Problem 1. «POLY»

During a job interview, Bob was proposed to think up a small cryptosystem that operates with integers. Bob invented and implemented a complex algorithm `POLY` that can be represented mathematically as a polynomial. Namely, if x is a plaintext, then ciphertext y is equal to $p(x)$, where p is a polynomial with integer coefficients.

Bob's employer decided to test it. At first, he encrypted the number 20 and obtained the number 7. Secondly, he encrypted the number 15 and obtained the number 5. After that he said to Bob that there was a mistake in the implementation of the algorithm and did not hire him. What was wrong?

Problem 2. «Stairs-Box»

Nicole was climbing stairs and has found a box containing a curious permutation on the set of elements $\{0, 1, \ldots, 63\}$:

$$S = [\ 13,18,20,55,23,24,34,\ 1,62,49,11,40,36,59,61,30,\\ 33,46,56,27,41,52,14,45,\ 0,29,39,\ 4,\ 8,\ 7,17,50,\\ 2,54,12,47,35,44,58,25,10,\ 5,19,48,43,31,37,\ 6,\\ 21,26,32,\ 3,15,16,22,53,38,57,63,28,60,51,\ 9,42\ ]$$

So, the element 0 it maps to 13, the element 1 to 18, etc.

Nicole understands that it is possible to consider such a permutation as a vectorial Boolean function $S : \mathbb{F}_2^6 \to \mathbb{F}_2^6$ if every number between 0 and 63 one replaces with a binary vector of length 6. For instance, $S(000010) = (010100)$, since S maps 2 to 20. She knows that S can be given in terms of coordinate functions as $S(x) = (s_1(x), \ldots, s_6(x))$, and each Boolean function s_i can be represented in the algebraic normal form using binary operations XOR and AND in the following way: $s_i(x) = \bigoplus_{I \in \mathcal{P}(N)} a_I \big(\prod_{i \in I} x_i \big)$, where $\mathcal{P}(N)$ is the power set of $N = \{1, \ldots, 6\}$ and $a_I \in \mathbb{F}_2$.

A label on the box said that the function S can be represented as a composition of three maps in the following way:

$$S = A \circ X \circ B,$$

where $A, B : \mathbb{F}_2^6 \to \mathbb{F}_2^6$ are **linear maps** and X is a function with a **short arithmetic expression modulo** 64. Nicole knows that a linear map over $\mathbb{F}_2^6$ can be defined by multiplication with a 6×6 matrix over $\mathbb{F}_2$. But she wonders what is supposed by "a short arithmetic expression modulo 64"? Probably, Nicole also should consider maps as classical modular operations such as addition, substraction, multiplication modulo 64?..

Help Nicole to find the secret function X and the respective maps A, B!

Problem 3. «Hidden RSA»

Bob has learned about the public-key cryptography and now anyone can send a secret message to him. The message is encoded by a nonnegative integer x which has at most 70 digits in the decimal representation. To send a message for Bob, one has to enter it on his webpage. After the message is entered, it is immediately encrypted using RSA. The encryption result is

$$\texttt{Encr}(x) = x^e \bmod n,$$

where n is a modulus (product of two distinct odd primes p and q) and e is a public exponent (coprime with $p-1$ and $q-1$). Bob is afraid of hackers and does not disclose either n or e (even though this contradicts the usual usage of the RSA cryptosystem).

Victor has intercepted the encrypted message

$$y = 7151189668132483345836139288518434493333315983086387860018921207377758217 8173,$$

which Alice has sent to Bob.

Help Victor to decrypt y. You can enter any allowed message x on the Bob's website and receive in response the corresponding ciphertext $\texttt{Encr}(x)$.

Problem 4. «Orthomorphisms»

A young cryptographer Bob wants to build a new block cipher based on the Lai-Massey scheme. The Lai-Massey scheme depends on a finite group G with the neutral element e and an orthomorphism of G. Bob decides to use a nonabelian group and chooses a dihedral group D_{2^m}, $m \geqslant 4$, generated by a, u with presentation

$$a^{2^{m-1}} = e,\ u^2 = e,\ ua = a^{-1}u.$$

Let θ be a permutation of a finite group G. Then θ is called an **orthomorphism of** G if the mapping $\pi : \alpha \mapsto \alpha^{-1}\theta(\alpha)$ is a permutation of G.

