Issues in Information Systems
Volume 17, Issue I, pp. 14-25, 2016

A CIPHER BASED ON THE RANDOM SEQUENCE OF DIGITS IN IRRATIONAL NUMBERS

J. L. González-Santander, martinez.gonzalez@ucv.es, Universidad Católica de Valencia "san Vicente mártir"
G. Martín González. german.martin@ucv.es, Universidad Católica de Valencia "san Vicente mártir"

ABSTRACT

An encryption method combining a transposition cipher with one-time pad cipher is proposed. The transposition cipher prevents the malleability of the messages and the randomness of one-time pad cipher is based on the normality of "almost" all irrational numbers. Further, authentication and perfect forward secrecy are implemented. This method is quite suitable for communication within groups of people who know one each other in advance, such as mobile chat groups.

Keywords: One-time Pad Cipher, Transposition Ciphers, Chat Mobile Groups Privacy, Forward Secrecy

INTRODUCTION

In cryptography, a cipher is a procedure for encoding and decoding a message in such a way that only authorized parties can write and read information about the message. Generally speaking, there are two main different cipher methods, transposition, and substitution ciphers, both methods being known from Antiquity. For instance, Caesar cipher consists in substitute each letter of the plaintext some fixed number of positions further down the alphabet. The name of this cipher came from Julius Caesar because he used this method taking a shift of three to communicate to his generals (Suetonius, c. 69-122 AD). In ancient Sparta, the transposition cipher entailed the use of a simple device, the scytale (skytálē) to encrypt and decrypt messages (Plutarch, c. 46-120 AD), which consisted of two equal wood rods and a ribbon rolled up on one of them. When you write down the message along the rod over the ribbon, one letter in each turn of the ribbon, and then you unroll the ribbon, the letters of the message along the ribbon were transposed. The recipient of the message could decipher it rolling up the ribbon over its twin rod. Systems combining substitution and transposition were used first in World War I by the Germans in the so-called ADFGX and ADFGVX ciphers. Invented by Colonel Fritz Nebel, these ciphers combined the substitution method of the Polybius square with a single columnar transposition method. The Germans believed the ADFGVX cipher was unbreakable, but Lieutenant Georges Painvin, a bright member of the *Bureau du Chiffre* of the French Army, broke this cipher in June 1918 (Childs, 2002). Nowadays, the combination of substitution and transposition methods is standard for block ciphers.

The aim of this paper is to combine also both methods, transposition, and substitution, by using the random sequence of digits in irrational numbers. This method offers a simple way to provide reasonable security in the communication within groups of people. This kind of moderate security would be applicable, for instance, in the nowadays chats through mobile applications.

This paper is organized as follows. Section 2 describes a brief history of the substitution method used in this paper, namely the one-time pad cipher, and how it is implemented here. According to the discussion of this method, it will be proposed a simple idea to overcome the main drawback of it, that is, the randomness of the key. Moreover, in order to provide more security, a very simple one-way function is described for updating the key in each message. Despite the fact that one-time pad cipher is absolute safe under certain conditions, the message could be intentionally changed by a third part without knowing the key. In order to avoid this *malleability* of the message, Section 3 shows how to combine the one-time pad cipher with a simple transposition cipher. Section 4 is devoted to second order encryption methods, that is to say, the ciphering of the keys and the authentication of the message. In Section 5 the operation steps for the proposed cryptosystem are collected. Finally, the conclusions are summarized in Section 6.

ONE-TIME PAD CIPHER

History, Advantages, and Drawbacks

In 1917, a cryptosystem for telegraph was patented by Gilbert Vernam (Vernam, 1926). However, recently has been found that Vernam's cipher was discovered first by Frank Miller in 1882 (Bellovin, 2011). In this cryptosystem, each character of the message is combined with a key in order to output a coded message (see below for details). Nonetheless, since the key could be reused in the original Vernam's system, the encrypted message could be deciphered. However, later on, Joseph Mauborgne realized that the cryptanalysis would be very difficult if the key was random. This new version of Vernam's system is usually termed as one-time pad cryptosystem. In 1949, Claude Shannon (Shannon, 1949), by using Information Theory, proved that a one-pad encrypted message was unbreakable in the sense that the latter does not give any information about the plaintext, that is to say, given the ciphertext, any plaintext is equally likely to have been encrypted. Despite the fact that the one-time pad cipher is the only one that has a mathematical proof about its absolute security, it has the following drawbacks:

1. The key must be perfectly random.
2. The exchange of keys must be safe.
3. We need to generate as many different keys as pairs of users in communication.
4. The key must be at least as long as the message.

