The Galois Field in Cryptography

Or Finite Fields (so-named in honor of Évariste Galois)
by Ivan Yurukov

1. What is Cryptography?

Cryptography is a method of protecting information and communications through the use of codes, so that only those for whom the information is intended can read and process it. The prefix "crypt-" means "hidden" or "vault" -- and the suffix "-graphy" stands for "writing."

2. What is perfect secrecy?

The perfect secret is when something is hidden or encrypted in a way that does not leave behind any footprints. And when you try to decrypt the hidden message or what is hidden you do not have any trace which can use to decrypt. The only way to decrypt is just to make random assumptions.

3. What is a one-time pad (Vernam Cipher)?

One Time Pad is a method for encrypting a message (which can be plain text, images or any file). This encryption method is most secure and proven mathematically in 1945 by Claude Shannon.

Imagine that Alice has to pass on a message to Bob "on the bridge". This message contained Latin characters, and the Latin alphabet is composed of 26 letters (interval is skipped for now). We have to generate a random key which we will use to encrypting and decrypting the message. For this purpose we use a dice with 26 sides and roll it 11 times (the numbers of symbols in the massage). We have the message, which is with 11 characters and any character corresponds to the number of the alphabet (a-1, b-2, c-3...26-z). And we have the key which is 11 characters. Now we can correlate any character from the message corresponding to a character from the key.

Alphabet	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Order	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

note: *In the alphabet we have `26 letters` . And when we addition two letters `s` (from the message) with number `19` with number `18` from the randomly generated key it is not `37` . Te equation is `19 + (7 + 11)` , which is `26 + 11` which in finite field of `26 elements` is `11` .*
Or `(X + Y) mod 26 => (19+18) mod 26 => 37 mod 26 = 11`

plaintext:	o	n		t	h	e	b		r	i	d	g	e
in numbers	15	14		20	8	5	2		18	9	4	7	5
addition	+	+		+	+	+	+		+	+	+	+	+
key numbers	2	9		15	4	21	22		9	13	4	16	19
equal	=	=		=	=	=	=		=	=	=	=	=
ciphertext_numbers mod()	17	23		35mod(26)	12	26	24		27mod(26)	24	8	23	24
ciphertext_numbers	17	23		9	12	26	24		1	24	8	23	24
ciphertext_text	q	w		i	l	z	x		a	x	h	w	x

ciphertext_text	**q**	**w**	**i**	**l**	**z**	**x**	**a**	**x**	**h**	**w**	**x**
ciphertext_numbers	17	23	9	12	26	24	1	24	8	23	24
subtraction	-	-	-	-	-	-	-	-	-	-	-
key numbers	2	9	15	4	21	22	9	13	4	16	19
equal	=	=	=	=	=	=	=	=	=	=	=
plaintext_numbers mod()	15	14	-6mod(26)	8	5	2	-8mod(26)	9	4	7	5
plaintext_numbers	15	14	20	8	5	2	18	9	4	7	5
plaintext	o	n	t	h	e	b	r	i	d	g	e

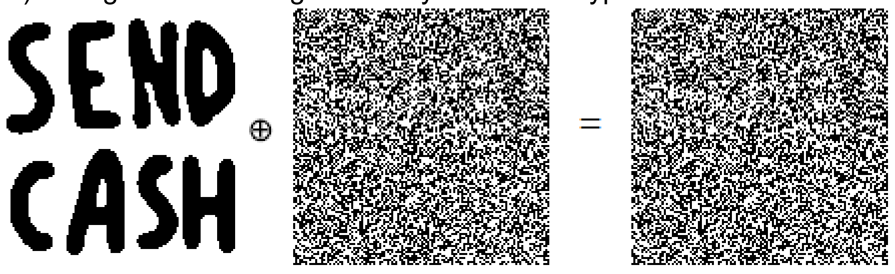
When the key is used only one time the One Time Pad meets the requirements for perfectly secret. We do not see any repetitions because the key length is equal to the message. There is not any information as the frequency of use of characters in different languages and etc. We can't find any trace for the key or the original plain text message. The only option is to make random assumptions for the key or message. In this case, we have 11 symbols and any of them has 26 options (letters) or `26*26*26*26*26*26*26*26*26*26*26 = 3 670 344 486 987 776` and any next symbol makes the combinations increase exponentially.
note: it is not necessary for the message to contain words from spoken language.

