INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2020, VOL. 66, NO. 4, PP. 647-653
Manuscript received August 18, 2020; revised October, 2020. DOI: 10.24425/ijet.2020.134023

Differential Cryptanalysis of New Qamal Encryption Algorithm

Kunbolat T. Algazy Ludmila K. Babenko Rustem G. Biyashev Evgeniya A. Ishchukova, Ryszard Romaniuk, Nursulu A. Kapalova, Saule E. Nysynbaeva, and Andrzej Smolarz

***Abstract*—Currently, the Republic of Kazakhstan is developing a new standard for symmetric data encryption. One of the candidates for the role of the standard is the Qamal encryption algorithm developed by the Institute of Information and Computer Technologies (Almaty, Republic of Kazakhstan). The article describes the algorithm. Differential properties of the main operations that make up the Qamal cypher are considered in the questions of stability. We have shown that for a version with a 128-bit data block and the same secret key size for three rounds of encryption it is difficult to find the right pairs of texts with a probability of 2^{-120}, which makes differential cryptanalysis not applicable to the Qamal cypher.**

***Keywords*—cryptography, block cypher, difference, differential cryptanalysis, probability**

I. Introduction

THE first of the well-known government standards for data encryption was the DES standard adopted in the United States in the early 1970s. It was the time when the first computers (electronic computers) gradually ceased to be exotic and began to enter the life and work of small firms and research laboratories. This led to the fact that the problem of data protection, stored and processed on them, was recognized by a growing number of specialists. Many large corporations, not to mention public services, have conducted their own research in this area. As a result, various encryption algorithms began to appear. One of the most famous research centres of this kind at that time was the IBM science laboratory, headed by Dr Horst Feistel [1]. As a result, a system of encryption called Lucifer was created. For this encryption system, Horst Feistel proposed a mathematical model, which is now called the "Feistel scheme". The principle of the Feistel scheme is that only half or part of the text is encrypted in one round. A block of text is divided into parts. One part goes through some mathematical transformation. And the result of this transformation is added up by modulo two with the second part of the text. After that, the parts of the text are swapped. Another advantage of the scheme was the fact that by using the "Exclusive-OR" operation or, as it is also called the modulo-two addition operation, it becomes possible to use the same scheme for both data encryption and data decryption, it is enough just to change the order of the round subkeys. Initially, the DES standard was adopted for a period of 5 years, but later it was repeatedly extended as a standard [2]. By the end of the 20th century, computers were already widespread and computing power increased significantly. Therefore, the U.S. government has thought about changing the standard. As a result, a tender was announced for the adoption of a new data encryption standard – the AES (Advanced Encryption Standard) competition. The competition was announced in 1997 by the National Institute of Standards and Technologies (NIST) [3]. Fifteen encryption algorithms created by scientists from different countries were announced for participation in the contest. As a result of a five-year study, the Rijndael encryption algorithm developed by two mathematicians from Belgium, Vincent Rijmen (V. Rijmen) and Joan Damen, was chosen as the new US standard. The Rijndael algorithm is built on a network scheme based on substitutions and permutations (SPN) and has the architecture of "Square". At that time, the "Square" architecture and the SP-network were an innovative solution. Now many algorithms are AES-like and follow the structure of the Rijndael cypher.

In parallel with the AES competition in January 2000, a very similar competition began in Europe, involving the selection of cryptographic standards of the European Union. This competition was called NESSIE (New European Schemes for Signature, Integrity and Encryption) [3]. As a result of the work on the NESSIE competition, a great work entitled "NESSIE security report" [3] was written by scientists-cryptographers, but the European standard was never chosen.

Under the influence of the US and European sentiment, the CRYPTREC project was created in Japan. CRYPTREC is an acronym from the Cryptography Research and Evaluation Committee [4]. The project was created to study cryptographic algorithms and then recommend specific algorithms for use in public and private organizations. As a result of the CRYPTREC project, a number of recommended encryption algorithms have been identified. CIPHERUNICORN-E, Hierocrypt-L1, MISTY1 and a three-key version of the Triple DES algorithm were recommended for 64-bit ciphers. For 128-bit: AES, Camellia, CIPHERUNICORN-A, Hierocrypt-3, SC2000.

This work was supported by a targeted funding program "Development of software and hardware and software for cryptographic protection of information during its transmission and storage in infocommunication systems and general purpose networks" from the Committee of Science of the Ministry of Education and Science of the Republic of Kazakhstan (registration number 0118PK01064).

Kunbolat Algazy (e-mail: kunbolat@mail.ru), Rustem Biyashev (e-mail: brg@ipic.kz), Nursulu Kapalova (e-mail: kapalova@ipic.kz), Saule Nysynbaeva (e-mail: sultasha1@mail.ru) are with Institute of Information and Computational Technologies of the Committee of Science of the Ministry of Education and Science of the Republic of Kazakhstan, Almaty;

Ludmila Babenko (e-mail: lkbabenko@sfedu.ru). Evgeniya Ishchukova (e-mail: uaishukova@sfedu.ru) are with Institute of Computer Technologies and Information Security of the Southern Federal University, Taganrog, Russia

Ryszard Romaniuk (e-mail: r.romaniuk@ise.pw.edu.pl) is with Warsaw University of Technology, Poland.