In order to avoid drawback 1, there are some well-known *Cryptographically Secure Pseudorandom Number Generators*, such as the new version of the simply RC4 method (Rivest and Schuldt, 2014) or the more sophisticated block cipher AES (Daemen and Rijmen, 2000). These generators are called "pseudorandom" because they produce a list of numbers by a deterministic procedure, but they simulate random sequences in the sense that statistical tests cannot distinguish between the output of the generator and the uniform distribution. If we do not trust enough pseudorandom numbers generated algorithmically, Linux users can use the random generator of the kernel /dev/random, which uses the environmental noise to generate random data (Gutterman, Pinkas, and Reinman, 2006).
To tackle drawback 2, Whitfield Diffie and Martin E. Hellman (Diffie and Hellman, 1976) invented a safe system to exchange information between two people without sharing any secret key. This method is based on the Discrete Logarithm Problem (DLP). Based also in DLP, ElGamal (ElGamal, 1984) proposed a similar method.
Another safe approach to share information is based on Quantum Mechanics. The first protocol of Quantum Cryptography used the transmission of polarized photons and it was devised in 1984 by Charles Bennett and Gilles Brassard (Bennett and Brassard, 1984). Artur Ekert developed a different approach to quantum key distribution based on peculiar quantum correlations known as quantum entanglement (Ekert, 1991). Nonetheless, despite the fact that Quantum Cryptography is completely safe, the practical usage of it in conventional communications is far to be generalizable.

Regarding drawback 3, if we have a group of N people, the number of different keys needed for all of them to be communicated would be $N(N-1)/2$. This would be cumbersome if the group is large. That is why nowadays the cryptosystems based on public encryption keys are preferred, such as RSA (Rivest, Shamir, and Adleman, 1978). In these public cryptosystems, the decryption key is kept secret, so that there is an asymmetry between encryption and decryption keys. In RSA, this asymmetry is based on the practical difficulty of factoring the product of two large prime numbers. However, RSA is computationally time-consuming, so that some protocols, such as PGP (Zimmermann, 1995), prefers to use RSA only to encrypt the keys of the classic symmetrical coding, being the latter much faster.

Drawback 4 is telling us that we have to be able to generate a random key potentially infinitely long. This feature is quite critical for pseudorandom number generators. Moreover, even for Linux users, when there is no more environmental noise available, the kernel needs to generate pseudo-random numbers by using /dev/urandom.

Random Digits in Mathematical Constants

Taking into account the above comments about the drawbacks of one-time pad cryptosystem, the scope of this paper is just to provide a very simple method for exchanging messages in a group of people with a reasonable security, although not absolute.
Regarding drawback 1, we propose the generation of the key from the binary digit expansion of mathematical fundamental constants and its combinations. By fundamental constants, we understand π, e, the golden ratio ϕ, log 2, sinh(1), the Euler's constant γ, the Catalan's constant G, etc.

In order that these mathematical constants and its combinations can generate random sequences, its normality is required. However, normality does not assure randomness, since there are normal sequences of digits which are predictable, such as Champernowne's constant (Champernowne, 1933). In any case, a real number is normal when its infinite sequence of digits distributes uniformly in any base b. This means that each of the b digit values has the same probability $1/b$ to be found in the infinite sequence of digits; also all possible b^2 pairs of digits are equally likely with probability $1/b^2$; all b^3 triplets of digits have the same probability $1/b^3$, and so on. This means that any combination of digits occurs with the same frequency as any other. This concept of normal numbers was introduced first by Emile Borel in 1909. He proved that the Lebesgue measure of the non-normal numbers is zero (Borel, 1909), thus, we can say that "almost all" real numbers are normal. Despite Borel's theorem on normal numbers, the constants that are known to be normal are given by artificial constructions, such as Champernowne's constant (Champernowne, 1933) or Copeland-Erdös' constant (Copeland and Erdös, 1946). Although there is no proof up to the present about the normality of the fundamental mathematical constants, such as π, e, log 2, $\sqrt{2}$, is highly suspected that every irrational algebraic number is absolutely normal, since no counter-examples are known. However, as aforementioned, randomness is not enough to assure security in Cryptography, since unpredictable keys are also needed, which is not the case of well-known constants such as π, e, log 2, $\sqrt{2}$ … Nonetheless, the different combinations of mathematical constants in a determined formula are as large as the imagination of the users. For instance, think in