When we have some plain text message which is combined with a randomly generated key the result is equivalent to the randomly generated symbols.

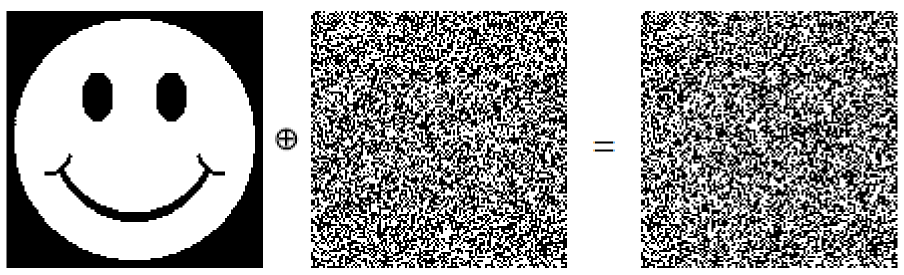
4. What happens if we try to use a one-time pad many times?

As the name "One Time Pad" suggests, it is obvious that this method can only be used once in cryptography. If we use a key two or more times or using it repeatedly in a single encrypted message, we make the encrypted message insecure because we leave traces. And these traces make it easier to be hit or calculate the key and original massage.

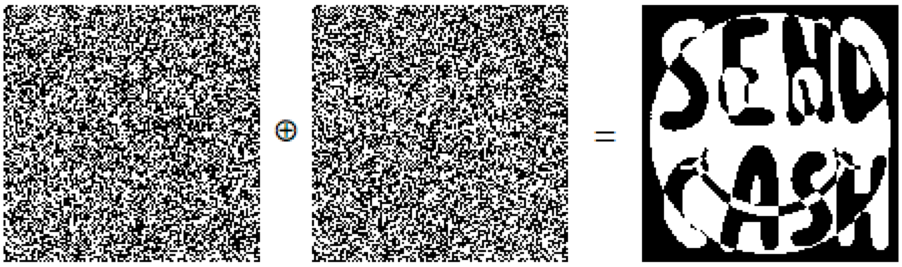
There is a great graphical representation (which is from cryptosmith) of the possible problems that arise from reusing a one-time pad. Let's say you have the image and you encrypt it by using the binary one-time-pad (xor-ing on black and white). You get the following extremely secure encryption



If you then encrypt a smiley face with the same one-time-pad, you get another secure encryption.



But if you have both and you xor them together, then you get the image which, as you can qualitatively and intuitively see is very insecure.



Reusing the same key multiple times is called giving the encryption 'depth' - and it is intuitive that the more depth given, the more likely it is that information about the plaintext is contained within the encrypted text.

5. What is symmetrical & asymmetrical encryption?

In the example mentioned above (What is a one-time pad) we discussed how with one key we encrypt and with the same key we decrypt the message. In real life and in different cases this option is not always the right one. In cryptography exists and another way which is asymmetric. It is a method which uses a pair of keys, one key for encrypting which is public and everybody knows it and can use(only to encrypt the message), and other is a private key which is known only on the owner of the pair keys and can be used for decrypting.

The main advantage of symmetrical encoding and decoding is the speed, which speed i faster than the asymtreic encoding ad decoding. Which explains the use of different methods in different situations.

For example if you want to encrypt the hard drive in your computer, the right way is to use a symmetric encryption method. In this case the key is with you and only you can use the hard drive and decrypt the information on it. But if you want to send an encrypted message to somebody one of the right methods is to use asymmetric encryption. If you know the public key from the other side you can use it to encrypt the message and send it. Nobody can decrypt it, because only the owner has the private key needed to be decrypt the message. In practice it is possible to securely exchange the key from symmetric encryption by the Diffie–Hellman method.
A popular symmetric method is AES and a popular asymmetric method is RSA.

