Problem 1. «Algebraic immunity»

Special Prize from the Program Committee!

A mapping F from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$ is called a *vectorial Boolean function* (recall that $\mathbb{F}_2^n$ is the vector space of all binary vectors of length n). If $m = 1$ then F is a *Boolean function* in n variables. A *component function* F_v of F is a Boolean function defined by a vector $v \in \mathbb{F}_2^m$ as follows $F_v = \langle v, F\rangle = v_1 f_1 \oplus \ldots \oplus v_m f_m$, where $f_1, \ldots, f_m$ are coordinate functions of F. A function F has its unique *algebraic normal form* (ANF)

$$F(x) = \bigoplus_{I \in \mathcal{P}(N)} a_I \Big(\prod_{i \in I} x_i\Big),$$

where $\mathcal{P}(N)$ is the power set of $N = \{1, \ldots, n\}$ and a_I belongs to $\mathbb{F}_2^m$. Here $\oplus$ denotes the coordinate-wise sum of vectors modulo 2. The *algebraic degree* of F is the degree of its ANF: $\deg(F) = \max\{|I| : \ a_I \neq (0, \ldots, 0), \ I \in \mathcal{P}(N)\}$.

Algebraic immunity $AI(f)$ of a Boolean function f is the minimal algebraic degree of a Boolean function g, $g \not\equiv 0$, such that $fg \equiv 0$ or $(f \oplus 1)g \equiv 0$. The notion was introduced by W. Meier, E. Pasalic, C. Carlet in 2004.

The tight upper bound of $AI(f)$. It is wellknown that $AI(f) \leqslant \lceil \frac{n}{2} \rceil$, where $\lceil x \rceil$ is the ceiling function of number x. There exist functions with $AI(f) = \lceil \frac{n}{2} \rceil$ for any n.

Component algebraic immunity $AI_{comp}(F)$ of a function from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$ is defined as the minimal algebraic immunity of its component functions F_v, $v \neq (0, \ldots, 0)$. Component algebraic immunity was considered by C. Carlet in 2009. It is easy to see that $AI_{comp}(F)$ is also upper bounded by $\lceil \frac{n}{2} \rceil$.

The problem. What is the tight upper bound of component algebraic immunity? For all possible combination of n and m, $n, m \leqslant 4$, vectorial Boolean functions with $AI_{comp}(F) = \lceil \frac{n}{2} \rceil$ exist.

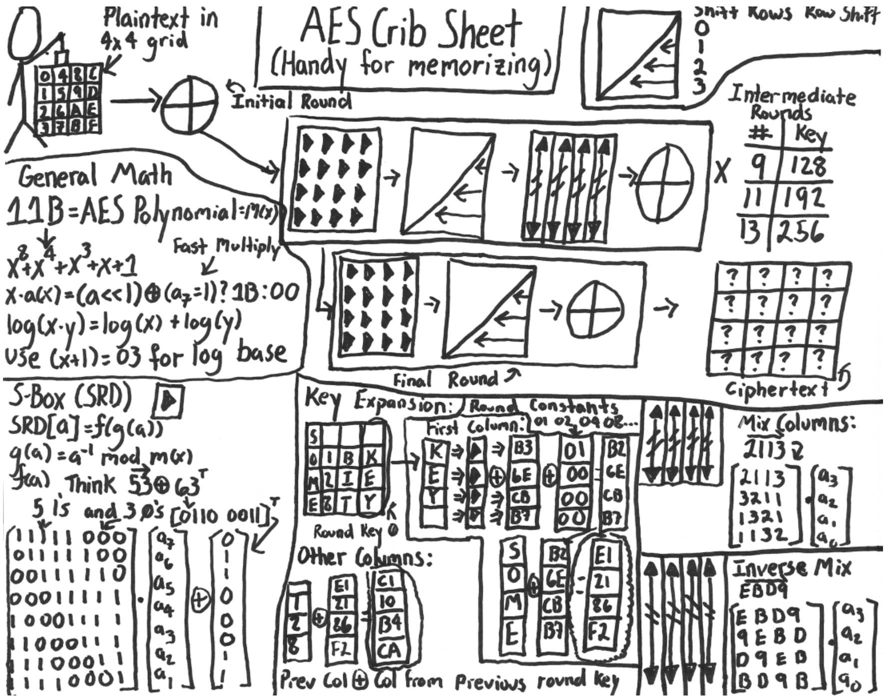
Construct $F : \mathbb{F}_2^5 \to \mathbb{F}_2^5$ with maximum possible algebraic component immunity 3 or prove that it does not exist.

