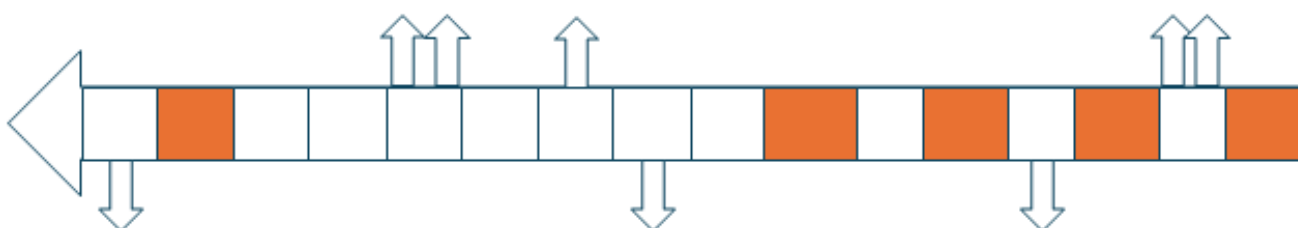




# Problem 1. «Where to meet?»

Alice discovered an encrypted message sent by Bob, along with a few hints. Using these hints, she was able to decrypt it immediately. What was the original message?

Encrypted message: vkbkxoqebbktpdvt





## Problem 2. «Studying quantum security»

### Problem for a special prize!

Starting from papers of O. Goldreich, Sh. Goldwasser, and S. Micali cryptographers study how to construct random functions. Ok, let the function be random if *it looks random* to adversaries. It is possible to define a *pseudorandom function (PRF)* / *pseudorandom permutation (PRP)* as a function/permutation with the following property: no efficient classical algorithm, when given oracle access, can distinguish it from a truly random function/permutation.

There are some modern variants of modeling quantum adversaries for ciphers, see the paper M. Kaplan, G. Leurent, A. Leverrier «Quantum Differential and Linear Cryptanalysis» // IACR Transactions on Symmetric Cryptology, Vol. 2016, No. 1, pp. 71–94. DOI: 10.13154/tosc.v2016.i1.71-94. Namely, there are

**Standard security:** a block cipher is *standard secure* against quantum adversaries if no efficient quantum algorithm can distinguish the block cipher from PRP (or a PRF) by making only *classical* queries (denote this type as Q1).

**Quantum security:** a block cipher is *quantum secure* against quantum adversaries if no efficient quantum algorithm can distinguish the block cipher from PRP (or a PRF) even by making *quantum* queries (denoted this type as Q2).

The Q2 model of attacks assumes that an adversary can query a quantum cryptographic oracle in superposition. This model can be implemented, when an algorithm under analysis runs on a quantum computer with adequate resources. It is known that most of standardized modes of operation for block ciphers are insecure in the Q2 model. You can read about it for example in the paper M. V. Anand, E. E. Targhi, G. N. Tabia, and D. Unruh «Post-quantum Security of the CBC, CFB, OFB, CTR, and XTS Modes of Operation» // Lecture Notes in Computer Science, Vol. 9606, pp. 44-63, 2016. DOI: 10.1007/978-3-319-29360-8\_4. This means that such block ciphers modes of operation can be considered broken after appearing a quantum computer with resources enough for implementing an attacked algorithm.

Propose and describe a new (previously unknown) block cipher mode of operation secure in the Q2 model, and justify its security in the Q2 model.

Block cipher modes  
for quantum security?





### Problem 3. «One-time signature keys»

The client uses one-time keys for ECDSA signing distinct messages in order to ensure their unlinkability. In order not to store many keys either on the client side or on the verifier side, the signing master key pair  $(sk, pk)$  is used, and each one-time key pair is produced from the master key using the identifier  $id$ .

For example, the following one-time keys will be generated

$pk_{1976}$ :

(61169738454268678583460034936064297717270301135969262814478294443307068229252,  
12811184070684365376227100876061995540032292788249803133543329830909909492736),

$pk_{2025}$ :

(24453843338537316254013968815895450156309709523339826431891869789646992767085,  
25869566496341048395665983125532875931564283027116920587554169802031201907034)

for the master public key  $pk$ :

(108166144854954578764781250917773475575161867139061909129716138841733602595564,  
21094420584895241134570605732830595252834312939967791385802578924623343135307)

and the NIST P-256 elliptic curve, the generator of the group is  $P$ :

(48439561293906451759052585252797914202762949526041747995844080717082404635286,  
36134250956749795798585127919587881956611106672985015071877198253568414405109).

Let's give one message  $m$  and its signature  $(r, s)$ , formed using a one-time key generated from some master key (not equal to the one given in the example above) using the identifier  $id$ . The problem is 1) to find the master public key from which the one-time signature generation key was obtained; 2) to forge signature for the message  $m^* \neq m$  for some new identifier (to find valid  $id^*$  and  $(r^*, s^*)$ ).