6. How to encrypt one-bit messages?

In the computer, the information is recorded in binary code and the smallest element of information is 1bit (0 or 1). Let's try to encrypt one-bit information (0 or 1). For this purpose, we have to create a key which has to be 1 bit long (0 or 1).

Example:

1bit message	**0**
	XOR
1bit key	1
equal	=
1bit cypherbit	**1**

1bit cypherbit	**1**
	XOR
1bit key	1
equal	=
1bit message	**0**

IN this case it is easy to understand the original information, because we have only two options - 0 or 1. This is the reason the binary information has to be encrypted in big blocks(more than 8 bit). We can use 32 bit, 64 bit, 128bit, 256bit blocks. Nowadays the use of 128-bit blocks/keys is accepted as a secure encryption method.

7. How to extend the one-bit encryption system to many bits?

In real life, we need to encrypt much bigger information than one bit, the information can be thousands of terabytes. In this case, the information which will be encrypted is divided into blocks and the length of any block depends on the length and the method which we will use to encrypt the information. **Exapmle with 8 bit**

8bit message	**1**	**0**	**1**	**0**	**0**	**1**	**1**	**0**
	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR
8bit key	1	1	0	1	0	1	0	1
equal	=	=	=	=	=	=	=	=
8bit cypherbits	**0**	**1**	**1**	**1**	**0**	**0**	**1**	**1**

8bit cypherbits	**0**	**1**	**1**	**1**	**0**	**0**	**1**	**1**
	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR
8bit key	1	1	0	1	0	1	0	1
equal	=	=	=	=	=	=	=	=
8bit message	**1**	**0**	**1**	**0**	**0**	**1**	**1**	**0**

Let's show the sentence "I will go to the river and sail with a boat." in binary code.

`01001001 00100000 01110111 01101001 01101100 01101100 00100000 01100111 01101111 00100000 01110100 01101111 00100000 01110100 01101000 01100101 00100000 01110010 01101001 01110110 01100101 01110010 00100000 01100001 01101110 01100100 00100000 01110011 01100001 01101001 01101100 00100000 01110111 01101001 01110100 01101000 00100000 01100001 00100000 01100010 01101111 01100001 01110100 00101110`

In case we use 128bit key the information can be divided in 128-bit blocks as: `[01001001 00100000 01110111 01101001 01101100 01101100 00100000 01100111 01101111 00100000 01110100 01101111 00100000 01110100 01101000 01100101] [00100000 01110010 01101001 01110110 01100101 01110010 00100000 01100001 01101110 01100100 00100000 01110011 01100001 01101001 01101100 00100000] [01110111 01101001 01110100 01101000 00100000 01100001 00100000 01100010 01101111 01100001 01110100 00101110 {}{}{}{}]`

We can start encrypting the information block by block and the latest block can be supplemented (padding with null information). The new blocks / information will be decryptable only with the key which is used for encrypting. In the real encrypting we must comply with other conditions to be encrypted in the right way as key derivation function and etc.

8. Why is the system decryptable? How do the participants decrypt the encrypted messages? Why isn't the eavesdropper able to decrypt?

Any invented cryptosystem nowadays is decryptable if we have near unlimited CPU resources. One of the reasons to be theoretically possible to decrypt any system is because they are invented to work with small 128-512 bit keys and they are smaller the original messages (and to be extended to the length of massage we use key derivation functions - roughly told). Nowadays 128bit keys are accepted to be enough and secure because we do not have enough resources (even if we use all the computers on the earth, even if the energy for them is free) to decrypt the message without the right keys. In some cases, the quantum computers are a special option but it is accepted that untill 2030 they can't decrypt a 128bit messages and after that, we need to consider how long, and how much energy will it take to decrypt 128-256-512 bit encrypted messages.

At this stage only the possessors on the crypting key can decrypt the information.