Bob needs to construct an orthomorphism of D_{2^m}. He considers the set DM_m consisting of all mappings $\theta^{(r_1,r_2,c_1,c_2)}_{(q_1,q_2,b_1,b_2)}$ on D_{2^m} given by

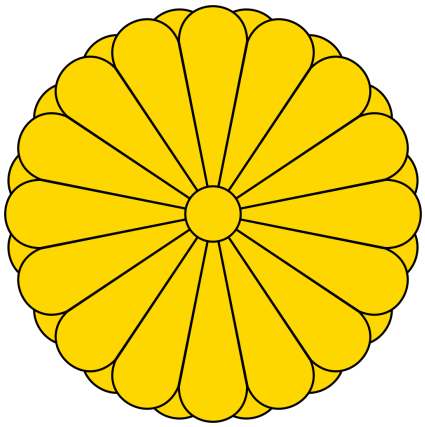
$$\theta^{(r_1,r_2,c_1,c_2)}_{(q_1,q_2,b_1,b_2)} : a^i \mapsto \begin{cases} a^{r_1 i + c_1} & \text{if } i \in \{0,\dots,2^{m-2}-1\}, \\ a^{r_2 i + c_2}u & \text{if } i \in \{2^{m-2},\dots,2^{m-1}-1\}, \end{cases}$$

$$\theta^{(r_1,r_2,c_1,c_2)}_{(q_1,q_2,b_1,b_2)} : a^i u \mapsto \begin{cases} a^{q_1 i + b_1}u, & \text{if } i \in \{0,\dots,2^{m-2}-1\}, \\ a^{q_2 i + b_2}, & \text{if } i \in \{2^{m-2},\dots,2^{m-1}-1\}, \end{cases}$$

and depending on $b_i, c_i, r_i, q_i \in \{0,\dots,2^{m-1}-1\}$ for $i \in \{1,2\}$, where the operations addition and multiplication are over the residue ring $\mathbb{Z}_{2^{m-1}}$.

Q1 Let $m = 4$. Help Bob to describe all orthomorphisms of DM_m and find their number.

Q2 For each $m \geqslant 4$, help Bob to describe all orthomorphisms of DM_m, i. e. give necessary and sufficient conditions on b_i, c_i, r_i, q_i for $i \in \{1,2\}$ such that $\theta^{(r_1,r_2,c_1,c_2)}_{(q_1,q_2,b_1,b_2)}$ is an orthomorphism of D_{2^m}.



Problem 5. «JPEG Encoding»

In order to decrease the readability of the exchanged messages, Alice and Bob decided to encode their messages using JPEG image compression. They write (or draw) their message in a graphics software, save it as a JPEG file and then encrypt the resulting file using some encryption algorithm.

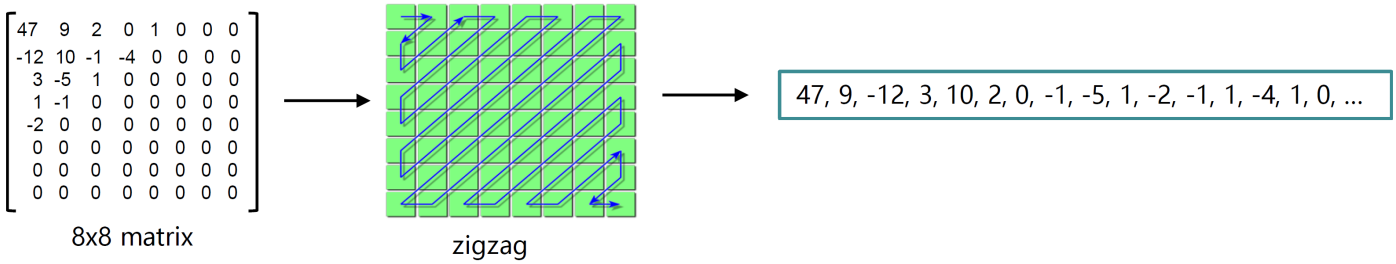
Let us describe the details of the JPEG encoding. The matrix of pixels is first divided into 8×8 matrices, and then the matrices of the type presented below are obtained from them using discrete cosine transform (DCT) and quantization. An interesting characteristic of these matrices is that most of the non-zero data is concentrated in the upper left corner of the matrix, and most of the data in the lower right corner is 0. After that, the matrix is encoded using 0's and 1's.