Problem 2. «Zerosum at AES»

Let $\mathtt{AES}_0$ be a mapping that represents the algorithm AES-256 with the all-zero key. Let $X_1, \ldots, X_{128} \in \mathbb{F}_2^{128}$ be pairwise different vectors such that

$$\bigoplus_{i=1}^{128} X_i = \bigoplus_{i=1}^{128} \mathtt{AES}_0(X_i).$$

1. Propose an effective algorithm to find an example of such vectors $X_1, \ldots, X_{128}$.
2. Provide an example of $X_1, \ldots, X_{128}$.



The picture is from the page of Jeff Moser.

Problem 3. «Latin square»

Alice has registered on Bob's server. During the registration Alice got the secret key that is represented as a latin square of order 10. A latin square is a 10×10 matrix filled with integers $0, 1, \ldots, 9$, each occurring exactly once in each row and exactly once in each column.

To get an access to Bob's resources Alice authenticates by the following algorithm:

1. Bob sends to Alice a decimal number $abcd$, where $a, b, c, d \in \{0, 1, \ldots, 9\}$ and $a \neq b$, $b \neq c$, $c \neq d$.
2. Alice performs three actions.
   - At first she finds the integer t_1 standing at the intersection of the row $(a + 1)$ and the column $(b + 1)$.
   - Then she finds t_2 standing at the intersection of the row $(t_1 + 1)$ and the column $(c + 1)$.
   - Finally, Alice finds t_3 standing at the intersection of the row $(t_2 + 1)$ and the column $(d + 1)$.
3. Alice sends to Bob the integer t_3.
4. Bob performs the same actions and verifies Alice's answer.
5. Steps 1-4 are repeated several times. In case of success Bob recognizes that Alice knows the secret latin square.

Find Alice's secret key if you can get the answer t_3 for any your correct input request $abcd$ here.

Problem 4. «*nsucoin*»

Alice, Bob, Caroline and Daniel are using a digital payment system ***nsucoin*** to buy from each other different sorts of flowers. Alice sells only chamomiles, Bob — only tulips, Caroline — only gerberas and Daniel — only roses. At the beginning each person has 5 flowers. The cost of each flower is 2 coins.

Transactions are used to make purchases by transferring coins in the system *nsucoin*. Each transaction involves two different users (the seller A and the buyer B) and distributes a certain amount of coins S between A and B, say $S = S_A + S_B$. The value S is equal to the sum of all the coins received by the buyer in the indicated k transactions, $1 \leqslant k \leqslant 2$. We will say that the current transaction is *based* on these k transactions. The value S_A is the amount of coins that the buyer pays the seller for his product, $S_A > 0$; the value S_B is the rest of available amount of coins S that returns to buyer (in further transactions B can spend these coins). At the same time, coins received by users in each transaction can not be distributed more than once in other transactions.

In order for transactions to be valid they must be verified. To do this **block chain** is used. Each block verifies from 1 to 4 transactions. Each transaction to be verified can be based on already verified transactions and transactions based on verified transactions.

There are 4 *special* transactions. Each of them brings 10 coins to one user. These transactions do not based on other transactions. The first block verifies all special transactions.

Define what bouquet Alice can make from the flowers she has if the last block in chain is the following string (hash of this block in `00004558`):

```
height:2;prevHash:0000593b;ctxHash:8fef76cb;nonce:17052
```

Turn to the next page.

Technical description of *nsucoin*.