Andrzej Smolarz (e-mail: a.smolarz@pollub.pl) is with Lublin University of Technology, Lublin, Poland.

In Russia, the standard of symmetric encryption GOST 28147-89 was adopted in 1989. However, until 1994 it remained classified. GOST 28147-89 is a 64-bit block cipher built according to the Feistel scheme. Developers have put in the cipher excessive resistance due to a large number of rounds of encryption (32 rounds) and the imposition of a secret key using the operation of addition on modulo 2^{32}. On January 1, 2016, a new data encryption standard was adopted in Russia – GOST R34.12-2015 [6]. The new encryption standard includes two encryption algorithms: Magma and Kuznechik. Magma is a former GOST 28147-89 standard with one exception. In the GOST 28147-89 standard, S-blocks were not fixed and could be selected randomly. In the Magma S-blocks algorithm are regulated by the standard. Kuznechik algorithm is a 128-bit symmetric block cipher, built on the principle of SP-network.

The states that used to make up the USSR inherited the GOST 28147-89 standard of encryption. At the moment, there is a tendency for these countries to develop their own national security systems, which, among other things, include the development of their own data encryption standards. Thus, in Belarus, the STB 34.101.31-2007 standard “Information technologies and security. Cryptographic encryption and integrity control algorithms” was developed [7]. First, in 2007, the standard STB 34.101.31-2007 was adopted as a preliminary standard. In 2011, STB 34.101.31-2007 was enacted as the final standard. In July 2015, the symmetric encryption standard was adopted in Ukraine [8]. Standard DSTU 7624: 2014 describes the operation of Kalina encryption algorithm, which is an AES-like encryption algorithm.

The Republic of Kazakhstan is also currently working on the creation of a state standard for symmetric data encryption within the framework of the project "Development of software and hardware for cryptographic protection of information during its transmission and storage in info-communication systems and general-purpose networks" from the Committee of Science of the Ministry of Education and Science of the Republic of Kazakhstan [9]. One of the design algorithms of encryption is the Qamal cipher, proposed for research in this paper. In the first part of the work (chapter II) we describe the Qamal project cipher and provide examples of the implementation of the basic cipher transformations that can be used as control points when performing a software cipher implementation. The following part (chapter III) is dedicated to the discussion of differential properties of all the basic operations that are part of the Qamal algorithm. Next chapter describes the technology of building multi-round characteristics and evaluates the stability of the Qamal cipher according to the differential cryptanalysis methods.

II. QAMAL ENCRYPTION ALGORITHM

A. Data encryption using Qamal's encryption algorithm

The Qamal encryption algorithm is a symmetrical block encryption algorithm, built on the principle of SP-network. The algorithm supports block and key lengths of 128, 192 and 256 bits, while the length of the processed block of data and the length of the secret key must always match. The number of encryption rounds depends on the length of the block and the key. The 128, 192 and 256 bit K keys correspond to eight, ten and twelve rounds of encryption, respectively. All rounds except the last are identical. In the last round, an additional round key is added. The scheme of the encryption algorithm is shown in Figure 1 [10].

The encryption algorithm includes developed key imposition procedures using the bitwise addition (XOR) operation, the S-block replacement, the mixing procedures Mixer1 and Mixer2.

In the first procedure, a key modulo 2 operation (XOR operation) is performed on the block of plaintext.

The second procedure is to replace the bytes using the S-box replacement. For this, a nonlinear conversion of bytes is performed: a nonlinear bijective substitution is applied to each byte. The resulting S-box is presented in Table I. In Table I, all data is given in hexadecimal. Using an S-block is similar to using an S-block for the AES data encryption standard: the original byte is divided into two halves, the top 4 bits (left side of the byte) indicate the row number in the replacement table, and the bottom 4 bits (right side of the byte) indicate the column number in the replacement table.