**ECDSA.** For convenience, we describe all signature schemes for the case of calculations in a prime order subgroup of an elliptic curve. Let  $P$  be the generator of the subgroup of prime order  $q$ . The elliptic curve is defined over a field  $\mathbb{Z}_p$ , where  $p$  is a prime. Let  $H$  be the hash function that maps binary strings to the elements in  $\mathbb{Z}_q$ .

Below the ECDSA signature scheme is given.

<u>ECDSA.Gen( )</u>	<u>ECDSA.Sign(<math>d, m</math>)</u>	<u>ECDSA.Verify(<math>Q, (r, s), m</math>)</u>
$d \xleftarrow{\mathcal{U}} \mathbb{Z}_q^*$	$e \leftarrow H(m) \bmod q$	<b>if</b> ( $r = 0 \vee s = 0$ ) : <b>return</b> 0
$Q \leftarrow d \cdot P$	$k \xleftarrow{\mathcal{U}} \mathbb{Z}_q^*$	$e \leftarrow H(m)$
<b>return</b> ( $d, Q$ )	$R \leftarrow k \cdot P$	$R \leftarrow s^{-1} \cdot e \cdot P + s^{-1} \cdot r \cdot Q$
	$r \leftarrow R.x \bmod q$	<b>if</b> $R.x \bmod q \neq r$ : <b>return</b> 0
	<b>if</b> $r = 0$ : <b>return</b> $\perp$	<b>return</b> 1
	$s \leftarrow k^{-1} \cdot (e + d \cdot r)$	
	<b>if</b> $s = 0$ : <b>return</b> $\perp$	
	<b>return</b> ( $r, s$ )	



## Problem 4. «Precomputed keys»

Ephemeral keys are often used in authenticated key establishment (AKE) protocols. However, their implementation on low-resource devices requires the generation of these keys, which is not always convenient. Therefore, it is possible to precompute these keys and store their private (public) parts in the (un)protected memory. So, let the adversary have the capability to replace the values written in the unprotected memory.

The problem is to make attacks on the AKE protocol described below that implement the following threats:

- 1) key compromise impersonation of the initiator;
- 2) key compromise impersonation of the responder;
- 3) forward secrecy breaking;

using only the capability to replace the values written in the unprotected memory and the capability to break the interaction (and the capability to know the key corresponding to the threats).

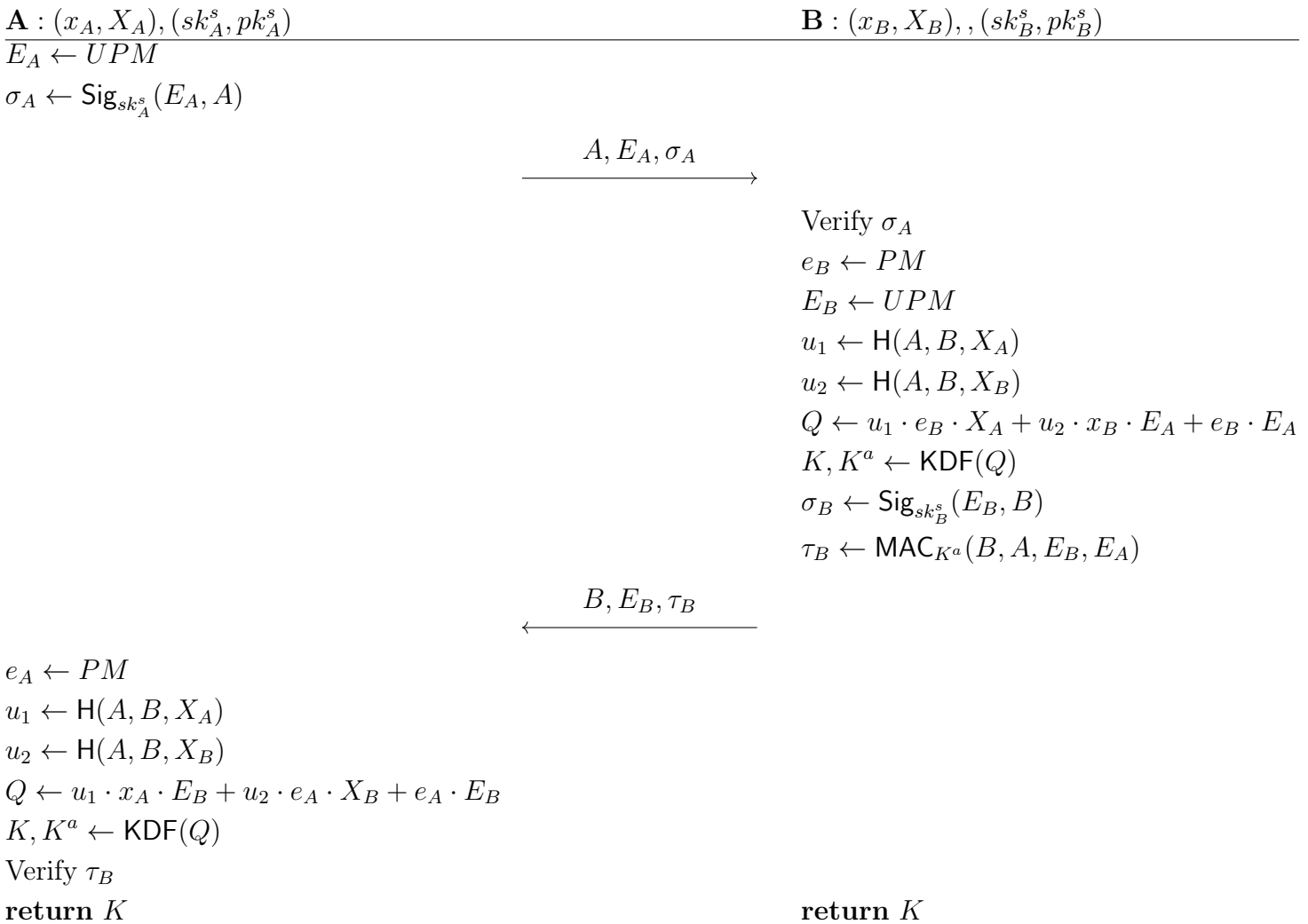
*Key compromise impersonation.* An adversary is said to impersonate a party  $B$  to party  $A$  if  $B$  is honest and  $A$  accepts the session (and establish the key) with  $B$  but party  $B$  has not completed (or even started) the protocol, and an adversary knows the established key. In a successful key compromise impersonation (KCI) attack, an adversary with the knowledge of the long-term private key of a party  $A$  can impersonate  $B$  to  $A$ .