- **Transactions.** Transaction is given by the string `transaction` of the following format:

```
transaction = "txHash:{hashValue};{transactionInfo}"
hashValue = Hash({transactionInfo})
transactionInfo = "inputTx:{Tx};{sellerInfo};{buyerInfo}"
Tx = "{Tx1}" or "{Tx1,Tx2}"
sellerInfo = "value1:{V1};pubKey1:{PK1};sign1:{S1}"
buyerInfo = "value2:{V2};pubKey2:{PK2};sign2:{S2}"
```

Here `Tx1, Tx2` are values of the field `txHash` of transactions which the current transaction based on. `V`i is a non-negative integer that is equal to the amount of coins received by the user with public key `PK`i, $0 \leqslant$ `V`$i \leqslant 10$, `V1`$\neq 0$. Digital signature

$$\texttt{S}i = \texttt{DecToHexStr(Signature(Key2,StrToByteDec(Hash(Tx1+Tx2+PK}i\texttt{))))},$$

where `+` is concatenation operation of strings. `Key2` is private key of buyer.

In the special transactions fields `inputTx`, `sign1` are empty and there is no `buyerInfo`. For example, one of the special transactions is the following:

`txHash:1a497b59;inputTx:;value1:10;pubKey1:11;sign1:`

- **Block chain.** Each block is given by the string `block` of the following format:

`block = "height:{Height};prevHash:{PrHash};ctxHash:{CTxHash};nonce:{Nonce}"`

Here `Height` is the block number in a chain, the first block has number 0. `PrHash` is hash of block with number `Height` -1. `CTxHash` is hash of concatenation of all the `TxHash` of transactions verified by this block. `Nonce` is the minimal number from 0 to 40000 such that block has hash of the form `0000####`.

Let `PrHash` = `00000000` for the first block.

- **Hash function.** `Hash` is calculated as reduced MD5: the result of hashing is the first 4 bytes of standard MD5 represented as a string. For example, `Hash`("`teststring`") = "`d67c5cbf`", `Hash`("`1a497b5917`") = "`e0b9e4a8`".

- **Digital signature.** `Signature`$(key, message)$ is RSA digital signature with n of order 64 bits, $n = 9101050456842973679$. Public exponents `PK` of users are the following:

User	Alice	Bob	Caroline	Daniel
PK	11	17	199	5

For example, `Signature`(2482104668331363539, 7291435795363422520) = 7538508415239841520.

- **Additional functions.** `StrToByteDec` decodes a string to bytes that are considered as a number. Given a number `DecToHexStr` returns a string that is equal to the hexadecimal representation of this number. For example, `StrToByteDec`("`e0b9e4a8`") = 7291435795363422520 and `DecToHexStr`(7538508415239841520) = "`689e297682a9e6f0`".

Strings are given in UTF-8.

Turn to the next page.

Examples of a transaction and a block.

• Suppose that Alice are buying from Bob 2 tulips. So, she must pay him 4 coins. The transaction of this operation, provided that Alice gets 10 coin in the transaction with hash `1a497b59`, is

```
txHash:98e93fd5;inputTx:1a497b59;value1:4;pubKey1:17;sign1:689e297682a9e6f0;
value2:6;pubKey2:11;sign2:fec9245898b829c
```

• The block on height 2 verifies transactions with hash values (values of `txHash`) `98e93fd5`, `c16d8b22`, `b782c145` and `e1e2c554`, provided that hash of the block on height 1 is `00003cc3`, is the following:

```
height:2;prevHash:00003cc3;ctxHash:9f8333d4;nonce:25181
```

Hash of this block is `0000642a`.

Problem 5. «Metrical cryptosystem»

Alice and Bob exchange messages using the following cryptosystem. Let $\mathbb{F}_2^n$ be an n-dimensional vector space over the field $\mathbb{F}_2 = \{0, 1\}$. Alice has a set $A \subseteq \mathbb{F}_2^n$ and Bob has a set $B \subseteq \mathbb{F}_2^n$ such that both A and B are metrically regular sets and they are metrical complements of each other. Let d be the Hamming distance between A and B. To send some number a $(0 \leqslant a \leqslant d)$ Alice chooses some vector $x \in \mathbb{F}_2^n$ at distance a from the set A and sends this vector to Bob. To obtain the number that Alice has sent Bob calculates the distance b from x to the set B and concludes that the initial number a is equal to $d - b$.