$$\frac{e^{\sqrt{\phi}}}{\pi}+\log 3,\ \frac{\sinh^{-1}1}{\tan\sqrt{2}},\ \sqrt[3]{\log\left(1+\pi^3\right)},\ldots \tag{1}$$

Also, these mathematical expressions provide an infinitely long sequence of random numbers (namely 0's and 1's in base $b = 2$) in a very compact way. Note that if the users can accord orally, face to face, a particular formula of mathematical constants as the key of their own messages, they can use one-time pad cipher to communicate themselves with a reasonable security, since the number of possible keys is as big as the users' imagination and there is no systematical way to test all of them. This hypothesis is quite reasonable if we consider a mathematical formula as a string of symbols: arithmetic symbols and parenthesis (+, −, /, *, ^, (,)), well-known mathematical constants (i, π, e, γ, G...), the exponential function and its inverse (exp, log), trigonometric functions and its inverses (sin, cos, tan, $\sin^{-1}$, $\cos^{-1}$, $\tan^{-1}$), hyperbolic functions and its inverses (sinh, cosh, tanh, $\sinh^{-1}$, $\cosh^{-1}$, $\tanh^{-1}$), the modifying functions (real part, imaginary part, ceiling, floor, round…) and natural numbers up to 100. If these symbols (around 130) are combined in a string of, say, 20 symbols, we will obtain 130^{20} different strings! Despite the fact that most of these strings are mathematically meaningless, according to authors' knowledge, there is no systematical way of building meaningful strings in an exhaustive way.

Nonetheless, if the users cannot communicate themselves face to face, another way to exchange the key is to use any standard method of a public key, such as the ones provided by Diffie-Hellmann, ElGamal or RSA.

Regarding to the computation of the digits in the formula of the mathematical constant, the BBP-method allows to compute very rapidly the n-th digit (in a particular base) of any mathematical constant given by an infinite series of a certain type without needing to compute any of the first $n - 1$ digits (Bailey, Borwein and Plouffe, 1997). However, the list provided by Bailey (Bailey, 2000) for such constants are limited and public, so it would not be very useful from a cryptographic point of view. Nonetheless, by using a commercial PC, the authors have tested that

MATHEMATICA's command Real Digits provides the millionth binary digit of the constants given in (1) (and many others) in less time than 2.5 s. Nevertheless, there is another way out to overcome the difficulty of many and very large messages, consisting in updating the key in each message, as it will be explained later on.
In summary, regarding the drawbacks given above, this paper tries to overcome them in the following way:

1. We may expect that the key is random and quite difficult to guess by brute force.
2. It is quite easy to exchange the keys in a very compact way, face to face. This is suitable for mobile groups of friends or workers in a company.
3. Every pair of persons in communication has to choose their keys.
4. We can provide efficient methods to compute as many digits as required in practice for exchanging messages. Alternatively, we can update the key in each message.

Implementation of the One-Time Pad Cipher

One of the most common ways to cipher a plaintext with the one-time pad is to use the XOR cipher (Churchhouse, 2002, p. 11), also known as *binary arithmetic* (Paar and Pelzl, 2009, p. 29). This cipher uses the XOR logic operator, whose truth table is given in Table 1.

Table 1. Truth Table of the XOR(x, y) Coding Function

XOR (x, y)		x	
		0	1
y	0	0	1
	1	1	0

The XOR operator is simple to implement and computationally cheap. Mathematically, it reads as:

$$\mathrm{XOR}(x, y) = (x + y) \bmod 2 .$$

The XOR cipher follows basically four steps:

1. Conversion of the plaintext into a vector of numbers $\vec{r}$ each of them being the corresponding ASCII code number (see Table 2).
2. Conversion of $\vec{r}$ into a binary matrix *R*, being each row of *R* the binary expression of each ASCII code number (Table 2).
3. Conversion of each matrix element of *R* to a matrix *S* according to the binary expression of the key (Table 4).
4. Conversion of *S* into a sequence of numbers $\vec{s}$ being each one the decimal expression of each row of *S* (Table 3).