One example of the matrix encoding is the following algorithm:

1. First, the zigzag rule is used to convert the 8×8 matrix into a one-dimensional vector;
2. Then the Exp-Golomb code is used to encode each number in the vector. Each number (aside from 0, which is encoded as just one bit 0) is encoded by three parts:
   - *length*: a sequence of 1's corresponding to the length of the binary representation of the number, followed by 0 to mark the end of the length sequence;
   - *sign*: a bit representing the sign of the number: 0 for negative, 1 for positive number;
   - *residual*: the binary representation of the number, with the leading 1 omitted.

   For example, the number 47 is encoded as the sequence $\underbrace{1111110}_{length}\ \underbrace{1}_{sign}\ \underbrace{01111}_{residual}$;
3. All encoded sequences are then concatenated and a 6-bit sequence is added to the front. These 6 bits represent the number of non-zero elements in the encoded sequence.

An example. Let us consider how the algorithm works.



We can see that after Exp-Golomb coding, the 8×8 DCT quantized matrix above can be binarized using 91 bits (see below). Note that using the inverse process of the encoding method, we can get the original 8×8 matrix from these 91 bits.

$$\underbrace{001110}_{\#\text{ of non-zero elements}}\ \underbrace{1111110101111}_{47}\ \underbrace{111101001}_{9}\ \underbrace{111100100}_{-12}\ \underbrace{11011}_{3}\ \underbrace{111101010}_{10}\ \underbrace{11010}_{2}\ \underbrace{0}_{0}\ \underbrace{100}_{-1}\ \underbrace{1110001}_{-5}\ \underbrace{101}_{1}\ \underbrace{11000}_{-2}\ \underbrace{100}_{-1}\ \underbrace{101}_{1}\ \underbrace{1110000}_{-4}\ \underbrace{101}_{1}$$

Problem for a special prize! Your task is to design an encoding algorithm providing as short as possible output strings for the given 100 000 matrices (here is a file with matrices, and non-zero elements of each matrix are concentrated in the upper left corner). The less the sum of the lengths of the strings, the more scores you get for this problem. The encoding process must be reversible, that is, the original matrix can be obtained from the bit string using inverse coding.

Problem 6. «Miller — Rabin revisited»

Bob decided to improve the famous Miller — Rabin primality test. The odd number n being tested is represented in the form $n-1 = 2^k 3^\ell m$, where m is not divisible by 2 or 3.

The modified primality test is the following:

1. Take a random $a \in \{2, \ldots, n-2\}$.
2. Put $a \leftarrow a^m \bmod n$. If $a = 1$, return "`PROBABLY PRIME`".
3. For $i = 0, 1, \ldots, \ell - 1$ do the following steps:
   (a) $b \leftarrow a^2 \bmod n$;
   (b) if $a + b + 1$ is divisible by n, return "`PROBABLY PRIME`";
   (c) $a \leftarrow ab \bmod n$.
4. For $i = 0, 1, \ldots, k - 1$ repeat:
   (a) if $a + 1$ is divisible by n, return "`PROBABLY PRIME`";
   (b) $a \leftarrow a^2 \bmod n$.
5. Return "`COMPOSITE`".

Q1 Prove that this algorithm does not fail, that is, not return "`COMPOSITE`", for a prime n.

Q2 **Bonus problem (extra scores, a special prize!)**

A composite integer n may be classified as "`PROBABLY PRIME`" by a mistake. It is known that for the usual Miller — Rabin test the error probability is less than $1/4$. Can this estimation be improved when we are switching to the described algorithm?

Remark. The expression $a \leftarrow a^m \bmod n$ means that a takes a new value that is equal to the remainder of dividing a^m by n.

Problem 7. «CPA game»

Suppose we have a system for the encryption of binary messages. The system has the following characteristics:

- Every message is divided into blocks of length n that are called plaintexts (it is supposed that the length of messages is divisible by n).
- The system employs a block cipher with the encryption function E in cipher block chaining (CBC) mode (see the picture below). A block, an initialization vector IV and a key lengths are equal to n. The result of encryption of the message is a concatenation of IV and the ciphertexts of all plaintexts it consists of.
- The IV for the first message is chosen randomly by using a secure pseudorandom number generator. The last ciphertext block of the i-th message is used as the IV for the $(i+1)$-st message.