Is this cryptosystem correct? In other words, does Bob correctly decrypt all sent messages, regardless of initial sets A, B satisfying given conditions and of the choice of vector x?

Remark I. Recall several definitions and notions. The *Hamming distance* $d(x, y)$ between vectors x and y is the number of coordinates in which these vectors differ. Distance from vector $y \in \mathbb{F}_2^n$ to the set $X \subseteq \mathbb{F}_2^n$ is defined as $d(y, X) = \min_{x \in X} d(y, x)$. The *metrical complement* of a set $X \subseteq \mathbb{F}_2^n$ (denoted by $\widehat{X}$) is the set of all vectors $y \in \mathbb{F}_2^n$ at maximum possible distance from X (this maximum distance is also known as *covering radius* of a set). A set $X \subseteq \mathbb{F}_2^n$ is called *metrically regular*, if its second metrical complement $\widehat{\widehat{X}}$ coincides with X.

Remark II. Let us consider several examples:

- Let X consist of a single vector $x \in \mathbb{F}_2^n$. It is easy to see that $\widehat{X} = \{x \oplus \mathbf{1}\}$, where $\mathbf{1}$ is the all-ones vector, and therefore $\widehat{\widehat{X}} = \{x \oplus \mathbf{1} \oplus \mathbf{1}\} = \{x\} = X$, so X is a metrically regular set; it is also easy to see that cryptosystem based on $A = \{x\}$, $B = \{x \oplus \mathbf{1}\}$ is correct;
- Let Y be a ball of radius $r > 0$ centered at x: $Y = B(r, x) = \{y \in \mathbb{F}_2^n : d(x, y) \leqslant r\}$. You can verify that $\widehat{Y} = \{x \oplus \mathbf{1}\}$, but $\widehat{\widehat{Y}} = \{x\} \neq Y$, and Y is not metrically regular;
- Let X be an arbitrary subset of $\mathbb{F}_2^n$. Then, if we denote $X_0 := X$, $X_{k+1} = \widehat{X_k}$ for $k \geqslant 0$, there exists a number M such that X_m is a metrically regular set for all $m > M$. You can prove this fact as a small exercise, or simply use it in your solution.

Problem 6. «Quadratic functions»

Alice and Bob are going to use the following pseudorandom binary sequence $u = \{u_i\}$, $u_i \in \mathbb{F}_2$:

- $u_1, \ldots, u_n$ are initial values;
- $u_{i+n} = f(u_i, u_{i+1}, \ldots, u_{i+n-1})$, where

$$f \in Q_n = \{a_0 \oplus \bigoplus_{i=1}^{n} a_i x_i \oplus \bigoplus_{1 \leqslant i < j \leqslant n} a_{ij} x_i x_j \mid a_0, a_i, a_{ij} \in \mathbb{F}_2\}.$$

Suppose that you have intercepted the elements $u_t, u_{t+1}, \ldots, u_{t+k-1}$ of a sequence for some t. Is it possible to uniquely reconstruct the elements

$$u_{t+k}, u_{t+k+1}, u_{t+k+2}, \ldots$$

provided $k \leqslant cn$, where c is a constant independent on n?

Problem 7. «Secret sharing»

Alena, Boris and Sergey developed the following secret sharing scheme to share a password $P \in \mathbb{F}_2^{32}$ into three parts to collectively manage money through online banking.

- Vectors $v_i^a, v_i^b, v_i^s \in \mathbb{F}_2^{32}$ and values $c_i^a, c_i^b, c_i^s \in \mathbb{F}_2$ are randomly generated for all $i = 1, \ldots, 32$.
- Vectors v_i^a, v_i^b, v_i^s are known to all participants of the scheme.
- Values $c_i^a, c_i^b, c_i^s \in \mathbb{F}_2$ are known only to Alena, Boris and Sergey respectively.
- Then the secret password P is calculated by the rule

$$P = \bigoplus_{i=1}^{32} c_i^a v_i^a \oplus \bigoplus_{i=1}^{32} c_i^b v_i^b \oplus \bigoplus_{i=1}^{32} c_i^s v_i^s.$$

What is the probability that Alena and Boris together can not get any information about the password P? What is the probability that they are able without Sergey to get a guaranteed access to online banking using not more than 23 attempts?

