Algebraic attacks on block ciphers using quantum annealing

Elżbieta Burek$^{1[0000-0003-2937-0833]}$, Michał Misztal$^{1[0000-0002-0826-1104]}$, Michał Wroński$^{1[0000-0002-8679-9399]}$

Military University of Technology, Kaliskiego Str. 2, Warsaw, Poland
{`elzbieta.burek, michal.misztal, michal.wronski`}`@wat.edu.pl`

Abstract. This paper presents method for transformation of algebraic equations of symmetric cipher into the QUBO problem. After transformation of given equations $f_1, f_2, \ldots, f_n$ to equations over integers $f'_1, f'_2, \ldots, f'_n$, one has to linearize each, obtaining $f'_{lin_i} = lin(f'_i)$, where lin denotes linearization operation. Finally, one can obtain problem in the QUBO form as $\left(f'_{lin_1}\right)^2 + \cdots + \left(f'_{lin_n}\right)^2 + Pen$, where Pen denotes penalties obtained during linearization of equations and n is the number of equations.
In this paper, we show examples of the transformation of some block ciphers to the QUBO problem. What is more, we present the results of the transformation of the full AES-128 cipher to the QUBO problem, where the number of variables of equivalent QUBO problem is equal to 237,915, which means, at least theoretically, that problem may be solved using the D-Wave Advantage quantum annealing computer. Unfortunately, it is hard to estimate the time this process would require.

Keywords: Cryptanalysis, AES, symmetric ciphers, algebraic attacks, quantum annealing

1 Introduction

This paper presents method for transformation of algebraic equations of symmetric cipher into the QUBO problem. After such transformation, obtained QUBO problem may be solved using quantum annealing approach, especially on D-Wave computer. At first, algebraic equations of cipher have to be obtained. The idea here is the same as in the case of algebraic attacks. After obtaining Boolean equations of given cipher in algebraic normal form, each equation f has to be transformed to equation of Boolean variables with integer coefficients as $f' = f - 2k$, where k is an integer, $k \leq \lfloor \frac{f_{max}}{2} \rfloor$ and f_{max} is maximal value polynomial f can take. Moreover, k has to be written as sum of Boolean variables $k = \sum_{i=0}^{bl(f_{max})-1} 2^i k_i + f_{max} + 1 - 2^{bl(f_{max}-1)}$, where $bl(x)$ denotes bit-length of integer x. These ideas may be found in [2]. After transformation of given equations, one has to linearize each, obtaining $f'_{lin_i} = lin(f'_i)$, where lin denotes linearization operation. Finally, one can obtain problem in the QUBO form as

$\left(f'_{lin_1}\right)^2 + \cdots + \left(f'_{lin_n}\right)^2 + Pen$, where Pen denotes penalties obtained during linearization of equations and n is the number of equations.

In this paper, we present the results of the transformation of the complete AES-128 cipher to the QUBO problem, where the number of variables of equivalent QUBO problem is equal to $237,915$, which means, at least theoretically, that problem may be solved using the D-Wave Advantage computer. Unfortunately, it is hard to estimate the time this process would require.

2 The idea of Algebraic Attacks

Algebraic attacks [1], [5], [4], [3] (rather than statistical like differential and linear cryptanalysis) in nature exploit the internal algebraic structure of the algorithm. The general idea is quite simple. First, obtain a representation of the cipher as a system of equations. Then try to solve it to recover unknowns which are secret key bits. In theory, most modern (block and stream) ciphers can be described by a system of multivariate polynomials over a finite field. For the majority of the ciphers, such systems are too complex for any practical solving method. The term Algebraic Attack typically refers to the technique of expressing the whole cryptosystem as a large system of multivariate polynomial equations. We have to include linear equations from the diffusion layer and key addition, non-linear equations from the substitution layer, and Key Schedule equations to obtain a polynomial system from a block cipher. For the non-linear equations, we distinct two cases: explicit equations which are equations of the form $y_i = f_i\left(x_0, x_1, \ldots, x_{n-1}\right)$, and implicit equations which are equations of the form $g\left(x_0, , x_{n-1}; y_0, , y_{m-1}\right) = 0$. We usually consider algebraic attacks when these equations have a small degree. When mounting an algebraic attack for each non-linear component of the cipher, we usually attempt to obtain as many low-degree, linearly independent equations as possible. That over defined systems are generally easier to solve. In its general form, an algebraic attack is mounted by expressing the whole cipher operation as a system of low-degree multivariate equations, involving the (known) plaintext and ciphertext values, the secret key, and a large number of intermediate variables arising in the cipher operations. It results in huge systems (typically over $GF(2)$). Attack usually requires only one single plaintext/ciphertext pair. The solution of the system is equivalent to key recovery. It means that for algebraic attacks, we need efficient algorithms for solving algebraic systems. So, the methods for solving polynomials systems are the essential ingredients of algebraic attacks and have recently started receiving special attention from the cryptographic community. To most common methods used in cryptology, we may include Linearisation principle, XL and variants, Groebner Basis algorithms (e.g., Buchberger, F4, F5), SAT-solvers, and others. Linearization is a well-known technique for solving large systems of multivariate polynomial equations, which we used in this paper. We consider all monomials in the system as independent variables and solve the system using linear algebra techniques (i.e., Gaussian reduction). The effectiveness of the method depends on the number of linearly independent polynomials in the system. In the case of Boolean functions with m variables, the total number of monomials of degree d is $M = \sum_{i=1}^{d} \binom{m}{i}$. The complexity of Gaussian reduction is $O(M^3)$. We may theoretically write $O(M^{\omega})$, where $\omega \approx 2+\epsilon$ if the matrix of the linearized system is sparse. Note that the problem of estimating the rank of the linearized system is challenging. In order to apply the linearization method, the number of linear equations in the system needs to be approximately the same as the number of monomials in the system. When this is not the case, several proposed techniques attempt to generate enough linear equations.

