength of a stream cipher is determined mostly by the strength (randomness) of its method of key-generation. The Vigenère cipher is a well-known and rudimentary example of a stream cipher.

The Vigenère cipher, a symmetric-key system, essentially involves a separate Caesar cipher for every letter of a message. A Caesar cipher offsets each letter of a message by a specific amount, which is the key. While a Caesar cipher involves a key of a single integer, a Vigenère cipher a key of a multiple repeated integers. To encrypt a plaintext message with a key using the Vigenère method, one must first come up with a key that is equal or lesser in length than the message itself. Then, he/she will have to match each letter in the message to a letter in the key, repeating the key as the key runs out of letters. Most of the time, the key length will not be a multiple of the message length, and thus part of the key will be cut off at the end, which is not a problem. Next, each letter of both the message and the key will be translated into a number. Following computer science conventions, “a” will start as zero; “b” will become one, and so on. Now, each number of the message will be added to its corresponding key number. If the number is greater than or equal to 26, subtracting 26 will solve the problem. Thus adding one to a “z” will make it wrap around and become an “a” (Dalkilic & Gungor, 2000). To get a clearer understanding, a visual step-by-step example is provided below. In Figure 1, the message “computers” is being encrypted by the key, “bacon”. Keep in mind that a more secure key would probably not be an actual English word.

```
(1)  c  o  m  p  u  t  e  r  s  [the message]
(2)  b  a  c  o  n  b  a  c  o  [repeating the key]
(3)  2  14 12 15 20 19 4  17 18 [message converted to numbers]
(4)  1  0  2  14 13 1  0  2  14 [key converted to numbers]
(5)  3  14 14 29 33 20 4  19 32 [add message to key]
(6)  3  14 14 3  7  20 4  19 6  [mod 26]
(7)  d  o  o  d  h  u  e  s  g  [converting back to letters]
```

Figure 1. Diagram showing the steps of a Vigenère cipher encryption sequence using the message, “computers” and the key, “bacon”. Note that the repetition of the key poses as the Vigenère’s main weakness.

Thus, the encrypted message is “doodhuesg”. To decrypt it, one must have the same key, and essentially perform the above steps backwards. However, like the original Caesar cipher, this method of encryption has become weak due to modern day computers being able to brute force combinations. Furthermore, since the key is repeated over again, oftentimes a brute force method is not even required, as the usually short repeated key often creates patterns in a long text.

One-Time Pads

A one-time pad (OTP) is any type of cryptographic protocol where if carried out properly, is infinitely secure (Schumacher & Westmoreland, 2008). However, these are usually extremely tedious and are accompanied by huge keys. We can actually use the Vigenère cipher as a one-time pad if the key length is as long as the entire message length and the key is generated completely randomly. However, this would mean that a 10,000 letter long document would need a 10,000 letter long key to encrypt and decrypt it. Such keys are extremely inconvenient because they are near impossible to memorize and take a substantial amount of space to store. On top of that, it is extremely hard to share such a long key. Again, its only advantage is that it is extremely secure, which means that there is no way to send the key to someone else over the internet, because anyone with the key would be able to decrypt further messages; encrypting the key also would not solve the problem because there would be no point of using a super-secure one-time pad if the key is sent through the internet encrypted by a less secure means. Thus, the only plausible secure sharing solution is to have a key exchange in real life, which is often inconvenient or simply not possible. Lastly, a typical OTP key, as the name implies, can only be used once to ensure maximum security. The more times that the same key is used, the more information is leaked. Using a key many times could allow hackers to guess parts of the key.

This paper's proposed solution was to use the binary expansion of algebraic irrational numbers as a keystream in order to simulate the one-time pad security, while not needing to store the entire OTP key. As will be discussed later, the key of the irrational number cipher is also reusable, unlike the OTP whose security decreases dramatically if used more than once. Little to no research has been done on using this exact method in encryption and decryption. Therefore, the purpose of this investigation was to propose, provide implementation, and analyze the strength and time complexities of different methods in encryption and decrypting data that utilize the binary expansion of algebraic irrational numbers.

Mathematical Framework

Usage of Irrational Numbers

Irrational numbers are non-terminating numbers that cannot be written as a ratio of two integers. There are two types of irrational numbers: algebraic and transcendental. Algebraic numbers are numbers that are the solution of any polynomial equation, and transcendental numbers (such as π) are anything else ("Transcendental numbers", n.d.). This paper will be analyzing the use of only algebraic irrational numbers.