Problem 8. «Biometric key»

Iris is one of the most reliable biometric characteristics of a human. While measuring let us take 128-bit biometric image of an iris. As in reality, we suppose that two 128-bit biometric images of *the same human* can differ not more than by 10–20%, while biometric images of *different people* have differences at least 40–60%.

Let a key k be an arbitrary 8-bit vector. It can be represented in hexadecimal notation. For example, $\texttt{e2} = 11100010$. We suppose that the key is a pin-code that should be used in order to get access to the bank account of a client.

To avoid situation when malefactor can steal the key of a some client and then be able to get an access to his account, the bank decided to combine usage of the key with biometric authentication of a client by iris-code. The following scheme of covering the key with biometric data was proposed:

1) on registration of a client take 128-bit biometric image $b_{template}$ of his iris;

2) extend 8-bit key k to 128-bit string s using Hadamard encoding, i. e. if $k = (k_1, \ldots, k_8)$, where $k_i \in \mathbb{F}_2$, then s is the vector of values of the Boolean function $f(x_1, \ldots, x_7) = k_1x_1 \oplus \ldots \oplus k_7x_7 \oplus k_8$, where $\oplus$ is summing modulo 2;

3) save the vector $c = b_{template} \oplus s$ on the smart-card and give it to the client. A vector c is called *biometrically encrypted key*.

To get an access to his account a client should

1) take a new 128-bit biometric image b of his iris;

2) using information from the smart-card count 128-bit vector s' as $s' = b \oplus c$;

3) decode s' to 8-bit vector k' using Hadamard decoding procedure.

Then the bank system checks: if $k' = k$ then the client is authenticated and the key is correct; hence bank provides an access to the account of this client. Otherwise, if $k' \neq k$ then bank signals about an attempt to get illegal access to the bank account.

The problem. One day a person, say X, came to the bank and tried to get an access to the bank account of Alice using the smart-card. This may be noticed that person X was in hurry and may be a little bit nervous. Suddenly, another person, say Y, appeared in the bank and declared loudly: "Please stop any operation! I am Alice! My smart-card was stolen."

Bank clerk, say Claude, stopped all operations. In order to solve the situation he took new biometric images b^X and b^Y of persons X and Y respectively, and with smart-card containing vector c leaved his post for consultations with bank specialists.

When Claude came back, he already knew who was Alice. He wanted to stop the other person and call to police but that person has already disappeared. So, can you solve this problem too? Who was real Alice? Determine her 8-bit key k. You can use the data b^X, b^Y and c presented on the picture. It is known also that the key of Alice contains odd number of ones.

Remark. The vector of values of a Boolean function f in n variables is a binary vector $(f(x^0), f(x^1), \ldots, f(x^{2^n-1}))$ of length 2^n, where $x^0 = (0, \ldots, 0, 0)$, $x^1 = (0, \ldots, 0, 1)$, $\ldots$, $x^{2^n-1} = (1, \ldots, 1, 1)$, ordered by lexicographical order; for example, the vector of values of the function $f(x_1, x_2) = x_1 \oplus x_2 \oplus 1$ is equal to $(1001) = \mathtt{9}$. The vector of values of the function $f(x_1, \ldots, x_7) = x_1 \oplus x_2 \oplus 1$ is `ffff ffff 0000 0000 0000 0000 ffff ffff`.

Problem 9. «Protocol»

Alena and Boris developed a new protocol for establishing session keys. It consists of the following three steps:

1. The system has a common prime modulus p and a generator g. Alena and Boris have their own private keys $\alpha_a \in \mathbb{Z}_{p-1}$, $\alpha_b \in \mathbb{Z}_{p-1}$ and corresponding public keys $P_a = g^{\alpha_a} \mod p$, $P_b = g^{\alpha_b} \mod p$.