Let Alice be an honest user of the system. Victor, an adversary, convinced her to play **chosen–plaintext attack game** (CPA game) with him.

The game is the following:

1. Alice selects a key $k \in \{0,1\}^n$ and chooses a bit $b \in \{0,1\}$.

2. Victor submits a sequence of q queries to Alice. For $i = 1, 2, \ldots, q$ repeat

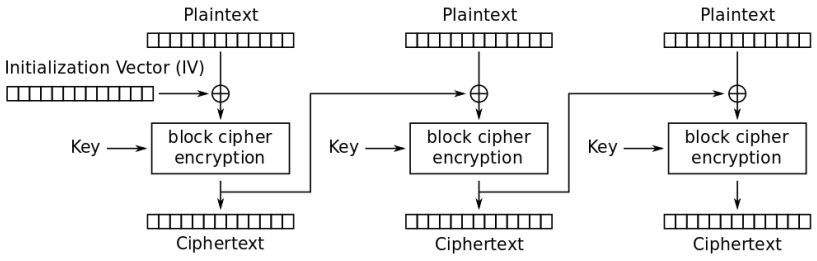
(a) Victor chooses a pair of messages, $m_{i,0}, m_{i,1}$ of the same length.

(b) Alice encrypts $m_{i,b}$ with the key k and gets c_i (the sequence of corresponding IV and ciphertexts). She sends c_i to Victor.

3. Victor outputs a bit $b^* \in \{0,1\}$.

Let W be the event that Victor guesses the bit, that is $b^* = b$. We define Victors's advantage with respect to E as CPAadv $:= |\Pr[\mathrm{W}] - 1/2|$. Victor wins the game if he can build an efficient algorithm such that CPAadv is not negligible.

Task. Construct an efficient probabilistic polynomial-time (PPT) algorithm that wins the CPA game against this implementation with an advantage close to $1/2$.



Cipher Block Chaining (CBC) mode encryption

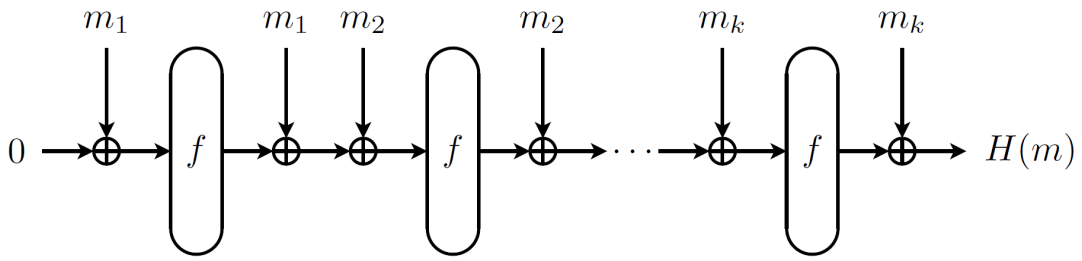
Problem 8. «Collisions»

Consider a hash function H that takes as its input a message m consisting of $k \cdot n$ bits and returns an n-bit hash value $H(m)$. The message m is at least one block long ($k \geqslant 1$), and can be split into k blocks of n bits each: m_1, m_2, …, m_k. Let f be a function which takes an n-bit input and returns an n-bit output. We will use $\oplus$ to denote the bitwise exclusive-or operator.

The hash function H is defined iteratively as follows:

$$h_i := m_i \oplus f(h_{i-1} \oplus m_i),$$

where all n bits of h_0 are zero, and $H(m) := h_k$. Below is an illustration of the hash function H.



A *collision* for H is defined as a pair of distinct messages (m, m') so that $H(m) = H(m')$. Given a message m and its corresponding hash value $H(m)$, a *second preimage* for H is defined as a message $m' \neq m$ so that $H(m) = H(m')$.

Suppose that f is a secret random function and that you have obtained $10 \cdot n$ random different pairs $(x, f(x))$ of argument and value of the function f. Under these restrictions, solve the following problems. Algorithms in Q1 and Q2 must give a solution with a high probability ($> 1/2$).