In the example given in the Tables of this Section, we have used as key the following irrational number:

$$k_{\mathrm{XOR}} = \sqrt{\pi} \approx 1.772453850... \tag{2}$$

Therefore, the sequence of binary digits of the above key to the left of the decimal point (our keystream) b_j is

$$b_j = 1, 1, 1, 0, 0, 0, 1, 0, ... \quad j \in \bullet .$$

Table 2. Plain Text Conversion into a Binary Matrix

Plain text		ASCII $\vec{r}$		Binary matrix R
H	→	72	→	0, 1, 0, 0, 1, 0, 0 0
e	→	101	→	0, 1, 1, 0, 0, 1, 0, 1
l	→	108	→	0, 1, 1, 0, 1, 1, 0, 0
l	→	108	→	0, 1, 1, 0, 1, 1, 0, 0
o	→	33	→	0, 1, 1, 0, 1, 1, 1, 1

Table 3. Codification of the Matrix and Associated ASCII Vector

Binary matrix R		Coded matrix S		ASCII $\vec{s}$
0, 1, 0, 0, 1, 0, 0, 0	→	0, 1, 0, 1, 0, 1, 0, 1	→	85
0, 1, 1, 0, 0, 1, 0, 1	→	0, 1, 0, 0, 0, 1, 0, 1	→	69
0, 1, 1, 0, 1, 1, 0, 0	→	0, 1, 0, 1, 0, 1, 1, 1	→	87
0, 1, 1, 0, 1, 1, 0, 0	→	0, 0, 0, 1, 1, 1, 1, 0	→	30
0, 1, 1, 0, 1, 1, 1, 1	→	0, 0, 1, 1, 0, 1, 1, 1	→	55

The matrix element of the coded matrix S is given by

$$s_{ij} = \mathrm{XOR}(m_{ij}, b_{i+j}). \tag{3}$$

Table 4. Flux Diagram for a Single Character Coding/Decoding

		Binary digits							
Plain H	↔	0	1	0	0	1	0	0	0
		b	b	b	b	b	b	b	b
Key $\sqrt{\pi}$	↔	1	1	1	0	0	0	1	0
		b	b	b	b	b	b	b	b
Coded U	↔	0	1	0	1	0	1	0	1

According to the truth table of the function $\mathrm{XOR}^2(x, y)$ (see Table 5) is clear that

$$\mathrm{XOR}^2(x, y) = \mathrm{XOR}\left(\mathrm{XOR}(x, y), y\right) = x.$$

Therefore, from the coded binary digits, we can recover the plain binary digits by using the same key and the same coding function $\mathrm{XOR}(x, y)$, that is to say, the coding is symmetric.

Table 5. Truth Table of the $XOR^2(x, y)$ Decoding Function

$XOR^2(x, y)$		x	
		0	1
y	0	0	1
	1	0	1

Updating the Keys

As aforementioned, the great advantage of the present method is that the random sequence of the key shared between sender and recipient is condensed in an irrational number. However, in every message sent or received, we have to keep track of the total number of characters exchanged in order to know in which digit of the keystream we have to continue the coding/decoding of the message. Therefore, if the number of messages was quite high, we would have to compute thousands of decimals each time. A way out to avoid the latter is to update the key each time we send/receive a message. For this purpose, note that in every coded message transmitted, there are two variables that sender and recipient share: the number messages exchanged m and the number of characters of each message n_m. Therefore, a possible way to update the key would be the following:

$$k_{\mathrm{XOR},m} = k_{\mathrm{XOR},0}\ q_m,\ \ m \in \bullet\ ,$$

being $k_{\mathrm{XOR},0}$ the initial key and q_m a coefficient that takes into account all the previous history between sender and recipient (i.e. m and n_m). This coefficient q_m can be calculated as follows:

$$q_{m+1} = F^k_{n,n_m}(q_m), \tag{4}$$

$$F^k_{n,n_m}(x) = \mathrm{frac}(n_m \sin(m\,x) + m\cos(n_m x)), \tag{5}$$

$$q_0 = \gamma,$$

where frac(x) denotes the fractional part of x (Oldham, Myland and Spanier, 2010, p. 69), and $F^k(x)$ denotes the $F(x)$ function iterated k times, that is to say, if k is a non-negative integer, then

$$F^0(x) = 1,$$
$$F^{k+1}(x) = F^k(F(x)).$$

Notice that (5) is not analytically invertible. Moreover, if k is large enough, namely $k \geq 10$, this way to calculate q_{m+1} assures that (4) is almost impossible to be inverted numerically (i.e. to compute q_m), so it acts like a one-way function (Goldreich, 2001, p. 31).

MALLEABILITY

Yet another issue has to be taken into account, namely the message authentication, which is not usually implemented in one-time pads. This lack exposes the messages to be changed by an attacker, even though the key remains unknown to him. For example, an attacker who knows that the message contains *Transfer 0000100.00$* to *account #221* at a particular point, can replace it by any other content of exactly the same length, such as *Transfer 0100000.00$* to *account #327*. This property is common to all stream ciphers and is known as malleability (Hanaoka, 2008). Also, the XOR operator is vulnerable to a known-plaintext attack, since XOR(plaintext, ciphertext) = key.

In order to avoid both types of attacks, we can transpose the message before using the one-time pad cipher. The following Section describes the most common transposition ciphers and the one proposed in this paper.

Transposition Algorithm

A transposition cipher is essentially a permutation of the characters (or group of characters) of the plaintext. The most popular transposition ciphers are Rail Fence Cipher, Route Cipher and Columnar Transposition Cipher (Smith, 1955, p. 29). Since transposition does not affect the frequency of individual characters, simple transposition can be easily detected by performing a frequency analysis.

Rail Fence Cipher consists of writing downwards and diagonally on successive "rails" of an imaginary fence, then moving up when we reach the bottom rail. When we reach the top rail, the message is written downwards again until the whole plaintext is written out. The message is then read off in rows. This cipher can be attacked by brute force because the number of keys is very low (the number of rails).

In Route Cipher, the plaintext is first written out in a grid of given dimensions and then is read off in a pattern given in the key. This cipher has many more keys than Rail Fence, but it can be attacked by anagramming (Smith, 1955, p. 96).

In Columnar Transposition, the message is written out in rows of a fixed length, and then read out again column by column, being chosen the columns in some scrambled order. Both the width of the rows and the permutation of the columns are usually defined by a keyword. This cipher has the property that a key close to the correct one shows long sections of legible plaintext. Therefore, it can be attacked by optimum searching algorithms such as genetic type algorithms (Matthews, 1993).

The transposition algorithm described here uses the same conjecture as the substitution method presented in Section 2, namely, the randomness of the digits in mathematical constants. Therefore, as in the case of one-time pad cipher, we will use the digits of an irrational number k_{trans} as the key for permuting the characters of the plaintext. For the sake of security, the transposition key must be different from the one-pad key, i.e. $k_{\text{trans}} \neq k_{\text{XOR}}$. Here it follows an example of how the transposition of characters is performed. Let us consider the message given in Table 6, whose length is $n = 15$.

Table 6. Message Not Permuted

M	a	r	y	_	h	a	s	_	a	_	l	a	m	b

Taking now as transposition key, $k_{\text{trans}} = \sqrt{e} = 1.648721271...$, let us group its digits, being each group of length

$$l = 1 + \lfloor \log_{10} n \rfloor = 2.$$

so we have the following sequence

$$d_i = 16, 48, 72, 12, 70, 70, 01, 28, 14, 68\ldots$$

In order to have a random sequence of integers within 1 and n, we compute

$$e_i = 1 + d_i \bmod n = 2,4,13,13,11,11... \tag{6}$$

We set every character of the plaintext in the position given by (6). In Table 7 is shown the permutation for the first 3 elements of the sequence. However, the next element in sequence (6) (i.e. $e_4 = 13$), indicates a position which is already occupied, so we can jump to the next free position so that $e_4 = 14$. By doing so, we can fill in all the positions with the characters of the initial message, obtaining eventually the permutation given in Table 8.