*Forward secrecy breaking.* An adversary with the knowledge of the long-term private key of a party  $A$  (or of a party  $B$ ) is said to break forward secrecy if he/she knows the session keys established before he knew the long-term key.

**The AKE protocol.** Calculations are described for the case of an additive group  $G$  of a prime order  $q$ . The generator of this group are denoted by  $P$ . We denote by  $\mathbb{Z}_q$  a finite field of characteristic  $q$  and assume the canonic representation of the elements in  $\mathbb{Z}_q$  as integers from the set  $\{0, 1, \dots, q - 1\}$ .

We use the following notations for cryptographic keys and shared non-secret values:  $(x, X)$  is a key pair consisting of a scalar  $x$  and corresponding multiple element of the group  $X = x \cdot P$ ;  $(sk^s, pk^s)$  is a key pair of a signature scheme;  $K^a$  is a key of a message authentication scheme;  $K$  is a secret key established as a result of the protocol.

We use the following notations for the used cryptographic mechanisms primitives:  $\text{Sig}_{sk^s}$  and  $\text{MAC}_{K^a}$  are signing and generating the message authentication code algorithms, verification of signatures and message authentication codes is denoted by «Verify  $\sigma, \tau$ »;  $\text{KDF}(\cdot)$  and  $\text{H}(\cdot)$  are key derivation and hash function. We assume that hash values are elements from  $\mathbb{Z}_q$ . Read operations from (un)protected memory are designated as  $a \leftarrow PM$  ( $a \leftarrow UPM$ ).





## Problem 5. «Understand the Oracle»

Bob challenges Alice by providing a matrix  $\mathbf{B} \in \mathbb{Z}^{2n \times 2n}$  and asks her to find an integer vector  $\mathbf{g} \in \mathbb{Z}^{2n}$  that is short in the Euclidean norm and satisfies  $\mathbf{B}\mathbf{f} = \mathbf{g}$ , for some  $\mathbf{f} \in \mathbb{Z}^{2n}$ .

Alice possesses a magical oracle capable of finding short vectors in dimension  $n$ , and it can be invoked any number of times. Bob further guarantees to Alice that there exists a way to reduce the dimension of  $\mathbf{B}$  to  $n$ , and that adding any two coefficients of the vector  $\mathbf{g}$  yields a small value. The first time, Bob challenged Alice by giving the matrix

$$\mathbf{B} = \begin{pmatrix} \mathbf{H}_0 & \mathbf{H}_1 \\ \mathbf{H}_1 & \mathbf{H}_0 \end{pmatrix}.$$

Alice quickly realized that if she splits  $\mathbf{g}$  as  $(\mathbf{g}_0, \mathbf{g}_1)$ , then she can provide the oracle with the matrix  $\mathbf{B}_0 = \mathbf{H}_0 + \mathbf{H}_1$  of dimension  $n$ , such that  $\mathbf{B}_0(\mathbf{f}_0 + \mathbf{f}_1) = (\mathbf{g}_0 + \mathbf{g}_1)$  holds. Since  $\mathbf{g}_0 + \mathbf{g}_1$  is still a short vector (by Bob's guarantee), the oracle can find  $\mathbf{g}_0 + \mathbf{g}_1$ . Similarly, she can invoke the oracle with  $\mathbf{B}_1 = \mathbf{H}_0 - \mathbf{H}_1$  to obtain the vector  $\mathbf{g}_0 - \mathbf{g}_1$ , and ultimately recover  $\mathbf{g}$ .