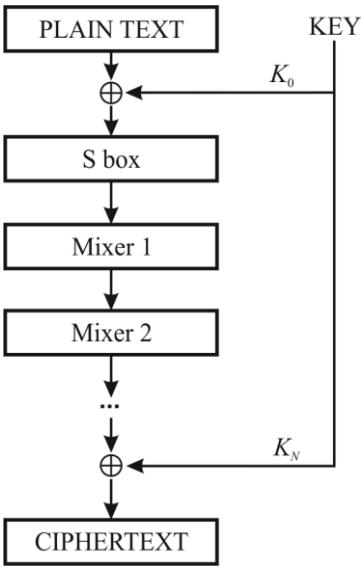

Fig.1. Qamal Encryption Algorithm Scheme

TABLE I
S1-BLOCK

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	C9	34	F0	18	55	86	21	6B	87	D2	6E	99	BD	31	98	89
1	29	73	83	8B	1A	19	E1	E4	F3	5B	72	3F	A6	F9	2E	A3
2	7E	10	94	07	EC	AD	2F	26	20	93	66	3D	DD	64	5F	C1
3	13	E0	80	25	D3	08	75	6A	B9	2D	D1	CC	FD	CA	3B	FC
4	D5	DA	E2	CE	A0	7F	AE	C8	9C	09	3C	95	BA	35	3E	7B
5	FA	8D	23	AB	D9	E8	74	2A	C3	A8	D8	52	45	B5	0A	0C
6	A4	61	9A	FB	AA	F6	78	84	C4	E9	EE	54	50	81	DF	90
7	36	B4	BB	44	C5	96	4B	28	14	E6	8F	FF	B0	1F	53	47
8	00	4C	40	2C	9B	9F	4A	01	7D	AF	92	56	7A	DB	8E	16
9	63	24	A9	1D	33	4D	E7	1C	70	69	B7	C6	32	E5	57	03
A	97	A5	EB	D4	BC	5D	F8	85	06	F2	59	F4	17	22	38	DC
B	0B	FE	BE	CD	41	82	04	0E	48	71	30	AC	EF	C7	2B	CB
C	B8	8C	5A	42	A7	4E	D0	46	BF	B3	91	E3	11	7C	6F	DE
D	88	58	1E	5C	9D	60	C0	62	05	79	ED	76	C2	02	65	D7
E	F1	8A	77	F7	37	B1	0F	67	CF	0D	A1	6C	4F	3A	39	1B
F	27	B6	5E	F5	EA	6D	15	9E	B2	12	A2	68	43	51	49	D6

The third procedure is the linear formation of the Mixer1 block. Bytes of the block are represented in the form of a two-dimensional array A with the size *m**4, where m takes on the value of 4, 6 or 8, depending on the size of the initial block:

$$\mathbf{A} = \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{10} & a_{10} & a_{10} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m0} & a_{m1} & a_{m2} & a_{m3} \end{bmatrix}.$$

Bytes of each column are added together modulo 256:

$$\mathrm{M}_1(\mathrm{b_{ij}}) = \sum_{\mathrm{i}=0}^{\mathrm{m}} \mathrm{a_{ij}} \bmod 256,\ \mathrm{j} = 0,1,2,3.$$

Then, the obtained new byte of the first column is placed in a place of the upper byte a_{00}, and the original bytes are shifted down by one position. This operation is repeated four times. As a result, we get four new bytes in the first column. Further, this operation is performed for the three remaining columns (Figure 2).

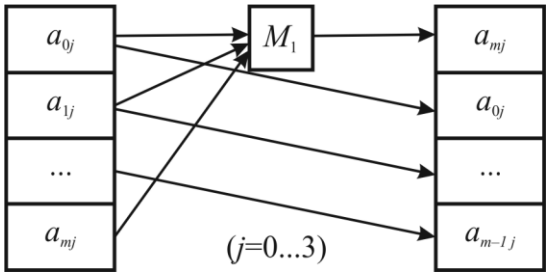

Fig. 2. Operation of the Mixer1

The formation of the Mixer1 block results in a new B array of *m**4 size, where m will be equal to 4, 6 or 8 depending on the block size (respectively *m*=4 for a block of 128 bits, *m*=6 for a block of 192 bits and *m*=8 for a block of 256 bits):

$$\mathrm{B} = \begin{bmatrix} b_{00} & b_{01} & b_{02} & b_{03} \\ b_{10} & b_{10} & b_{10} & b_{10} \\ \vdots & \vdots & \vdots & \vdots \\ b_{m0} & b_{m1} & b_{m2} & b_{m3} \end{bmatrix}.$$

Each row of the array is represented as a polynomial and a polynomial multiplication of the form $b_i(x) * m_i(x) \bmod p(x)$ occurs, where:

$p = 100000000000000000000000100110111_2 = 0x100000137;$
$m_0 = 10101000001000101011101110111010_2 = 0xA822BBBA;$
$m_1 = 11010010001101011101001001100101_2 = 0xD235D265;$
$m_2 = 11011010000110011001011011010010_2 = 0xDA1996D2;$
$m_3 = 10010000010010111001111000011011_2 = 0x904B9E1B;$
$m_4 = 10100011000001000110111101101010_2 = 0xA3046F6A;$
$m_5 = 10010110111011010000110100110101_2 = 0x96ED0D35;$
$m_6 = 01100011001110110110100011001101_2 = 0x633B68CD;$
$m_7 = 10100111001100011111000110011010_2 = 0xA731F19A.$

The $m_i(x)$ values are also presented as polynomials and are applied as follows. With an open block length of 128 bits, the first four values of $m_0(x)$, $m_1(x)$, $m_2(x)$, $m_3(x)$ are used. For the block length of 192 bits the first 6 six values of $m_0(x)$, $m_1(x)$, $m_2(x)$, $m_3(x)$, $m_4(x)$, $m_5(x)$ are taken. For the third possible block length, all eight $m_i(x)$ values are used [10].