Table 7. Initial Permutation of the Message

Old position		1		2								3		
New position		2		4								13		
Mew message		M		a								r		

Table 8. Final Permutation of the Message

_	M	a	a	_	l	m	a	a	b	_	h	r	i	s

In order to recover the original message, notice that if we depart from a message with the natural numbers up to n in order, and then we apply the above algorithm with the same key, we will obtain the positions of the characters in the new message (see Table 9).

Table 9. Permutation of Indices

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
9	1	7	2	11	12	14	13	10	15	5	6	3	4	8

Then, according to Tables 8 and 9, the first character in the permuted message, namely "_", was in ninth position in the original message; the second character "M" was in the first position; and so on. In this way, the original message can be reconstructed easily. Also, as in the case of k_{XOR}, the transposition key k_{trans} can be modified in each message as it has been explained in Section 2.

SECOND ORDER ENCRYPTION

So far, we have been dealing with the encryption of the message. However, for the operation of the ciphering method described above, on the one hand, the sender needs both a substitution as a transposition key, k_{XOR} and k_{trans}, and on the other hand, the recipient needs to identify the sender. This information must be encrypted as well, thus, a second order encryption is also needed.

Ciphering the Keys

Knowing how it works the encryption method described above, the effort of a potential attacker should be concentrated in finding out the keys, that is to say, the formulas of the mathematical constants that sender and recipient share. Since these keys must be recorded somewhere, a hacker could enter into our PC and simply copy them. A way to prevent this kind of attack is to cipher the keys as well. If we convert the mathematical formulas into strings, we can use again the one-time pad to cipher the latter. What key should use each user to encrypt his collection of shared keys? A simple way to answer is to use the own user's password to log into the cryptosystem. But, how can we convert the password string into an irrational number? In order to answer this, let us consider the vector $\vec{v}$ as the ASCII code numbers of the password characters, for example

password = HelloWorld,
$\vec{v}$ = (72, 101, 108, 108, 111, 87, 111, 114, 108, 100).

Then, the new key k_{pass} to encrypt the shared keys with other users could be built as

$$k_{\text{pass}} = \sum_{j=1}^{\dim \vec{v}} j \log_{j+\sqrt{2}} v_i \approx 128.2359446599...$$

In order to show how this works, let us consider the following example. If we have the following formula as the key shared between two people

$$k = \tan\sqrt{2}, \tag{7}$$

(being $k = k_{\text{XOR}}$ or k_{trans}) we can convert it into a string of characters

$$\text{Tan}\left(\text{Sqrt}\left(2\right)\right) \tag{8}$$

Now, applying the XOR cipher to (8) as explained above, we will obtain a vector as the encrypted formula

$$(74, 82, 1, 24, 72, 131, 160, 18, 177, 20, 73, 112). \tag{9}$$

An interesting feature of this encoding is that an infinite set of encrypted keys corresponds in fact to the same key. For instance, the following key is equivalent to (7),

$$0 + \tan\sqrt{2/1},$$

but its encryption is completely different to (9), namely

$$(46, 19, 68, 99, 79, 147, 188, 76, 185, 87, 102, 89, 47, 115, 32, 27, 151).$$

On the one hand, to rely the security of the system described above on user's password is quite unsafe because of the keystroke logging attack, such as Magic Lantern (Sullivan, 2001). Nevertheless, screen keyboards for touch pad devices can overcome this type of attack.

On the other hand, forward secrecy would be required to ensure that past messages cannot be recovered retroactively from a third party, as in Off-the-Record Communication (Borisov, Goldberg, and Brewer, 2004). Nonetheless, since the actual keys for ciphering the messages are continuously changing by using (4), and this updating is not invertible, then, though the attacker got to know the password (and thus the keys), he could not decrypt past messages. In any case, for the sake of security, read messages must be deleted from the recipient's system.

Authentication of the Message

When someone receives a message from another member of the group, the identification of the sender is needed in order to apply the correct key shared between both. Further, the recipient wants to ensure that there is not a third party superseding the identity of someone else in the group. Obviously, if the message is private, it would be desirable that sender's name was also encrypted. This can be done by using the same encryption as the message, that is to say, the transposition method described in Section 3, followed by the one-time pad cipher described in Section 2. Since the number of members of the group is usually small, the recipient may attempt to decode the name of the sender, testing one by one the keys that he or she knows belonging to each member of that group. If none of the decoded names is on the recipient's list of names, then the message is not authenticated. Remember that the keys shared between sender and recipient are being updated in each message, preventing a brute force attack from a potential hacker.