Bob then made the challenge harder and gave Alice a matrix of the form

$$\mathbf{B} = \begin{pmatrix} \mathbf{H}_0 & \mathbf{H}_1 \\ \mathbf{H}_1^T & \mathbf{H}_0^T \end{pmatrix},$$

where  $\mathbf{H}_0$  is a right circulant matrix and  $\mathbf{H}_1$  is a left-circulant matrix, and  $\mathbf{H}_0^T, \mathbf{H}_1^T$  denote the transposes of  $\mathbf{H}_0$  and  $\mathbf{H}_1$ , respectively. Can you help Alice to reduce this matrix to dimension  $n$  so that she can use the oracle?

**Note 1:** We say that  $\mathbf{H}_0$  is a *right circulant matrix* if it has the form

$$\begin{pmatrix} a_0 & a_1 & \dots & a_{n-1} \\ a_{n-1} & a_0 & \dots & a_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ a_1 & a_2 & \dots & a_0 \end{pmatrix},$$

and similarly  $\mathbf{H}_1$  is a *left circulant matrix* if it has the form

$$\begin{pmatrix} b_0 & b_1 & \dots & b_{n-1} \\ b_1 & b_2 & \dots & b_0 \\ \vdots & \vdots & \ddots & \vdots \\ b_{n-1} & b_0 & \dots & b_{n-2} \end{pmatrix}.$$

**Note 2:** This problem is related to lattice-based cryptography, as finding the short vector  $\mathbf{g}$  corresponds to solving the Shortest Vector Problem (SVP) in the lattice generated by  $\mathbf{B}$ .



## Problem 6. «NTRU-style equation»

### Problem for a special prize!

Alice is fascinated by the NTRU cryptosystem and its simple key equation  $h = f^{-1} * g \pmod{q}$ . In the early proposal of NTRU, the cryptosystem was constructed over the ring  $\mathcal{R} = \frac{\mathbb{Z}[x]}{(x^N-1)}$ , where both  $f$  and  $g$  are sampled with small coefficients, typically from the set  $\{-1, 0, 1\}$ .

Alice assumed that choosing  $N = 127$  and by sampling  $f, g$  with enough entropy would provide sufficient security, as the combinatorial cost of recovering either  $f$  or  $g$  is large, making it difficult for Bob to break. Alice published  $h$  and challenged Bob to find  $f$  and  $g$ . After a few days, however, Bob managed to recover Alice's key and explained that it is easy to attack NTRU keys using lattice reduction techniques in relatively small dimensions by constructing the lattice

$$\mathcal{L}_{CS} = \begin{pmatrix} \mathbf{I}_N & \text{mat}(h) \\ \mathbf{0}_N & q \cdot \mathbf{I}_N \end{pmatrix},$$

and applying lattice reduction, where  $\text{mat}(h)$  is the matrix representation of  $h$  with respect to the ring  $\mathcal{R}$ , which in this case is a right-circulant matrix.

Alice was frustrated by this and did not want to increase  $N$  much. She started reviewing the literature and found a noncommutative variant of NTRU built over a slightly more complex ring: the group ring of the dihedral group  $\mathcal{R}_{D_N} = \mathbb{Z}D_N$ , where  $D_N$  denotes the dihedral group of order  $2N$ , defined as  $D_N = \langle x, y \mid x^N = 1, y^2 = 1, xy = yx^{N-1} \rangle$ . By abuse of notation, we can write

$$\mathcal{R}_{D_N} \approx \frac{\mathbb{Z}[x, y]}{\langle x, y \mid x^N = 1, y^2 = 1, xy = yx^{N-1} \rangle},$$

and any  $f \in \mathcal{R}_{D_N}$  can be written, for simplicity, as  $f = f_0(x) + yf_1(x)$ , where  $f_0(x)$  and  $f_1(x)$  are two polynomials of maximum degree  $N - 1$ .

Alice then proposed modifying the NTRU key equation over this new noncommutative ring as

$$h = f_1^{-1} * g * f_2^{-1} \pmod{q},$$

where  $f_1, f_2 \in \mathcal{S}$ , and  $\mathcal{S}$  is defined as the commutative subring of  $\mathcal{R}_{D_N}$ :

$$\mathcal{S} = \{ f \in \mathbb{Z}D_N \mid fy = yf \}.$$

Here,  $g$  is sampled randomly from  $\mathcal{R}_{D_N}$  with small coefficients from the set  $\{-1, 0, 1\}$ . Alice believed that this construction would prevent Bob from applying lattice reduction directly.

However, when she gave  $h$  to Bob, he pointed out that lattice reduction could still be applied indirectly by constructing  $h + yhy$ , which transforms the key equation into a commutative one:

$$h + yhy \pmod{q} = f_1^{-1} * (g + ygy) * f_2^{-1} \pmod{q} = (f_1 * f_2)^{-1} * (g + ygy) \pmod{q},$$

which is again an NTRU instance.