Normal Numbers

A number is said to be normal in base b if its expansion in that base has each digit appearing with a mean frequency leaning towards b^{-1}. In other words, the amount of each digit appearing in its decimal representation (to be normal in base 10) is the about the same over a long section of decimal places (Weisstein, n.d.). An example of a rational normal number is as follows:

$$\frac{123{,}456{,}789}{9{,}999{,}999{,}999} = 0.\overline{0123456789}$$

This number is obviously normal in base 10 because its repeating part has every digit once. This however, is not an example of an absolute normal number, which would have to be normal in any base $b \geq 2$.

Although no currently existing proof can confirm the statement that all algebraic irrational numbers are normal, an example of a non-normal algebraic irrational is yet to be found (Bailey & Crandall, 2002). Thus, it is conjectured that algebraic irrationals, such as $\sqrt{2}$. Numbers like π and e have also been conjectured to be normal.

Methodology

Procedure

An algorithm was written in the programming language, Java, to automate the processes of this cipher. One function was written to take in the plaintext with the keystream, which would then generate the ciphertext using a combination technique (presumably performing the XOR binary operation on the plaintext and the keystream). A second function was written to take in the ciphertext with the key, which would in turn output the plaintext, using the same strategy.

Simplified Algorithm Explanation

For the encryption process, the binary expansion was first generated from a given seed and a formula which will be discussed later in the paper. Next, the keystream was generated from certain specific numbers of the expansion. After the key was generated, it was combined with the plaintext to form the ciphertext. For the decryption process, the encryption process was reversed. The ciphertext was combined with the generated keystream in order to output the plaintext.

Different Methods of Key-Generation

First, the binary or decimal expansion is generated from the formula in Equation 1:

$$b = \sqrt[q]{p} \qquad (1)$$

Note that b is the binary expansion and p and q are both prime numbers. This guarantees that b is an irrational number. The simplest way now to generate the key from the binary expansion would be to take the initial bits/digits of the expansion. Assuming that *l* is the length of the plaintext, we would simply take the first *l* digits of the expansion and combine it with the plaintext to generate the ciphertext.

To increase obscurity, instead of just using the initial bits, the key could potentially consist of an initial starting index and a skip index. Now, the key would be generated by taking into consideration a position to start in the expansion, and skip every few digits (both numbers specified in the key).

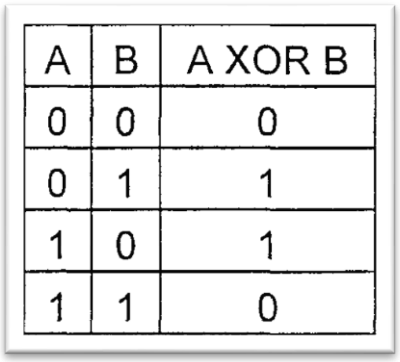
Unlike the aforementioned OTP, this method allows the seed and key to be reused. An OTP key cannot be reused because it gives the attacker crucial information about the key. If the attack has two different ciphertexts that are known to have been encrypted with the same OTP key, patterns can be found leading to the strength of the OTP diminishing immensely. Furthermore, if the attacker has a copy of a plaintext/ciphertext pair, the key is exposed, which allows easy decryption of further messages sent with the same key. However, with the proposed method, the same seed can be used multiple times without giving away too much information. For example, if person A and B both have the seed, after A sends a message to B, B can simply send a message back with the same seed, except with a different initial bit agreed upon by both people – perhaps just starting from the next unused bit.

Even if the attacker has a copy of a plaintext/ciphertext pair, only that part of the key is exposed – the seed is not exposed. Thus, when that attacker gets access to a new ciphertext, he/she is unable to use the key obtained previously to decrypt it. The attacker would only be able to decipher a ciphertext if the *seed* is known. Because there exist an infinite amount of algebraic irrational numbers and there are an infinite amount of these numbers that contain any specific key somehow lodged into them (keeping in mind that the key could be derived from an arbitrary initial bit and "skip" number), it would be extremely difficult for an attacker to guess the correct seed from just a part of the key. In other words, the attacker would have to, given some digits that appear in an irrational number, get back the closed-form formula (essentially the seed) in order to crack the next message sent if it was encrypted through the same seed.