B. Data decryption using Qamal's encryption algorithm

To decrypt the ciphertext, all cryptographic transformations used for encryption are inverted and used in the decryption algorithm in reverse order. Round keys are also used in reverse order. When decrypting each of these block lengths, 8, 10, or 12 rounds are performed, respectively, with Inv*S*, InvM_1, and InvM_2 inverse conversions performed in each block length. The Inv*S* conversion is inverse to the byte change operation through S-block using Table I. For example, if byte 00 was replaced by byte C9, byte C9 must be replaced by byte 00 for inverse conversion (see Table II).

TABLE II
INVERSE S-BLOCK

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	80	87	DD	9F	B6	D8	A8	23	35	49	5E	B0	5F	E9	B7	E6
1	21	CC	F9	30	78	F6	8F	AC	03	15	14	EF	97	93	D2	7D
2	28	06	AD	52	91	33	27	F0	77	10	57	BE	83	39	1E	26
3	BA	0D	9C	94	01	4D	70	E4	AE	EE	ED	3E	4A	2B	4E	1B
4	82	B4	C3	FC	73	5C	C7	7F	B8	FE	86	76	81	95	C5	EC
5	6C	FD	5B	7E	6B	04	8B	9E	D1	AA	C2	19	D3	A5	F2	2E
6	D5	61	D7	90	2D	DE	2A	E7	FB	99	37	07	EB	F5	0A	CE
7	98	B9	1A	11	56	36	DB	E2	66	D9	8C	4F	CD	88	20	45
8	32	6D	B5	12	67	A7	05	08	D0	0F	E1	13	C1	51	8E	7A
9	6F	CA	8A	29	22	4B	75	A0	0E	0B	62	84	48	D4	F7	85
A	44	EA	FA	1F	60	A1	1C	C4	59	92	64	53	BB	25	46	89
B	7C	E5	F8	C9	71	5D	F1	9A	C0	38	4C	72	A4	0C	B2	C8
C	D6	2F	DC	58	68	74	9B	BD	47	00	3D	BF	3B	B3	43	E8
D	C6	3A	09	34	A3	40	FF	DF	5A	54	41	8D	AF	2C	CF	6E
E	31	16	42	CB	17	9D	79	96	55	69	F4	A2	24	DA	6A	BC
F	02	E0	A9	18	AB	F3	65	E3	A6	1D	50	63	3F	3C	B1	7B

The InvM_1 transform is the reverse of the Mixer1 transform. This means that the addition operation modulo 256 must be replaced by a sequential subtraction operation modulo 256.
The conversion of InvM_2 is inverse to the procedure for obtaining a Mixer2 block. To obtain a block inverse Mixer2, each row of the array is treated as a polynomial, which is multiplied by fixed polynomials modulo $p(x)$, where:

$m_0^{-1} = 11110011\ 01001000\ 10001001\ 11010101_2 = 0xf3488ad5;$
$m_1^{-1} = 00010110\ 01110011\ 11010000\ 11010111_2 = 0x1673d0d7;$
$m_2^{-1} = 10001010\ 00101110\ 10001011\ 10111010_2 = 0x8a2e8bba;$
$m_3^{-1} = 11000000\ 10100010\ 00111100\ 10110000_2 = 0xc0a23cb0;$
$m_4^{-1} = 11111101\ 10100101\ 01100100\ 01010010_2 = 0xfda56452;$
$m_5^{-1} = 01111111\ 10011100\ 00110000\ 00100010_2 = 0x7f9c3022;$
$m_6^{-1} = 10011000\ 01001011\ 10011101\ 00111110_2 = 0x984b9d3e;$
$m_7^{-1} = 00011110\ 01110011\ 00011111\ 10001000_2 = 0x1e731f88;$

The first four values $m_0^{-1}(x)$, $m_1^{-1}(x)$, $m_2^{-1}(x)$, $m_3^{-1}(x)$ are used for an open block of 128 bits. For the block length of 192 bits the first six values of $m_0^{-1}(x)$, $m_1^{-1}(x)$, $m_2^{-1}(x)$, $m_3^{-1}(x)$, $m_4^{-1}(x)$, $m_5^{-1}(x)$ are taken. For the third possible block length, all eight $m_i^{-1}(x)$ values are used [10].

C. Round key generation algorithm

Round keys Ki are generated from the cipher key K using the key expansion procedure. As a result, an array of round keys is formed, from which the necessary round key is then directly selected. The scheme for obtaining round keys is shown in Figure 3.

The original secret key K is the first round subkey. To generate the following secret key, ten rounds of conversions are performed: the replacement with the S-box shown in Table III, the transformation Mixer1 (Figure 2) and the transformation Mixer2, which is similar to the transformation of the same name in the encryption procedure.