Alice, now even more determined, constructed her key as

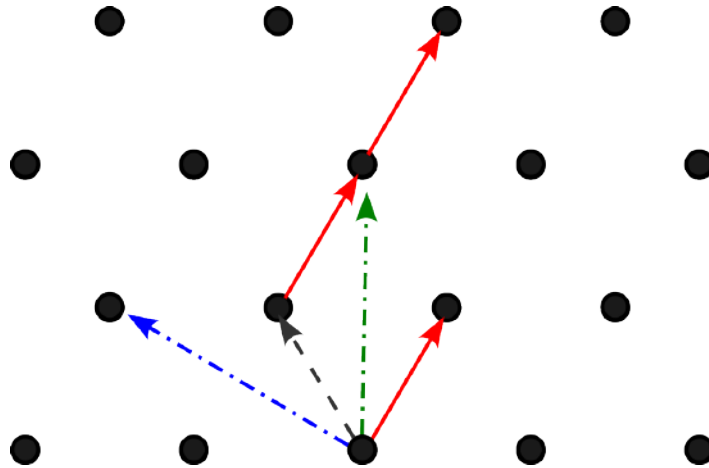
$$h = f_1^{-1} * g * f_2^{-1} \pmod{q},$$

sampling  $f_1, f_2$  from  $\mathcal{S}$  but restricting  $g$  to be sampled from

$$\mathcal{S}^- = \{f \in \mathbb{Z}D_N \mid fy = -yf\}.$$

This time, Alice is convinced that Bob cannot convert  $h$  into an NTRU-style equation and therefore cannot apply lattice attacks directly.

Could you help Bob to disprove Alice, or do you agree with her? Explain.



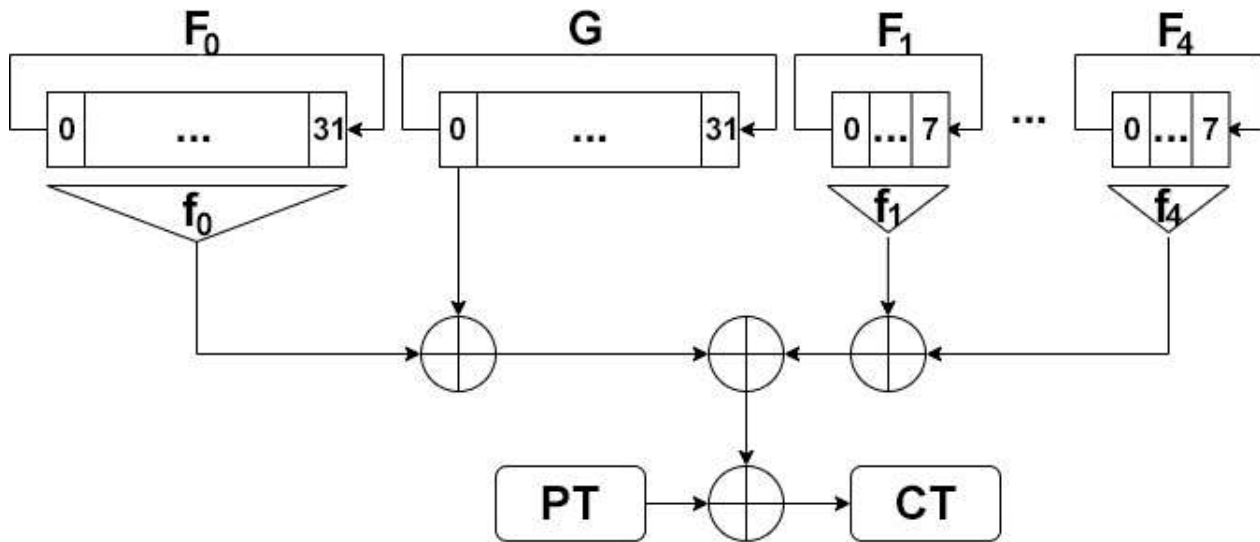


## Problem 7. «Negotiations»

Before the upcoming negotiations, our intelligence managed to intercept an encrypted telegram sent from the enemy headquarters to its negotiation team:

289e1fa87b606203a790f93e3a3e4c1a  
 7b60474a1df415badf49b12b6b6e16ef  
 7e559f015c.

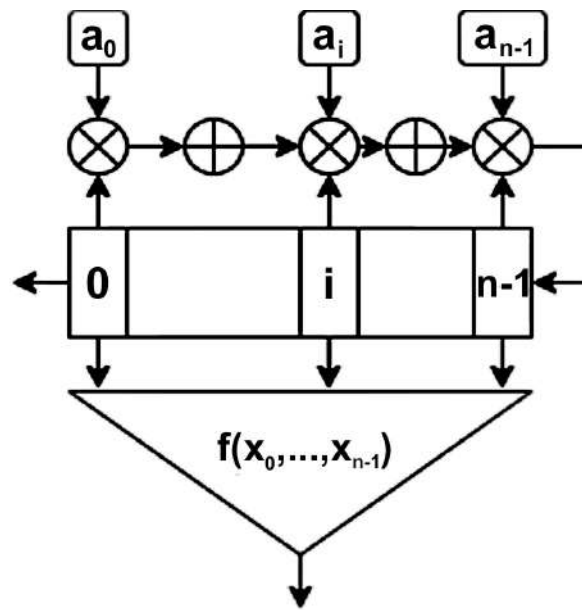
Undercover operative Jack Beam managed to recover the schematics of the cipher device in use. It is given below:



The cipher device is a stream cipher consisting of six binary shift registers, where for each clock cycle a bit of the output keystream is XORed with a bit of plaintext.