Established cryptosystems are sufficiently secure to be widely used nowadays and eavesdroppers(internet providers or 3rd participants) who don't possess the key can't decrypt the messages because of the mentioned reasons.

9. What is a field?

Formally, a field is a set F together with two binary operations on F called addition and multiplication. A binary operation on F is a mapping F × F → F, that is, a correspondence that associates with each ordered pair of elements of F a uniquely determined element of F. The result of the addition of a and b is called the sum of a and b, and is denoted a + b. Similarly, the result of the multiplication of a and b is called the product of a and b, and is denoted ab or a ⋅ b. These operations are required to satisfy the following properties, referred to as field axioms. In these axioms, a, b, and c are arbitrary elements of the field F.

- Associativity of addition and multiplication: a + (b + c) = (a + b) + c, and a ⋅ (b ⋅ c) = (a ⋅ b) ⋅ c.
- Commutativity of addition and multiplication: a + b = b + a, and a ⋅ b = b ⋅ a.
- Additive and multiplicative identity: there exist two different elements 0 and 1 in F such that a + 0 = a and a ⋅ 1 = a.
- Additive inverses: for every a in F, there exists an element in F, denoted −a, called the additive inverse of a, such that a + (−a) = 0.
- Multiplicative inverses: for every a ≠ 0 in F, there exists an element in F, denoted by a−1 or 1/a, called the multiplicative inverse of a, such that a ⋅ a−1 = 1.
- Distributivity of multiplication over addition: a ⋅ (b + c) = (a ⋅ b) + (a ⋅ c).

This may be summarized by saying: a field has two operations, called addition and multiplication; it is an abelian group under addition with 0 as the additive identity; the nonzero elements are an abelian group under multiplication with 1 as the multiplicative identity; and multiplication distributes over addition.

Example of field is alphabet represented as numbers from 1 to 26 `{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26}` which we used. A field can be ASCII-128 table represented as numbers from 1 to 128 and any bit GF(2) {0,1} or bite GF(2^8) or combination of bites GF(2^128).

10. What is GF(2)? Why is it an algebraic field?

In mathematics, a finite field or Galois field (so-named in honor of Évariste Galois) is a field that contains a finite number of elements. As with any field, a finite field is a set on which the operations of multiplication, addition, subtraction and division are defined and satisfy certain basic rules. The most common examples of finite fields are GF{p^z}, p is prime number, z is positive intiger.

Because computers use a binary bit system (0 and 1), in computer cryptography we use a finite field of 2 elements GF {2} or GF {0,1}

The field GF (2 ^ 8) represents one bite. For computers, we group information into bytes. One bit is 0 or 1, 1 byte is made up of 8 bits. `01010101` is a typical end field GF (2 ^ 8) providing 256 possible end combinations. GF {2 ^ 128} is equal to 16 bytes of 8 bits and for each bit, we have two possible options 0 or 1. `[ 01001001 00100000 01110111 01101001 01101100 01101100 00100000 01100111 01101111 00100000 01110100 01101111 00100000 01110100 01101000 01100101 ]` The possible number of combinations is 2 ^ 128 (a very large number). On this field we can perform the familiar operations addition, subtraction, multiplication, and division, which is why the field is algebraic.

11. What are some current enterprise-grade applications of encryption over GF(2)?

GF (2) is used in one of the most widely used cryptographic standards AES (Advanced Encryption Standard). Accordingly, in SSL certificates, where we also use AES, ie GF is used in the encrypted amount of information on the Internet, in encrypts hard drives and other places where it is used as a method of encryption - the established standard AES. AES appears to be the successor to DES where GF (2) is also used.

12. Implement a cryptosystem based on GF - One Time Pad

Implementation of a one-time Python subsystem. As a final field we use an ASCII table with 128 characters. The example has the ability to encrypt short messages that cannot be decrypted without an encryption key. This key must be used for one message only. If you used it more than once the encrypted message, it is not considered secure.