Different Methods of Key Combination

Once the key is generated, there also exist multiple ways to combine the generated key and the plaintext to output the ciphertext, each with varying efficiencies and usages. The basic way of combination would mimic that of the Vigenère cipher, except instead of using a repeated key, we just use one long non-repeated keystream. This key will be the base-26 representation of the previously generated key. Assigning each letter of the message to a digit and offsetting each letter by that amount, we can generate the plaintext. One disadvantage to this method is that it can only encrypt and decrypt textual messages comprised of letters of the alphabet, and to handle more symbols, higher base representations of the irrational number will be needed.

A more advanced key-combination technique would employ the XOR logical operator, which is used in many advanced crypto-algorithms today. Figure 1 shows the XOR truth table. A and A XOR B can be seen as either the ciphertext or plaintext, and B is the key.



A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Figure 2. Truth table of the XOR logical operator. Taken from http://patentimages.storage.googleapis.com/. Notice that XOR is its own inverse, meaning that (A XOR B) XOR B is logically equivalent to A. In other words, if A is the plaintext in binary, B is the key, and A XOR B is the ciphertext, then XORing the ciphertext with the key will reproduce the plaintext.

Since XOR is its own inverse, we can XOR each binary bit plaintext and binary bits of the key to get the ciphertext, and XOR the ciphertext and keystream to get back the plaintext. The advantage of this is that it works with information stored as binary, which is very flexible. The XOR combination method will allow literally anything stored on a computer to be encrypted and decrypted.

By integrating this with charsets such as Extended ASCII or UTF-32 (USC-4) will allow for the encryption of over 400 million different characters. UTF-32 is a commonly used charset that maps individual characters to binary numbers. The number 32 in UTF-32 indicates that each of its characters will be represented by exactly 32 bits, meaning it can hold 2^{32} (roughly 400 million) pieces of information, or in this case, characters. Other variants include the charsets UTF-16 and UTF-8, which will allow for the encryption of fewer characters, but will save space on a hard drive. After using the UTF-32 mapping system to convert characters into binary, the result could then be XORed with the key in order to generate a ciphertext. A flowchart of the encryption process of a plaintext message into binary ciphertext using the algebraic irrational cipher is shown below in Figure 3.

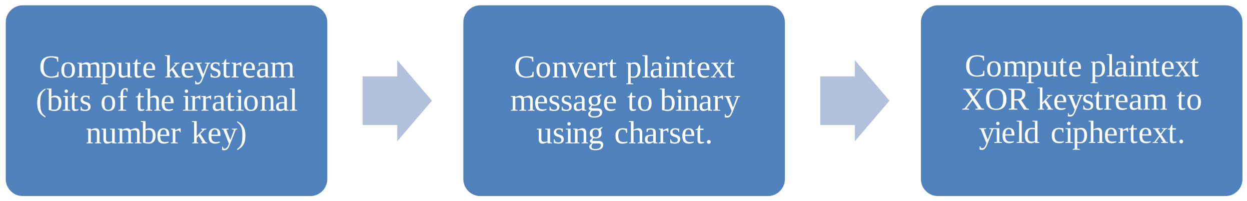

Compute keystream (bits of the irrational number key)
Convert plaintext message to binary using charset.
Compute plaintext XOR keystream to yield ciphertext.

Figure 3. Flowchart of proposed algebraic irrational number cipher encryption process. Note that the decryption process is very similar; instead of XORing the plaintext and key to yield the ciphertext, the ciphertext must be XORed with the key to yield back the plaintext.

Analysis and Discussion

Efficiency Analysis

The Newton-Raphson method (UBC Mathematics Department, n.d.) is an efficient iterative algorithm used to find successfully better approximations of the roots of a function, which is why it was utilized in the author's implementation of the irrational number cipher. Equation 2 shows the iterative step of the Newton-Raphson method:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \qquad (2)$$

Where: x_{n+1} is the next estimate of the root of function f,

x_n is the current estimate, and

f' is the first derivative of function f.