The last procedure of the round is called Module pi(x) and works as follows. Let $g_1(x), g_2(x), \dots, g_S(x)$ – irreducible binary polynomials used as working bases, where $G(x) = g_1(x)g_2(x) \dots g_S(x)$. The degree of the polynomial $G(x)$ corresponds to the value $N = m_1 + m_2 + \cdots + m_S$ and is equal to the block length (i.e.128, 192, 256). The output from the Mixer2 block is represented as an $N(x)$ polynomial with binary coefficients. $k_1(x)$, $k_2(x)$, ..., $k_s(x)$ – the remains of the division of the polynomial N(x) on the corresponding bases $p_i(x)$, i = 1,...,s. Where $p_i(x)$, i = 1,...,s are the secret elements of the key deployment procedure.

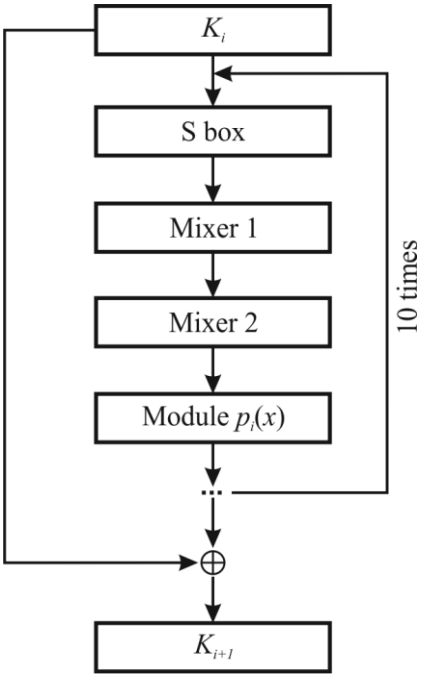

Fig. 3. Round key *Ki* expansion, where *i*=0,1,…,8 [9,13]

TABLE III.

S2-BLOCK, USED FOR ROUND KEY FORMATION

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	55	A8	78	9C	C3	ED	B1	DE	CD	2C	09	51	27	2D	43	C2
1	CA	45	3A	CE	7B	79	84	7D	BF	E6	69	1F	5E	CB	9E	E2
2	49	38	8E	7C	31	DF	98	42	91	57	90	A6	BD	F1	41	AC
3	20	96	8C	C7	4B	BE	70	E9	D0	4D	1A	A1	B0	DA	5D	D3
4	88	B5	30	47	6B	35	12	B2	B4	17	10	A2	60	9B	0D	FD
5	E4	C6	54	EB	B7	B9	7F	AF	21	5C	D4	99	5F	3E	A9	F3
6	3C	C0	67	13	6A	2F	1C	29	89	58	73	EC	14	39	D8	4E
7	44	02	59	23	F2	0C	FC	AB	74	87	92	36	82	04	16	0E
8	BB	01	F6	15	E7	DC	8F	07	4A	FF	65	1B	25	8B	75	D7
9	A5	7A	A7	FA	24	E5	AE	61	CF	9D	32	66	AA	05	D2	62
A	8D	C4	4F	26	06	0A	D9	7E	F7	E3	F0	34	40	0F	FB	1E
B	6F	A3	D1	BA	95	3D	33	71	83	18	E0	CC	2B	A0	D5	28
C	E1	64	9F	97	4C	A4	76	B3	19	08	68	C1	22	1D	B8	8A
D	E8	50	00	C9	46	56	5A	72	F5	3B	63	94	93	9A	0B	AD
E	DD	C8	FE	5B	53	85	6E	EE	86	80	F9	52	81	11	2A	48
F	C5	EA	EF	DB	B6	3F	37	77	6D	03	2E	D6	F4	BC	F8	6C

The values of the polynomials $p(x)$ for the Module operation are secret, that is, in fact, they actually make up additional key information. For the purposes of this paper, we will use the following polynomial values as a secret element $p_i(x)$ to obtain round keys:

$p_1(x)$=10000000000101011$_2$ = 0x1002B;
$p_2(x)$=10000000000101101$_2$ = 0x1002D;
$p_3(x)$=10000000000111001$_2$ = 0x10039;
$p_4(x)$=10000000000111111$_2$ = 0x1003F;
$p_5(x)$=10000000001000111$_2$ = 0x10047;
$p_6(x)$=10000000001010011$_2$ = 0x10053;
$p_7(x)$=10000000010001101$_2$ = 0x1008D;
$p_8(x)$=10000000010111101$_2$ = 0x100BD;

After ten rounds of transformations, the result is added modulo two with the value of the round subkey from which the current subkey was generated, resulting in a new round key. To get the next round key, you need to re-run a cycle of ten transformations [10].

III. DIFFERENTIAL PROPERTIES OF THE BASIC QAMAL CIPHER TRANSFORMATIONS

A. Main stages of differential analysis

Before proceeding to the differential analysis of the Qamal encryption algorithm, it is necessary to consider the differential properties of each of its operations separately. A detailed description of the method of differential analysis can be found in [12 – 14]. Note that there are four main stages of using the method of differential analysis.

Stage 1. Analysis of the differential properties of all conversion components in the encryption algorithm.

Stage 2. Search for the most likely value of the differential, that is, such a pair of input difference – output difference, the appearance of which is most likely.