Another undercover operative, Jim Daniels, managed to provide a more detailed description of the shift registers.

For each clock cycle, the register output is formed by feeding its state into a filter function  $f$  (least significant bits are on the left). The state is then bitwise multiplied with the coefficients  $a_0, \dots, a_{n-1} \in GF(2)$  of a feedback polynomial  $F(x) = x^n + a_{n-1}x^{n-1} + \dots + a_1x + a_0$ . The results of the multiplications are summed modulo 2, the register state is shifted one cell towards the least significant bits, and the result of the sum is written into the most significant cell, see the next figure.



Jim Daniels wrote down the feedback polynomials:

$$\begin{aligned}
 F_0(x) &= x^{32} + x^{15} + x^9 + x^7 + x^4 + x^3 + 1, \\
 G(x) &= x^{32} + x^{31} + x^{29} + x^{25} + x^{19} + x^{18} + x^{17} + x^{16} + x^9 + x^8 + x^7 + x^3 + x^2 + x + 1, \\
 F_1(x) &= x^8 + x^7 + x^6 + x^5 + x^2 + x + 1, \\
 F_2(x) &= x^8 + x^5 + x^3 + x + 1, \\
 F_3(x) &= x^8 + x^6 + x^5 + x^2 + 1, \\
 F_4(x) &= x^8 + x^6 + x^5 + x + 1.
 \end{aligned}$$

He found the truth tables in hexadecimal format for the filter functions  $f_1, \dots, f_4$ :

$$\begin{aligned}
 f_1 &: 431DCF02529EF04356F2B59E90860BD22FBCEFDDB380F8838767DD716E9712A04, \\
 f_2 &: 4617142FE4475B3FF3D50CBB58E7A0F40CAC85B8C278A0C131FFD85413BE9E4A, \\
 f_3 &: 0138CCC35F889EAA5D8F1EE0442FB54D2AEAEFB96288DB3303FD1FBF65860A20, \\
 f_4 &: 095F8FFC8CA0C53C5D3782243D36EE575186DBD244CB9DE1E0ED1730CACF2053.
 \end{aligned}$$

Also he got the function  $f_0$ . Thus,  $f_0(x_0, \dots, x_{31}) = f(x_5, x_{21}, x_{29}, x_1, x_0, x_{27}, x_{12})$ , where it holds  $f(x_0, \dots, x_7) = x_0x_2x_5x_6 + x_0x_3x_5x_6 + x_0x_1x_5x_6 + x_1x_2x_5x_6 + x_0x_2x_3x_6 + x_1x_3x_4x_6 + x_1x_3x_5x_6 + x_0x_2x_4 + x_0x_2x_3 + x_0x_1x_3 + x_0x_2x_6 + x_0x_1x_4 + x_0x_1x_6 + x_1x_2x_6 + x_2x_5x_6 + x_0x_3x_5 + x_1x_4x_6 + x_1x_2x_5 + x_0x_3 + x_0x_5 + x_1x_3 + x_1x_5 + x_1x_6 + x_0x_2 + x_1 + x_2x_3 + x_2x_5 + x_2x_6 + x_4x_5 + x_5x_6 + x_2 + x_3 + x_5$ .

The second register lacks a filter function, its output is the least significant (leftmost) bit of the register state.

It is known that the first 22 bytes of the intercepted message contain direct output of the cipher device without XORing any plaintext, and the subsequent 15 bytes contain an encrypted message, capable of altering the course of the upcoming negotiations.

The encrypted message consists of letters of the Latin alphabet in a 5-bit encoding:

Symbol	Code	Symbol	Code	Symbol	Code	Symbol	Code
A	00001	H	01000	O	01111	V	10110
B	00010	I	01001	P	10000	W	10111
C	00011	J	01010	Q	10001	X	11000
D	00100	K	01011	R	10010	Y	11001
E	00101	L	01100	S	10011	Z	11010
F	00110	M	01101	T	10100		
G	00111	N	01110	U	10101		

**Question** Could you help your group to decrypt the message and in this way to strengthen its negotiation position?

**Remark.** Let us present here some test vectors for the problem.

**Register 1.**

Initial state: 000000000000000001000001010001000  
 LFSR output: 00000000000000000100000101000100001000000000011001100  
 Register output: 0101001100000001001011011101111011010110110100011100

**Register 2.**

Initial state: 11010001000010010000101011001011  
 Register output: 1101000100001001000010101100101100100000111011101101

**Register 3.**

Initial state: 11111111  
 LFSR output: 11111111011101010011  
 Register output: 00100011010010010110

**Register 4.**

Initial state: 00101110  
 LFSR output: 00101110110111100101  
 Register output: 10001101001001011010

**Register 5.**

Initial state: 00101110  
 LFSR output: 00101110111101100111  
 Register output: 00110100100101101000

**Register 6.**

Initial state: 00101100  
 LFSR output: 00101100110110000111  
 Register output: 00101100010011011000



## Problem 8. «Password to coins»

A famous Roman found a wallet with coins during his morning walk. To open it, he needs to enter a password hidden in the following line:

```
5655555556f012346789abcde5f012346789abcde5f012346789abcde5f012346789
abcde5555555558d6ac5b6c8c8bec8b8c7cec5c9b4b3c8cab8c7cec5c9b47657f607
18293a4bcde56789abcde5f01234f60718293a4bcde56789abcde5f0123444444444
56f57a7b5555555556ecbfe699badb4c3aaff8e4529b553cd616e459a11057a82ddf
155555555
```

Help Julius to get the password and take 5 coins from the wallet.