In [*]:

```
import random

def encrypting(string_input, key):
    """Encryption function, returning an encrypted string in hexadecimal encoding format se
    if len(string_input) == 0 or len(key) == 0: raise Exception("You must put a minimum one
    if len(string_input) > len(key): raise Exception("The KEY must be longer than STRING")
    array_input = list(string_input) # Seperate the incoming string in array
    key = list(key)
    cipher_output = []
    for i in range(len(key)): # We use the length of the key because we don't want to show
        if i >= len(array_input):
            cipher_output.append(0 + ord(key[i])) # When the message characters are ended,
        else:
            cipher_output.append(ord(array_input[i]) + ord(key[i])) # In any character, we
        cipher_output[i] = format((cipher_output[i] % 128), 'x') # mod(128) becouse we want
    cipher_text = ",".join(cipher_output) # convert array to string
    return cipher_text

def decrypting(cipher_text_input, key):
    """Decrypting function, return decrypted string.
    For cipher string have to be encrypted with 'encrypting' and the generated key for this
    cipher_text_input = cipher_text_input.split(",")
    if len(cipher_text_input) > len(key): raise Exception("The KEY must be longer than STRI
    decrypted_symbols = []
    decrypted_string = ""
    key_symbols = list(key)
    for i in range(len(cipher_text_input)):
        decrypted_symbols.append(chr((int(cipher_text_input[i], 16) - ord(key_symbols[i]))
    return decrypted_string.join(decrypted_symbols)

def key_generator(length = 100):
    """Pseudo randomly generated key N characters"""
    #pre_generated_key = [] # in combination with #print("###In decimal:"...... used more t
    #return pre_generated_key
    key_array = [random.randint(33,126) for i in range(length)] # 33-126 use symbols which
    key_string = ""
    for symbol in key_array:
        key_string += chr(symbol)
    return key_string

ans = True
RED = "\033[1;31m" # Bold Red Text
BLUE = "\033[1;34m"
BLACK ="\033[1;30m"
ENDC = '\033[m' # reset to the defaults
while ans:
    print ("""
---ONE TIME PAD---\n1.Encrypting - Enter a plain text message \n2.Decrypting - Enter a ciph
    """)
    #print (RED + "Das ist es!" , ENDC)
    ans=input("Encrypt or Decrypt - Which is your choice?: ")
    if ans=="1":
        plaint_text = input("Type your plain text (up to a hundred ASCII-128 symbols): ")
        key = key_generator()
        print("###Key for decryption:")
        print(BLACK, end = "")
        print(key, ENDC)
        print("###Encrypted/ciphertext:")
        print(RED, end = "")
        print(encrypting(plaint_text, key), ENDC)
    elif ans=="2":
        cipher_text = input("Type your cipher text:")
        key = input("Type cipher key:")
        print("###Decrypted message:")
        print(BLUE, end = "")
        print(decrypting(cipher_text, key), ENDC)
    elif ans=="3":
        print("###Key for decryption and encription - only for one time use:")
        print(BLACK, end = "")
        print(key_generator(), ENDC)
    elif ans=="4":
        plaint_text = input("Type your plain text (use up to a hundred ASCII-128 symbols):
        key = input("Put your key for encryption: ")
        print("###Encrypted/ciphertext:")
        print(RED, end = "")
        print(encrypting(plaint_text, key), ENDC)
    elif ans=="0":
        print("\n Goodbye")
        break
    elif ans !="":
        print("\n Not Valid Choice Try again")
    #ans = False
```

```
---ONE TIME PAD---
1.Encrypting - Enter a plain text message
2.Decrypting - Enter a ciphertext message and a key
3.Generate a key - for one time use
4.Encrypting with a custom key - Enter a plain text message and a key
0.Exit/Quit

Encrypt or Decrypt - Which is your choice?: 1
Type your plain text (up to a hundred ASCII-128 symbols): Hi, I'm a crypto
graphic algorithm One Time Pad.
###Key for decryption:
/$lYPMm%x6,4Z/8~*3V~d'</4/J)[%47zq4[e}1mGPO23ZM2t2Y<~!?N|HsCS5?x;7>JQTj0Ns
v\AK$t%J,{prx$R72Z/R]N0G'8
###Encrypted/ciphertext:
77,d,18,79,19,74,5a,45,59,56,f,26,53,1f,2c,6d,11,25,37,6e,4c,10,1f,4f,15,1
b,31,18,4d,e,28,1f,67,11,3,49,4a,1d,5,56,34,35,6f,2,14,3e,7b,32,74,32,59,3
c,7e,21,3f,4e,7c,48,73,43,53,35,3f,78,3b,37,3e,4a,51,54,6a,30,4e,73,76,5c,
41,4b,24,74,25,4a,2c,7b,70,72,78,24,52,37,32,5a,2f,52,5d,4e,30,47,27,38

---ONE TIME PAD---
1.Encrypting - Enter a plain text message
2.Decrypting - Enter a ciphertext message and a key
3.Generate a key - for one time use
4.Encrypting with a custom key - Enter a plain text message and a key
0.Exit/Quit

Encrypt or Decrypt - Which is your choice?: 2
Type your cipher text:77,d,18,79,19,74,5a,45,59,56,f,26,53,1f,2c,6d,11,25,
37,6e,4c,10,1f,4f,15,1b,31,18,4d,e,28,1f,67,11,3,49,4a,1d,5,56,34,35,6f,2,
14,3e,7b,32,74,32,59,3c,7e,21,3f,4e,7c,48,73,43,53,35,3f,78,3b,37,3e,4a,5
1,54,6a,30,4e,73,76,5c,41,4b,24,74,25,4a,2c,7b,70,72,78,24,52,37,32,5a,2f,
52,5d,4e,30,47,27,38
Type cipher key:/$lYPMm%x6,4Z/8~*3V~d'</4/J)[%47zq4[e}1mGPO23ZM2t2Y<~!?N|H
sCS5?x;7>JQTj0Nsv\AK$t%J,{prx$R72Z/R]N0G'8
```

```
###Decrypted message:
Hi, I'm a cryptographic algorithm One Time Pad.

---ONE TIME PAD---
1.Encrypting - Enter a plain text message
2.Decrypting - Enter a ciphertext message and a key
3.Generate a key - for one time use
4.Encrypting with a custom key - Enter a plain text message and a key
0.Exit/Quit

Encrypt or Decrypt - Which is your choice?:
```