Stage 3. Search for the right pairs of texts. That is, such texts for which the sum modulo two at the input to the encryption algorithm coincides with the input difference, and the sum of the values at the output of the encryption algorithm coincides with the output difference.

Stage 4. Analysis of the correct pairs of texts in order to determine the bits of the secret key.

The main difficulty of differential cryptanalysis lies in the difficulty of finding the correct pairs of texts, which in turn depends directly on the value of the probability of the considered differential. That is why finding the differential that has the highest probability is of primary importance. Knowing the difference in the most probable differential, we can predict how successful the analysis of the cipher itself, or its reduced version will be. This means determining the number of cipher rounds for which differential cryptanalysis is still possible.

B. Differential properties of the addition operation modulo 2

In differential cryptanalysis, the texts being transformed are considered not separately, but jointly. To be more precise, their difference is considered, which is defined as a result of the addition module two of these texts: $\Delta X = X \oplus X1$.

In this case, the difference value ΔX will contain zeros in those positions in which the original messages were equal and «1» where the bits differed.

It is known that the operation of adding data with the secret key does not affect the change of the difference between the texts. This is due to the fact that the same secret encryption key is used for encryption. Thus, the texts will be added up with the same K value, which in turn will form a value equal to zero when stacked together: $\Delta X = X \oplus K_i \oplus X1 \oplus K_i = = X \oplus X1$.

C. Differential properties of bit change operation with S-block

Since the S-block of the cipher changes 8 bits to 8 bits, the possible range of input differences coincides with the range of output differences and is in the range from 0 to 255. We have constructed a table of the dependence of output differences ΔC of the S-block on the value of input difference ΔA and revealed the following properties:

Property 1. The value ΔC = 0 at the output of the transformation can be obtained only in the case when ΔA = 0. In this case, the probability of the appearance of the difference at the output is 1.

Property 2. In the constructed analysis table, the maximum probability value is 6/256 = 3/128.

Property 3. There are values of input differences ΔA, which remain unchanged after passing through the S-block. These are, for example, values (in decimal form) such as ΔA = 2, 3, 4, 6, 15, 16, 17, 18 and others.

Property 4. The value of the input difference ΔA = 254 (ΔA = 0xFE) is converted to the value of the difference ΔC = 128 (ΔC = 0x80) with the probability p = 4/256 = 1/64.

D. Differential properties of the transformation Mixer1

Although the fact that the Mixer1 transformation is a linear transformation, it is necessary to determine how the different values change when applying the addition modulo 256. It is known that when performing the addition modulo 2^n, the difference remains unchanged with probability $p = 1$ only if the input difference contains only one nonzero bit in the most significant bit. Thus, if the value of the difference equal to 0x80 is used in the Mixer1 transform, then whatever transformations we make with it, the probability will always remain equal to 1. The Mixer1 transformation depends on four bytes of one column. Therefore, it is important to consider how the output values will change. In this case, from the point of view of differential cryptanalysis, we are interested in those variants that affect the least number of active bytes. Since in the Mixer1 operation, addition is performed modulo 256, the 0x80 difference value will always remain the same. So, the addition of two identical differences 0x80 and 0x80 modulo 256 (0x100) will lead to zero. Thus, we can consider 15 options for populating the source column of the Mixer1 transformation, where the byte values can only be 0x00 or 0x80. An example of one such conversion is shown in Table IV.

TABLE IV.
THE RESULT OF THE DIFFERENCE CONVERSION IN THE MIXER1 TRANSFORMATION. OPTION 1

Source state	First change	Second change	Third change	Fourth change
0x80	0x80	0	0	0
0	0x80	0x80	0	0
0	0	0x80	0x80	0
0	0	0	0x80	0x80

E. Differential properties of the transformation Mixer1

The Mixer2 conversion is a linear conversion. It has no effect on the change in the probability of the difference transformation. However, in order to build multi-round characteristics, it is important to determine exactly how the value of strings containing the value of 0x80 in one of the bytes that will be obtained after the Mixer1 conversion will be converted. It is important to remember that each string uses its own polynomial *m* for multiplication. Each line contains 4 bytes. If to consider that each byte can contain the value of difference equal 0, or value of a difference equal 0x80 it turns out all 15 possible fillings for each line from 0x0000000080 to 0x80808080. Let's consider how these differences will be transformed with the use of polynomials *m* (for the block version of 128 bits a total of 60 variants are obtained: 15 variants of filling and 4 polynomials *m*). We are interested in cases when bytes at the output of the Mixer 2 transformation, after passing through the replacement S-block, can be converted to 0x80 values. That is, in the table of dependences ΔA and ΔC at the intersection of ΔA formed from the byte of the conversion output Mixer2 and ΔC=0x80, there must be a value different from 0. We have developed a program with the help of which we have calculated all possible variants.

As a result of using this program, it was found that only one of the 60 considered combinations satisfies the given condition. The input difference equal to 0x80808000 is converted to the difference 0xBBC868CF and after passing through the S-block of replacement can be converted to the difference value 0x80808080. Exactly this combination we will use in the future to build multi-round characteristics.