It is important to note that there is a possibility that using the Newton-Raphson method will not converge to a root (or converge to a wrong root) if the initial estimate is too far away from the true root. However this should not be an issue if Java's (or any high level programming language's) native power/root function is used to calculate the initial estimate. This being said, assuming that this method works (which it most likely will), the correct number of decimal places roughly doubles with each successful iteration. Thus, the time complexity $T(n)$ for calculating a root of function of f correct to n decimal places is shown in Equation 3:

$$T(n) = O(\log(n) * F(n)) \qquad (3)$$

Where: $F(n)$ denotes the cost of calculating $\frac{f(x_n)}{f'(x_n)}$ with n-digit precision.

Cryptanalysis

The entire algorithm's security is based upon the assumption that it is extremely difficult to identify the key from its decimal/binary expansion (keystream), even if he/she has access to old plaintext/ciphertext pairs (which implies access to the keystream as well). With access to the keystream, the attacker would need to identify which part of the irrational number the keystream came from (the initial starting index), and possibly how many digits were skipped in between each used digit (the skip index). This should be extremely hard because there exist an infinite amount of algebraic irrational numbers, and their decimal expansions are conjectured to contain every possible combination of integers. This also alludes to the fact that using common algebraic irrational numbers such as $\sqrt{2}$ and common non-algebraic irrationals such as e and π would create huge weaknesses, as the decimal expansions to these numbers have already been calculated to millions of digits and are readily available to the public domain.

Statistical Tests

Tests from the NIST (National Institute of Science and Technology) Test Suite were conducted on four pseudo-randomly selected keys of varying length (5 to 100). Each key, which was essentially an algebraic irrational number, was then calculated to over a million bits, and then was put through 15 different statistical tests. The yielded results are shown in Table 1. Note that the alpha value (α) was chosen to be 0.01 for all tests conducted, meaning all p-values *greater* than 0.01 were considered passing (sufficiently random).

Table 1. Results from over one million bits of each of four randomly selected keys, each of which was put through 15 different statistical tests from the NIST test suite. All p-values > 0.01 were considered passing (sufficiently random). TTsts that resulted in multiple p-values were displayed as a percent pass rate in the chart. Also, due to certain limitations, some tests were unable to run successfully, and thus are displayed as N/A. Since all four streams had 100% or near 100% pass rates, the bits of the irrational number expansion were considered sufficiently random.

Test Name	Results			
	Stream 1	Stream 2	Stream 3	Stream 4
Frequency Test	p = 0.905789	p = 0.088261	p = 0.650975	p = 0.393506
Freq. Test Within a Block	p = 0.369436	p = 0.200118	p = 0.776591	p = 0.197926
Runs Test	p = 0.139549	p = 0.727626	p = 0.172836	p = 0.689028
Longest Run of Ones Test	p = 0.693090	p = 0.690371	p = 0.164593	p = 0.638967
Binary Matrix Rank Test	p = 0.697531	p = 0.556623	p = 0.436289	p = 0.987842
Discrete Fourier Transform (Spectral) Test	p = 0.012006	p = 0.771215	p = 0.952507	p = 0.310464
Non-overlapping Template Matching Test	97.97% Pass	100% Pass	99.32% Pass	98.65% Pass
Overlapping Template Matching Test	p = 0.604300	p = 0.731979	p = 0.906156	p = 0.866049
Maurer's "Universal Statistical" Test	p = 0.735972	p = 0.574979	p = 0.832597	p = 0.103442
Linear Complexity Test	p = 0.052507	p = 0.717385	p = 0.743378	p = 0.544535
Approximate Entropy Test	p = 0.231035	p = 0.092834	p = 0.132923	p = 0.197926
Serial Test [1]	p = 0.331630	p = 0.191884	p = 0.443167	p = 0.484639
Serial Test [2]	p = 0.617122	p = 0.537435	p = 0.852684	p = 0.110276
Cumulative Sums Test (Forward)	p = 0.603010	p = 0.172637	p = 0.891918	p = 0.542761
Cumulative Sums Test (Backward)	p = 0.501403	p = 0.119366	p = 0.619408	p = 0.741157
Random Excursions Test	100% Pass	100% Pass	100% Pass	N/A
Random Excursions Variant Test	100% Pass	100% Pass	100% Pass	N/A

Test stream 2 completed all 15 tests with a 100% pass rate, while the other three test streams had near 100% pass rates. Thus, the computed bits from algebraic irrational numbers should be random enough to be used as keys in a cryptosystem.