Protocol

In order to set the ideas given in the previous Sections, here we provide the steps for the operation of this encryption method, following an example.

1. Create a crypto-chat group, for instance: Friends.
2. Create the nodes of this group, for instance: Alice, Bob, and Charles.
3. Each pair of nodes in communication has to share two irrationals numbers acting as k_{XOR} and k_{trans}.
4. Each node has to encrypt its shared keys with other nodes according to its own password and then record them into a table. For instance, imagine that Alice shares keys with Bob and Charles and her password is "Wonderland", then a possible configuration is given in Table 10. Notice that even the characters π and e are repeated, in the encryption of the keys there is no track of this repetition.

Table 10. Alice's Table (Original and Encrypted)

Name	$k_{XOR,0}$	$k_{trans,0}$	Alice's	$k_{XOR,0}$	$k_{trans,0}$
Bob	π^2	e^2	↔	(155, 55, 21, 227)	(174, 85, 20)
Charles	π/e	e/π	password	(124, 69, 93, 15)	(33, 31, 231, 94)

Once the keys are set, we proceed to send a message according to the following steps:

1. Selection of crypto-group and recipient: Alice wants to send a message to Bob within the group Friends.
2. Alice decrypts her recorded encrypted keys shared with Bob with her password. According to Table 11, Alice gets $k_{XOR,0} = \pi^2$ and $k_{trans,0} = e^2$.
3. Alice encodes her name and her message to Bob with the decrypted keys $k_{XOR,0}$ and $k_{trans,0}$.
4. Alice calculates q_1 with (4) for updating keys and then she encrypts and records $k_{XOR,1}$, $k_{trans,1}$ and q_1.
5. Alice sends the encrypted message to Bob, putting as heading her name encrypted.
6. Alice erases the message sent to Bob.

When Bob receives Alice's message, he has to perform the following steps for decryption:

1. Authentication of the message by the correct decoding of the heading, i.e. the name of Alice.
2. With the same keys used for the decryption of the heading, Bob decodes the message.
3. Bob updates and records the keys shared with Alice as she did when she sent the message.
4. Bob erases the message received from Alice.

Notice that the cryptosystem must not allow to any node send messages until its inbox is empty, because both, sender and recipient, must have their keys updated. Nonetheless, if Alice and Bob send a message one each other almost simultaneously, then the key updating will not be synchronized and the decoding will not work. However, the higher is the speed of message transmission, the more unlikely this will happen.

CONCLUSIONS

A simple cryptographic method is proposed, being very suitable for communication within groups of people who know one each other in advance (family, friends, working peers...). It consists in combining a transposition cipher followed by a one-time pad cipher. The transposition cipher is used to avoid malleability and known plaintext attack. The security of the one-time pad cipher is based on the randomness of the digit expansion of irrational numbers, which are assumed to be normal. Also, the non-prediction of the key is assured by users in communication, who can imagine a formula that combines different mathematical constants quite difficult to guess. Therefore, it is assumed also that users have some knowledge about mathematical constants and elementary functions. If the users know one each other in advance, the distribution of the key has to be done face to face between them. If this is not the case,

they can still exchange the key by using any reliable standard public key. The transposition cipher explained in this paper performs a random permutation of the characters in the message taking into account again the randomness of the digits of an irrational number. For security, the key used for the one-time pad and for the transposition method must be different, thus, the users would have to accord two keys. Knowing these two keys, the message can be encrypted and decrypted very easily. Another feature of this cryptographic method is the updating of the keys in every sent message with a new one-way function, very easy to implement, providing perfect forward secrecy. Finally, in order to prevent hackers attacks, a method for the encryption of the keys and the authentication of the messages are also proposed.

REFERENCES

Bailey, D. H. (2000). A compendium of BBP-type formulas for mathematical constants. Preprint, http://crd. lbl. gov/dhbailey/dhbpapers/index.html

Bailey, D., Borwein, P., & Plouffe, S. (1997). On the rapid computation of various polylogarithmic constants. *Mathematics of Computation of the American Mathematical Society, 66*(218), 903-913.