IV. CONSTRUCTION OF MULTI-ROUND CHARACTERISTICS

Based on the differential properties of the main Qamal cipher operations, let's build a multi-round characteristic and determine its probability. Our task is to construct the characteristic in such a way that as few active S-blocks as possible are affected. This directly affects the probability of finding the right pair of texts for a given characteristic. Our task is to determine how many Qamal rounds can be analyzed faster than using the full brute force method. A 128-bit block of data uses a 128-bit secret key, which means that the complexity of a full brute force attack is 2^{128}.

TABLE V.
DIFFERENCE CONVERSION FOR THE FIRST ROUND OF QAMAL CIPHER

First round input			
0xFE	0xFE	0xFE	0
0	0	0	0
0	0	0	0
0xFE	0xFE	0xFE	0
Mixer 1 transformation input			
0x80	0x80	0x80	0
0	0	0	0
0	0	0	0
0x80	0x80	0x80	0
Mixer 2 transformation input			
0	0	0	0
0	0	0	0
0x80	0x80	0x80	0
0	0	0	0

Consider the first round of encryption. We omit the addition operation with the round subkey since it does not affect the change in the difference of texts. We need the value 0x80

bytes to appear at the input of the Mixer1 transformation. In accordance with property 4 of section 4.3, the value 0xfe will be converted to the value 0x80 with a probability of 4/256 = 1/64 = 2^{-6}. At the same time, we should form the input of the first round in such a way that after the Mixer1 transformation the non-zero difference is in the third line of the state array. If the input difference affects the first and fourth bytes for the first three state columns, then after the replacement S-boxes, all non-zero bytes are converted to 0x80 bytes with a probability (2^{-6}) 6 = 2^{-36}. It can be seen that already from the first round of encryption the probability of obtaining a round characteristic is rather small. The Mixer1 transformation will change the state array without affecting the overall probability. As a result, nonzero bytes will appear only in the first three positions of the third row. All other values will be zero. The conversion scheme for the first round is presented in detail in table V.

In section 2.5 it was shown that if the input of the third line in the Mixer2 transformation is 0x80808000, the output will be 0xBBC868CF. Each byte of the difference 0xBBC868CF can be converted to the byte 0x80. The probability that all four bytes will be converted to 0x80 values is $(2^{-7})^4 = 2^{-28}$. Thus, the final probability for two rounds of encryption is 2^{-64}. After the Mixer1 function, the second and fourth lines will be filled with 0x80 bytes, as shown in Table VI.

TABLE VI.
DIFFERENCE CONVERSION FOR THE SECOND ROUND OF QAMAL CIPHER

S-block input			
0	0	0	0
0	0	0	0
0xBB	0xC8	0x68	0xCF
0	0	0	0
Mixer 1 transformation input			
0	0	0	0
0	0	0	0
0x80	0x80	0x80	0x80
0	0	0	0
Mixer 2 transformation input			
0	0	0	0
0x80	0x80	0x80	0x80
0	0	0	0
0x80	0x80	0x80	0x80

As a result of the analysis of the differential properties of the Mixer2 transformation, it was found that for the second and fourth lines the input difference 0x80808080 cannot be converted so that in the next round after the S-transformation all the difference bytes are equal to 0x80. Therefore, we have considered other options for conversions. The second and fourth status lines, containing the difference value 0x95D14821, obtained after the Mixer2 transformation, will be input of the S-block of the third round (table VII). In accordance with the table of differential properties, we determined that byte 0x95 can be replaced by byte 0x80. For the remaining bytes, a replacement option has been selected that will affect three non-zero bytes of the column (out of four) after the Mixer1 conversion. The 0xD1 byte, according to the analysis table, has a chance to be converted to 0x40 byte and 0xC0 byte. In this case, the Mixer1 transformation will be performed according to Table VIII. 0x48 and 0x21 bytes cannot be converted in the same way as 0xD1 bytes, so it was found that they can be converted to 0x10 and 0xF0 bytes, in which case the Mixer1 transformation will be performed according to Table IX.

The conversion probability of each byte on the S-block replacement for the third round is 2^{-7}. A total of 8 non-zero blocks are used in the third round. Thus, the probability of conversion in the third round is $(2^{-7})^8 = 2^{-56}$. It turns out that the probability for three rounds of cipher will be 2^{-120}, which is very close to the value of the probability of a complete search. Therefore, it makes no sense to consider the transformation of the difference further. We need to determine the value of the difference at the output of the third round of encryption. To do this, let's consider the difference at the input to the Mixer2 transform of the third round (Table VII). Applying the Mixer1 and Mixer2 transformations to it, we obtain the state of differences as shown in Table X.