2. To establish a session key Alena generates a random number $R_a \in \mathbb{Z}_{p-1}$, computes $X_a = (\alpha_a + R_a) \mod (p-1)$ and sends it to Boris. Then Boris generates his random number R_b, computes X_b in the same way as Alena and sends it back to her.

3. Alena computes the session key in the following way:

$$K_{a,b} = (g^{X_b} P_b^{-1})^{R_a} \mod p.$$

Bob computes the session key in the following way:

$$K_{b,a} = (g^{X_a} P_a^{-1})^{R_b} \mod p.$$

How can an attacker Evgeniy compute any future session key between Alena and Boris, if he steals the only one session key $K_{a,b}$?

Problem 10. «Find the key»

The key of a cipher is the set of positive integers a, b, c, d, e, f, g, such that the following relation holds:

$$a^3 + b^3 + c^3 + d^3 + e^3 + f^3 + g^3 = 2016^{2017}.$$

Find the key!

Problem 11. «Labyrinth»

Read the message hidden in the labyrinth!

Problem 12. «Big Fermat numbers»

Special Prize from the Program Committee!

It is known that constructing big prime numbers is very actual and complicated problem interesting for cryptographic applications. One of the popular way to find them is... to guess! For example to guess them between numbers of some special form. For checking there are Mersenne numbers $2^k - 1$, Fermat numbers $F_k = 2^{2^k} + 1$ for nonnegative integer k, etc.

Let us concentrate our attention on Fermat's numbers.

It is known that Fermat numbers $F_0 = 3$, $F_1 = 5$, $F_2 = 17$, $F_3 = 257$, $F_4 = 65537$ are prime. But the number $F_5 = 4\,284\,967\,297 = 641 \cdot 6\,700\,417$ is already composite as was proven by L. Euler in XVIII.

For now it is known that all Fermat numbers, where $k = 5, \ldots, 32$, are composite and there is the hypothesis that every Fermat number F_k, where $k \geqslant 5$ is composite.

Could you prove that for any big number N there exists a composite Fermat number F_k such that $F_K > N$?

381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400	401
380	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	402
379	306	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	326	403
378	305	240	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	258	327	404
377	304	239	182	133	134	135	136	137	138	139	140	141	142	143	144	145	198	259	328	405
376	303	238	181	132	91	92	93	94	95	96	97	98	99	100	101	146	199	260	329	406
375	302	237	180	131	90	57	58	59	60	61	62	63	64	65	102	147	200	261	330	407
374	301	236	179	130	89	56	31	32	33	34	35	36	37	66	103	148	201	262	331	408
373	300	235	178	129	88	55	30	13	14	15	16	17	38	67	104	149	202	263	332	409
372	299	234	177	128	87	54	29	12	3	4	5	18	39	68	105	150	203	264	333	410
371	298	233	176	127	86	53	28	11	2	1	6	19	40	69	106	151	204	265	334	411
370	297	232	175	126	85	52	27	10	9	8	7	20	41	70	107	152	205	266	335	412
369	296	231	174	125	84	51	26	25	24	23	22	21	42	71	108	153	206	267	336	413
368	295	230	173	124	83	50	49	48	47	46	45	44	43	72	109	154	207	268	337	414
367	294	229	172	123	82	81	80	79	78	77	76	75	74	73	110	155	208	269	338	415
366	293	228	171	122	121	120	119	118	117	116	115	114	113	112	111	156	209	270	339	416
365	292	227	170	169	168	167	166	165	164	163	162	161	160	159	158	157	210	271	340	417
364	291	226	225	224	223	222	221	220	219	218	217	216	215	214	213	212	211	272	341	418
363	290	289	288	287	286	285	284	283	282	281	280	279	278	277	276	275	274	273	342	419
362	361	360	359	358	357	356	355	354	353	352	351	350	349	348	347	346	345	344	343	420
441	440	439	438	437	436	435	434	433	432	431	430	429	428	427	426	425	424	423	422	421