Bailey, D. H., & Crandall, R. E. (2001). On the random character of fundamental constant expansions. *Experimental Mathematics, 10*(2), 175-190.

Bellovin, S. M. (2011). Frank Miller: Inventor of the one-time pad. *Cryptologia, 35*(3), 203-222.

Bennett, C. H., & Brassard, G. (1984). Quantum cryptography: Public key distribution and coin tossing. In I*nternational Conference on Computer System and Signal Processing, IEEE, 1984* (pp. 175-179).

Borel, E. (1909). Les probabilités dénombrables et leurs applications arithmétiques. *Rendiconti del Circolo Matematico di Palermo 27*(1), 247-271.

Borisov, N., Goldberg, I., & Brewer, E. (2004). Off-the-record communication, or, why not to use PGP. In *Proceedings of the 2004 ACM workshop on Privacy in the electronic society* (pp. 77-84). ACM.

Champernowne, D. G. (1933). The construction of decimals normal in the scale of ten. *Journal of the London Mathematical Society, 1*(4), 254-260.

Churchhouse, R. F. (2002). *Codes and ciphers: Julius Caesar, the Enigma, and the Internet*. Cambridge University Press.

Copeland, A. H., & Erdös, P. (1946). Note on normal numbers. *Bulletin of the American Mathematical Society, 52*(10), 857-860.

Daemen, J., & Rijmen, V. (1998). The block cipher Rijndael. In *Smart Card Research and Applications* (pp. 277-284). Springer Berlin Heidelberg.

Diffie, W., & Hellman, M. E. (1976). New directions in cryptography. *Information Theory, IEEE Transactions on, 22*(6), 644-654.

ElGamal, T. (1984). A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in cryptology* (pp. 10-18). Springer Berlin Heidelberg.

Ekert, A. K. (1991). Quantum cryptography based on Bell's theorem. *Physical review letters, 67*(6), 661-663.

Goldreich, O. (2001). *Foundations of Cryptography. Vol. 1. Basic Tools*. Cambridge University Press.

Gutterman Z., Pinkas, B., &, Reinman, T. (2006, May). Analysis of the Linux random number generator. In *Security and Privacy*, (pp. 371-385). IEEE.

Hanaoka, G. (2008). Some information theoretic arguments for encryption: Non-malleability and chosen-ciphertext security (invited talk). In *Information Theoretic Security* (pp. 223-231). Springer Berlin Heidelberg.

Matthews, R. A. (1993). The use of genetic algorithms in cryptanalysis. *Cryptologia, 17*(2), 187-201.

Von Mises, R. (1957). *Probability, statistics, and truth*. Courier Corporation.

Oldham, K. B., Myland, J., & Spanier, J. (2010). *An atlas of functions: with equator, the atlas function calculator*. Springer Science & Business Media.

Paar, C., & Pelzl, J. (2009). Understanding cryptography: a textbook for students and practitioners. Springer Science & Business Media.

Plutarch. *The Parallel Lives: The Life of Lysander* 19, 5-7.

Childs, J. R. (2002). *General Solution of the ADFGVX Cipher System*. Aegean Park Press.

Rivest, R. L., & Schuldt, J. C. (2014). Spritz-a spongy RC4-like stream cipher and hash function. *Proceedings of the Charles River Crypto Day, Palo Alto, CA, USA, 24*.

Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM, 21*(2), 120-126.

Shannon, C. (1949). Communication theory of secrecy systems. *Bell System Technical Journal, 28*(4), 656-715.

Smith, L. D. (1955). *Cryptography: the science of secret writing*. Courier Corporation.

Suetonius, T. *The Lives of the Caesars: The Life of Julius Caesar* 56, 6.

Sullivan, B. (2001). FBI software cracks encryption wall. *MSNBC*. http://www.nbcnews.com/id/3341694/ns/technology_and_science-security/t/fbi-software-cracks-encryption-wall

Vernam, G. S. (1926). Cipher printing telegraph systems: For secret wire and radio telegraphic communications. *Journal of the AIEE, 45*(2), 109-115.

Zimmermann, P. R. (1995). *The official PGP user's guide*. MIT press.