References

https://www.youtube.com/watch?v=KCSZ4QhOw0I (https://www.youtube.com/watch?v=KCSZ4QhOw0I) - Field Definition (expanded) - Abstract Algebra
https://www.youtube.com/watch?v=M984ihHNlp0 (https://www.youtube.com/watch?v=M984ihHNlp0) - Galois Field {GF(2), GF(3), GF(5), GF(7)}
https://www.youtube.com/watch?v=vKRMWewGE9A (https://www.youtube.com/watch?v=vKRMWewGE9A) - Perfect secrecy | Journey into cryptography | Computer Science | Khan Academy
https://www.youtube.com/watch?v=ERp8420ucGs (https://www.youtube.com/watch?v=ERp8420ucGs) - Symmetric Key and Public Key Encryption
https://stackoverflow.com/questions/2543566/encryption-of-a-single-character (https://stackoverflow.com/questions/2543566/encryption-of-a-single-character) - encryption of a single character
https://en.wikipedia.org/wiki/One-time_pad (https://en.wikipedia.org/wiki/One-time_pad) - One-time pad
https://www.khanacademy.org/computing/computer-science/cryptography/crypt/v/one-time-pad (https://www.khanacademy.org/computing/computer-science/cryptography/crypt/v/one-time-pad) - The one-time pad
https://crypto.stackexchange.com/questions/59/taking-advantage-of-one-time-pad-key-reuse (https://crypto.stackexchange.com/questions/59/taking-advantage-of-one-time-pad-key-reuse) - Taking advantage of one-time pad key reuse?
https://core.ac.uk/download/pdf/8767685.pdf (https://core.ac.uk/download/pdf/8767685.pdf) - Some Basic Cryptographic Requirements for Chaos-Based Cryptosystems
...