## Problem 9. «Bijections for ciphers»

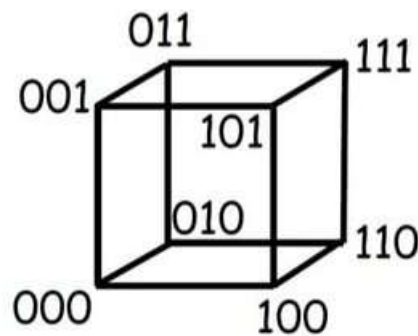
A mapping  $F : \{0, 1\}^n \rightarrow \{0, 1\}^n$ ,  $F = (f_1, \dots, f_n)$ , is such that all coordinate Boolean functions  $f_i$ ,  $i = 1, \dots, n$ , depend on  $k$  variables,  $k \leq n$ . Each function  $f_i$  is defined as follows: by a Boolean function  $g_i$  in  $k$  variables and an integer vector  $m_i$  of length  $k$ , containing the indices of the essential variables.

**Example:** Let  $n = 3$ ,  $k = 2$ ,  $g_1 = g_2 = g_3 = x_1x_2$ ,  $m_1 = (2, 3)$ ,  $m_2 = (1, 3)$ ,  $m_3 = (1, 2)$ ; then  $f_1 = x_2x_3$ ,  $f_2 = x_1x_3$ ,  $f_3 = x_1x_2$ , and the mapping  $F$  is given by the table:

$x_1$	$x_2$	$x_3$	$F$
0	0	0	000
0	0	1	000
0	1	0	000
0	1	1	100
1	0	0	000
1	0	1	010
1	1	0	001
1	1	1	111

**A problem.** Formulate conditions (necessary; sufficient; both) on the functions  $g_i$  and vectors  $m_i$  under which the mapping  $F$  is a bijection (an one-to-one function).

**A requirement:** It should be not necessary to construct the truth table for  $F$  while checking the conditions.





## Problem 10. «Crypto noise»

Bob obtained from Alice the ciphertext  $c = (c_1, c_2, \dots, c_{20})$  that is a vector over  $\mathbb{Z}_{16}$ . He knows that the initial message is a vector  $m = (m_1, m_2, m_3, m_4)$  over  $\mathbb{Z}_{16}$ . He also provided the information that for the ciphertext it holds  $c = mA + e \pmod{16}$  where  $A$  is a  $4 \times 20$  integer matrix

$$A = \left( \begin{array}{cccc|cccccccccccccccccccc} 2 & 2 & 1 & 1 & -1 & 0 & 3 & 7 & -2 & -2 & -1 & -1 & -2 & -2 & -1 & -1 \\ 2 & 1 & 2 & 1 & -2 & -1 & -2 & -1 & -1 & 1 & 2 & 7 & -2 & -1 & -2 & -1 & -2 & -1 \\ 1 & 2 & 1 & 2 & -1 & -2 & -1 & -2 & -1 & -2 & -1 & -2 & 0 & 0 & 3 & 6 & -1 & -2 & -1 & -2 \\ 1 & 1 & 2 & 2 & -1 & -1 & -2 & -2 & -1 & -1 & -2 & -2 & -1 & -1 & -2 & -2 & 0 & 1 & 2 & 6 \end{array} \right)$$

and  $e = (e_1, e_2, \dots, e_{20})$  is an unknown «noise» vector with elements from the set  $\{-1, 0, +1\}$ .

Let  $c = (4, 2, 15, 11, 7, 4, 9, 5, 7, 2, 9, 4, 2, 14, 14, 13, 0, 8, 4, 12)$  and  $\sum_{i=1}^{20} e_i^2 = 14$ , provide the most efficient way to restore the initial message  $m$  (or any possible candidates for it).

**Remark.** The points for the solutions obtained via brute force or any computer algebra systems will be reduced.





# Problem 11. «Unbalanced compression»

Michelle developed a new cryptographic hash function based on the Merkle-Damgård construction. A message is split into 224-bit blocks, while a padding scheme is applied to make sure the splitting occurs regardless the message's length. A compression function takes a message block as an input and results in a 128-bit output.