TABLE VII.
DIFFERENCE CONVERSION FOR THE THIRD ROUND OF QAMAL CIPHER

S-block input			
0	0	0	0
0x95	0xD1	0x48	0x21
0	0	0	0
0x95	0xD1	0x48	0x21
Mixer 1 transformation input			
0	0	0	0
0x80	0x40	0x10	0x10
0	0	0	0
0x80	0xC0	0x1F	0x1F
Mixer 2 transformation input			
0x80	0xc0	0x30	0x30
0	0x80	0x20	0x20
0x80	0x40	0x10	0x10
0	0	0	0

TABLE VIII.
MIXER1 TRANSFORMATION FOR 0X40 AND 0XC0 BYTES

Input differences	First step	Second step	Third step	Output difference
0x00	0x00	0x40	0x80	0xC0
0x40	0x00	0x00	0x40	0x80
0x00	0x40	0x00	0x00	0x40
0xC0	0x00	0x40	0x00	0x00

TABLE IX.
MIXER1 TRANSFORMATION FOR 0X10 AND 0XF0 BYTES

Input differences	First step	Second step	Third step	Output difference
0x00	0x00	0x10	0x20	0x30
0x10	0x00	0x00	0x10	0x20
0x00	0x10	0x00	0x00	0x10
0xF0	0x00	0x10	0x00	0x00

TABLE X.
THE STATE OF DIFFERENCES AFTER THE THIRD ROUND OF ENCRYPTION

0x4C	0x6B	0x94	0xEA
0xAD	0xDE	0x47	0x5B
0xE1	0xB2	0xD3	0xB1
0x00	0x00	0x00	0x00

V. CONCLUSIONS

We have considered the project algorithm of symmetric Qamal encryption, which is considered as a candidate for the standard of data encryption in the Republic of Kazakhstan. We have shown that for a version with a 128-bit data block and a secret key of the same length, differential cryptanalysis becomes inapplicable after three rounds of encryption. The encryption has yet to be thoroughly tested using other cryptoanalytic attacks to fully verify its reliability.

REFERENCES

[1] D. Coppersmith, C. Holloway, S. Matyas and N. Zunic, “The Data Encryption Standard,” *Information Security Technical Report*, vol. 2, no. 2, 1997, pp. 22-24

[2] B. Shnier, “Chapter 12 – Data Encryption Standard (DES),” in *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, Hoboken, NJ, USA: John Wiley & Sons, 1996, pp. 370-421.

[3] B. Preneel The NESSIE Project: Towards New Cryptographic Algorithms https://pdfs.semanticscholar.org/0e15/db04d5fcb3b39fb8aad1435c7aa348586a94.pdf

[4] Specifications of e-Government Recommended Ciphers, https://www.cryptrec.go.jp/en/method.html

[5] GOST 28147-89: Encryption, Decryption and Message Authentication Code (MAC) Algorithms, https://tools.ietf.org/html/rfc5830

[6] GOST R 34.12–2015 “Information technology. Cryptographic data security. Block ciphers,” https://tc26.ru/en/standards/standards/gost-r/gost-r-34-12-2015-information-technology-cryptographic-data-security-block-ciphers.html

[7] P. Jovanovic, I. Polian, “Fault-based Attacks on the Bel-T Block Cipher Family,” in *Proc. Design, Automation & Test in Europe – DATE 2015*, 2015, https://zerobyte.io/publications/2015-JP-belt.pdf, DOI: 10.7873/DATE.2015.0046 ·

[8] A New Encryption Standard of Ukraine: The Kalyna Block Cipher - https://pdfs.semanticscholar.org/7771/8fbf6c2044b6f1aa2e66a1eda99121caa4da.pdf

[9] Report on the research work "Development of software and hardware for cryptographic protection of information during its transmission and storage in info communication systems and general-purpose networks", *Committee of Science of the Ministry of Education and Science of the Republic of Kazakhstan, Institute of Information and Computing Technologies, State Registration No. 0118RK01064*.

[10] K. Algazy, R. Biyashev, N. Kapalova, L. Babenko, E. Ishchukova, S. Nyssanbayeva, "Investigation of the different implementations for the new cipher Qamal", in *Proceedings of the 12th International Conference on Security of Information and Networks - SIN '19*, 2019, . Association for Computing Machinery, New York, NY, USA, Article 8, 1–8. DOI:https://doi.org/10.1145/3357613.3357622

[11] V. Korchynskyi, V. Kildishev, O.Riabukha, O.Berdnikov, “The generating random sequences with the increased cryptographic strength,” *Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska*, vol. 10, no. 1, pp. 20-23, January 2020, https://doi.org/10.35784/iapgos.916

[12] E. Biham, A. Shamir: “Differential Cryptanalysis of the Full 16-round DES, Crypto'92, Berlin: Springer-Verlag, 1998, p.19.

[13] E. Biham, A. Shamir: “Differential Cryptanalysis of DES-like Cryptosystems”, *Extended Abstract, Crypto'90*, Berlin: Springer-Verlag, 1998, p.21

[14] E.A. Ishchukova, E.A. Tolomanenko, L.K. Babenko, “Differential analysis of 3 round Kuznyechik,” in *Proceedings of the 10th International Conference on Security of Information and Networks (SIN '17).* Association for Computing Machinery, New York, NY, USA, 29–36. DOI:https://doi.org/10.1145/3136825.3136880