Brute Force Attack Analysis

The brute force attack method involves breaking a cipher trying every single possible key combination. To analyze a 128-bit irrational number cipher's (the key is $\sqrt[p]{q}$ where p and q are both 128 bits in length) resistance towards a brute force attack, several assumptions must first be made: (1) each key test calculation takes 10 CPU operations and (2) a modern computer can do 10^{15} operations per second. Equation 4 shows that it will take 10^{78} total CPU operations, and equation 5 shows that it will take roughly 10^{54} centuries to crack the cipher.

$$2^{128} * 2^{128} * 10 = 10^{78}\ total\ CPU\ operations \tag{4}$$

$$\frac{10^{78}\ operations}{10^{15}\ \frac{operations}{second}} = 10^{63}\ seconds = 10^{54}\ centuries \tag{5}$$

Overall, the passing of the NIST statistical tests and the analysis of the brute force attack on the irrational number cipher indicate that this cipher is secure, although extended research may be required to truly confirm its security.

Limitations and Future Research

The nature of this study and its limitations allowed for many opportunities for future research. For example, a symmetric-key irrational number cipher was proposed and analyzed, but in the future one could investigate the usage of irrational numbers in public-key (asymmetric-key) cryptography, which may have more practical usages. Further cryptanalysis could also be done regarding various cipher-specific attacks that could possibly be done on the cipher aside from just the brute force method. Lastly, the overall study would benefit from direct and indirect comparisons to cryptosystems besides RSA.

Summary

This paper proposed and analyzed a symmetric-key cipher based on the binary expansion of algebraic irrational numbers. Statistical tests from the NIST testing suite concluded that the bits of the selected irrationals were sufficiently random for cryptography, and the brute force attack analysis indicated that such an attack would be impossible to carry out in a feasible amount of time.

Literature Cited

Agrawal, M., & Mishra, P. (2012). A comparative survey on symmetric key encryption techniques. *International Journal of Computer Science and Engineering*, *4*(5). Retrieved from http://www.enggjournals.com/ijcse/doc/IJCSE12-04-05-237.pdf

Bailey, D. H., & Crandall, R. E. (2002). Random generators and normal numbers. *Experiment. Math*, *11*(4), 527-546.

Dalkilic, M. E., & Gungor, C. (2000). An interactive cryptanalysis algorithm for the vigenere cipher. *Advances in Information Systems*, 341-351. Retrieved from http://www.researchgate.net/profile/Cengiz_Guengoer/publication/221581427_An_Interactive_Cryptanalysis_Algorithm_for_the_Vigenere_Cipher/links/00b49518bb5dec3260000000.pdf

Kane, A. M. (2013). On the use of continued fractions for stream ciphers. *Département de Mathématiques et de Statistiques, Université Laval*.

Kazmeyer, M. (n.d.). Advantages and disadvantages of symmetrical and asymmetrical encryption. Retrieved May 31, 2015, from http://science.opposingviews.com/advantages-disadvantages-symmetrical-asymmetrical-encryption-2143.html

Lander, S. (n.d.). Advantages & disadvantages of symmetric key encryption. Retrieved May 31, 2015, from http://science.opposingviews.com/advantages-disadvantages-symmetric-key-encryption-2609.html

Moses, T. (2009, January). *Quantum computing and technology: Their impact on cryptographic practice*. Retrieved March 26, 2016, from Entrust website: https://www.entrust.com/wp-content/uploads/2013/05/WP_QuantumCrypto_Jan09.pdf

The newton-raphson method [Lecture notes]. (n.d.). Retrieved February 8, 2016, from UBS Math Dept., p.d. website: https://www.math.ubc.ca/~anstee/math104/104newtonmethod.pdf

Schumacher, B., & Westmoreland, M. D. (2008, February). *Quantum mutual information and the one-time pad*. Retrieved from http://arxiv.org/pdf/quant-ph/0604207.pdf

Transcendental numbers. (n.d.). Retrieved June 1, 2015, from Math is Fun website: https://www.mathsisfun.com/numbers/transcendental-numbers.html

Weisstein, E. W. (n.d.). Normal number. Retrieved June 2, 2015, from Wolfram Math World website: http://mathworld.wolfram.com/NormalNumber.html