Q1 Propose an algorithm which finds a collision for H.

Q2 Propose an algorithm which, given a message m and its corresponding hash value $H(m)$, finds a second preimage m' for H.

Q3 Suppose that $n = 256$ bits and the message m is "`A random matrix is likely decent`". Find a second preimage m' for this message.

Remark 1. The text message is converted into a bit sequence as follows: first, each character is converted into a 8-bit integer according to the UTF-8 encoding, and then these integers are concatenated together using the big-endian ordering. For example, the string "`Hello`" is converted into the sequence of integers $(72, 101, 108, 108, 111)$ which then gives the following binary string: `0100100001100101011011000110110001101111`. You can give your answer to this task in the form of a binary sequence or a hexadecimal sequence.

Remark 2. You can evaluate the hash function H on any input message here. The message being hashed should be presented as either a binary sequence or a hexadecimal sequence, starting with a symbol `b` or `h` which specifies the representation. Here you can find a list of values of the function f on 512 different inputs (binary sequences are presented as integers).

Problem 9. «Bases»

Problem for a special prize! Let us consider the vector space $\mathbb{F}_2^r$ consisting of all binary vectors of length r. For any d vectors $x^i = (x_1^i, \ldots, x_r^i)$, $i = 1, \ldots, d$, $d > 0$, it is defined the componentwise product of these vectors equal to $(x_1^1 \ldots x_1^d, \ldots, x_r^1 \ldots x_r^d)$. The empty product (when no element is involved in it) equals the all-ones vector.

Let $s \geqslant d > 1$ be positive integers and let r be defined by the formula $r = \sum_{i=0}^{d} \binom{s}{i}$, where $\binom{s}{i}$ denotes the binomial coefficient. Let $\mathcal{B}$ be a basis of the vector space $\mathbb{F}_2^r$, and let $\mathcal{F} \subseteq \mathbb{F}_2^r$ be a family of s binary vectors such that all possible componentwise products of up to d vectors from the family $\mathcal{F}$ (including the empty product) form the basis $\mathcal{B}$.

Given s, d, r defined above, describe all (or at least some) bases $\mathcal{B}$ for which such family $\mathcal{F}$ exists or prove that such bases do not exist.

Suggest practical applications of such bases.

Example. Let $s = 2$, $d = 2$ and $r = 4$. Consider the following family of 2 vectors $\mathcal{F} = \{(1100), (0110)\}$. Then all componentwise products of 0, 1 and 2 vectors from the family $\mathcal{F}$ form the basis $\mathcal{B} = \{(1111), (1100), (0110), (0100)\}$ of the vector space $\mathbb{F}_2^4$.

Problem 10. «AES-GCM»

Alice is a student majoring in cryptography. She wants to use AES-GCM-256 to encrypt the communication messages between her and Bob (see the next page for a description of AES-GCM-256). The message format is as follows:

However, Alice made some mistakes in the encryption process since she is new to AES-GCM. Your task is to attack the communications.

Q1 You intercepted some messages sent by Alice. You can find these messages in the directory "Task_1".

Moreover, you know that the plaintext (unencrypted payload) of the first message (0.message) is "**Hello, Bob! How's everything?**" (without quotes, encoded in UTF-8).

Try to decrypt any message in the directory "Task_1".

Q2 In this task, you further know that the AAD (additional authenticated data) used by Alice in each message is Header || Initialization Vector:

You want to tamper some messages in the directory "Task_2".

You pass this task if you can modify at least one bit in some message so that Bob can still decrypt the message successfully.

Q3 Alice has noticed that the messages sent by her have been tampered with. So she decides to enhance the security of her encryption process.

Instead of using Header || Initialization Vector as the additional authenticated data (AAD), Alice further generates 8 bytes data X by some deterministic function f and the AES secret key K, where

$$X = f(K).$$

In each message, she uses Header || Initialization Vector || X as the AAD.

You also intercepted some messages sent by Alice, see these messages in the directory "Task_3". Try to tamper any message!

Q4 **Bonus problem (extra scores, a special prize!)**

You have successfully tampered with the messages in Q2. However, the attacks will be easy to detect if the tampered message cannot be decrypted to some meaningful plaintext.