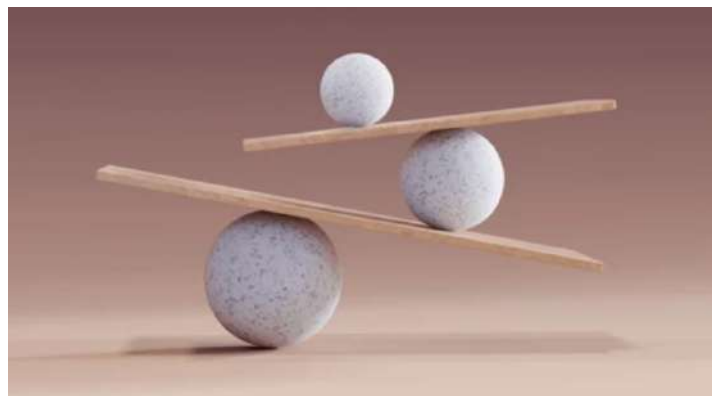
Algorithm 1 (see below) presents the compression function when it processes the first message block  $M$ . The block is divided into seven 32-bit integers  $M_0, \dots, M_6$ . On the first message block, 32-bit integers  $A, B, C, D$  are initialized with constants and then during 40 rounds they are updated. Finally, the output is formed as a concatenation of  $A, B, C, D$ . The function  $[txyz e s]_F$  stands for  $t = (t + F(x, y, z) + e) \lll s$ , where " $\lll s$ " is the circular shifting to the left by  $s$  bits position, while "+" is the addition modulo  $2^{32}$ . The functions  $[txyz e s]_G$  and  $[txyz e s]_H$  stand for  $t = (t + G(x, y, z) + e + 0x5a827999) \lll s$  and  $t = (t + H(x, y, z) + e + 0x6ed9eba1) \lll s$ , respectively.

Also let  $F(x, y, z) = (x \wedge y) \vee (\neg x \wedge z)$ ;  $G(x, y, z) = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$ ;  $H(x, y, z) = x \oplus y \oplus z$ .

In the pseudocode, " $\leftarrow$ " stands for assignment, while " $=$ " stands for equality.

**For example**, " $[DABC M_4 7]_F; D = K$ " means that  $D$  is updated by  $D \leftarrow (D + F(A, B, C) + M_4) \lll 7$  and then it turns out that the updated  $D$  is equal to  $K$ , so  $K = (D + F(A, B, C) + M_4) \lll 7$ .

Michelle managed to find all preimages for several compression function's outputs using algebraic cryptanalysis. It turned out there are usually 0, 1, or 2 preimages in total for an output, i.e. there are outputs with no preimages. However, Michelle expected about  $2^{96}$  preimages for each output because the compression function maps  $2^{224}$  onto  $2^{128}$ . Please, help Michelle to find out why the compression function is unbalanced.



**Algorithm 1** A compression function on the first 224-bit message block.

**Input:** 224-bit message block  $M$ .

**Output:** 128-bit output  $out$ .

$AA \leftarrow A \leftarrow 0x67452301; BB \leftarrow B \leftarrow 0xefcdab89$

$CC \leftarrow C \leftarrow 0x98badcfe; DD \leftarrow D \leftarrow 0x10325476$

$K \leftarrow 0xffffffff; P \leftarrow 0xa57d8668$

$[ABCD P 3]_F [DABC P 7]_F [CDAB P 11]_F [BCDA M_0 19]_F$  ▷ Rounds 1-4

$[ABCD P 3]_F [DABC P 7]_F [CDAB P 11]_F [BCDA M_1 19]_F$

$[ABCD P 3]_F [DABC P 7]_F [CDAB P 11]_F [BCDA M_2 19]_F$

$[ABCD M_3 3]_F; A = K$  ▷ The updated A equals K

$[DABC M_4 7]_F; D = K$  ▷ The updated D equals K

$[CDAB M_5 11]_F; C = K$  ▷ The updated C equals K

$[BCDA M_6 19]_F$

$[ABCD P 3]_G; A = K$  ▷ The updated A equals K

$[DABC P 5]_G; D = K$  ▷ The updated D equals K

$[CDAB P 9]_G; C = K$  ▷ The updated C equals K

$[BCDA M_3 13]_G$

$[ABCD P 3]_G; A = K$  ▷ The updated A equals K

$[DABC P 5]_G; D = K$  ▷ The updated D equals K

$[CDAB P 9]_G; C = K$  ▷ The updated C equals K

$[BCDA M_4 13]_G$

$[ABCD P 3]_G; A = K$  ▷ The updated A equals K

$[DABC P 5]_G; D = K$  ▷ The updated D equals K

$[CDAB P 9]_G; C = K$  ▷ The updated C equals K

$[BCDA M_5 13]_G$  ▷ Round 28

$[ABCD M_0 3]_G [DABC M_1 5]_G [CDAB M_2 9]_G [BCDA M_6 13]_G$

$[ABCD P 3]_H [DABC P 9]_H [CDAB P 11]_H [BCDA M_3 15]_H$

$[ABCD P 3]_H [DABC P 9]_H [CDAB P 11]_H [BCDA M_5 15]_H$

$A \leftarrow A + AA; B \leftarrow B + BB$

$C \leftarrow C + CC; D \leftarrow D + DD$

$out \leftarrow concatenation(A, B, C, D)$