In this task, try to tamper the messages in Q2 so that the tampered message can still be decrypted to some plaintext that people can understand. **Remark:** Tampering with the Header or Initialization Vector of a message will not be accepted as a solution, you need to tamper with the encrypted payload to produce some other ciphertext which did not appear in any message included.

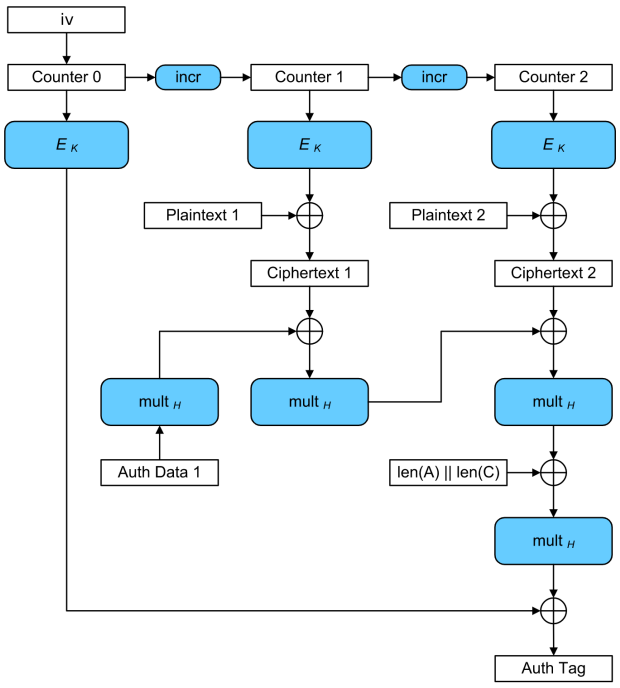
Turn to the next page.

Remark. In cryptography, Galois/Counter Mode (GCM) is a mode of operation for symmetric-key cryptographic block ciphers widely adopted for its performance.

The operation is an authenticated encryption algorithm designed to provide both data authenticity (integrity) and confidentiality.

Let us describe the AES-GCM algorithm. First, let us describe some notations used:

1. IV (Initializaton Vector) is a block of 96 bit; Alice arbitrarily chooses it before the encryption;
2. E_K is an AES-256 encryption function, operating on the blocks of 128 bits and encrypting them using the key K;
3. $mult_H$ is the multiplication operation. It multiplies an input block and a block H of 128 bits as numbers in a Galois field of order 2^{128}; the result is a 128-bit block;
4. H is the result of an AES-256 encryption of the 128-bit all-zero vector;
5. $incr$ is a certain (non-secret) increment function, which transforms blocks of 128 bits into blocks of 128 bits;



The *encryption* is performed as follows:

1. First, Alice chooses IV and splits the plaintext P into the blocks of 128 bits: $P = P_1||P_2||\dots||P_{n-1}||P_n$. The last block is not padded, let us denote the length of it by r;
2. The vector IV is transformed into a 128-bit block CB_0 (Counter 0 on the picture);
3. For $i = 1, \dots, n$, the block CB_{i-1} is incremented into the block CB_i; the latter is then encrypted with AES encryption E_K and summed with the block P_i of the plaintext using bitwise XOR operation. The result is the block C_i of the ciphertext;
4. For the last block, only the r leftmost bits of the encrypted block CB_n are summed with P_n;
5. The resulting ciphertext is the concatenation of the obtained blocks: $C := C_1||C_2||\dots||C_{n-1}||C_n$.

The *Authentication Tag* is obtained as follows:

1. The block of Additional Authentication Data is multiplied with $mult_H$, resulting in a block Y_0;
2. For $i = 1, \dots, n$, the block C_i of the ciphertext is summed with the block Y_{i-1}; the sum is then multiplied using $mult_H$ again, resulting in a block Y_i;
3. The block Y_n is summed with the block $len(A)||len(C)$, which consists of the encoded lengths of the AAD and of the ciphertext C; the result is once again multiplied using $mult_H$ and then summed with the AES-encrypted block CB_0; this gives us the Authentication Tag.

For more details of GCM, we refer to [1].

[1] Dworkin M. Sp 800-38d. Recommendation for block cipher modes of operation: Galois/counter mode (GCM) and GMAC. National Institute of Standards & Technology, 2007.