3 Quantum computing

Quantum computing is one of the most promising approaches used to cryptanalysis of many cryptographic algorithms. The first major paper in this field was a paper written by Peter W. Shor [10], where he described a quantum algorithm for factorization and discrete logarithm computation. Since that time, many efforts have been made to construct a quantum computer, which would break algorithms used in real worlds applications. However, the progress in this field is huge, and till now, the biggest quantum computer made by Google has 72 working qubits [8]. It is still too little to break real-world algorithms.

In the last few years, the second approach of quantum computing has gained much popularity. This approach is quantum annealing, and such computers are built by the D-Wave company. Unfortunately, such computers may solve only Ising problems. Of course, other problems as QUBO (Quadratic Unconstrained Binary Optimization) and DQM (Discrete Quadratic Model) may be transformed to the Ising problem.

The Quantum annealing approach allowed us to gain some success in factorization, where for some time, the quantum factorization record had belonged to the D-Wave computer. Using transformation of integer factorization to the QUBO problem, Dridi and Alghassi [7] were able to factorize integer $200,099$, which result was later beaten by Jiang et al. [9], and by Wang et al. [11], who factorized 20-bit integer $1,028,171$. Such transformation of factorization problem to the QUBO problem requires approximately $\frac{n^2}{4}$ logical qubits, where n is the bit-length of factorized integer.

Moreover, it is also possible to transform discrete logarithm problem over prime fields to the QUBO problem [12], where approximately $\frac{n^3}{2}$ logical qubits are necessary, where n is the bit-length of p.

These all arguments make that quantum annealing may be used in cryptanalysis of cryptographic algorithms.

4 QUBO problem [6]

QUBO (Quadratic Unconstrained Binary Optimization) [6] is a significant problem with many real-world applications. One can express the QUBO model by the following optimization problem:

$$\min_{x\in\{0,1\}^n} x^T Q x, \tag{1}$$

where Q is an $N \times N$ upper-diagonal matrix of real weights, and x is a vector of binary variables. Moreover, diagonal terms $Q_{i,i}$ are linear coefficients, and the nonzero off-diagonal terms are quadratic coefficients $Q_{i,j}$.

QUBO problem may be also viewed as problem of minimizing the function

$$f(x) = \sum_i Q_{i,i} x_i + \sum_{i<j} Q_{i,j} x_i x_j. \tag{2}$$

4.1 Linearization for the QUBO problem

Now we will show how the resulting 2-local terms may be reduced to 1-local terms. Let us note that each penalty monomial of the form $x_i x_j$ will be transformed, according to [9], in the following way

$$x_i x_j \rightarrow u_k + 2(x_i x_j - 2u_k(x_i + x_j) + 3u_k). \tag{3}$$

It means that

$$x_i x_j = u_k + 2(x_i x_j - 2u_k(x_i + x_j) + 3u_k), \tag{4}$$

if $x_i x_j = u_k$ and

$$x_i x_j < u_k + 2(x_i x_j - 2u_k(x_i + x_j) + 3u_k), \tag{5}$$

if $x_i x_j \neq u_k$.

It results in that $x_i x_j$ term may be transformed to linear form by replacing $x_i x_j$ with u_k. Additionally, one has to add a constraint, given by penalty term:

$$min(x_i x_j) = min\left(u_k + 2(x_i x_j - 2u_k(x_i + x_j) + 3u_k)\right). \tag{6}$$

5 Transformation of multivariate polynomial equations from block ciphers to QUBO problem

The D-Wave computers using quantum annealing have limited resources, so in our work, we focus primarily on the number of equations, the number of variables, and the number of monomials. Therefore, the equations generated for block ciphers are of degree two or less.

We will explain the transformation of Boolean block cipher equations into the QUBO problem using a very small block cipher created for this article, called SmallCipher.

The overall structure of SmallCipher is shown in the Figure 1. The plaintext is processed with 4-bit blocks using a 4-bit round key. The result of processing a single block is a 4-bit ciphertext block. In the example, encryption is performed in one round (Figure 1.a) and two (Figure 1.b).

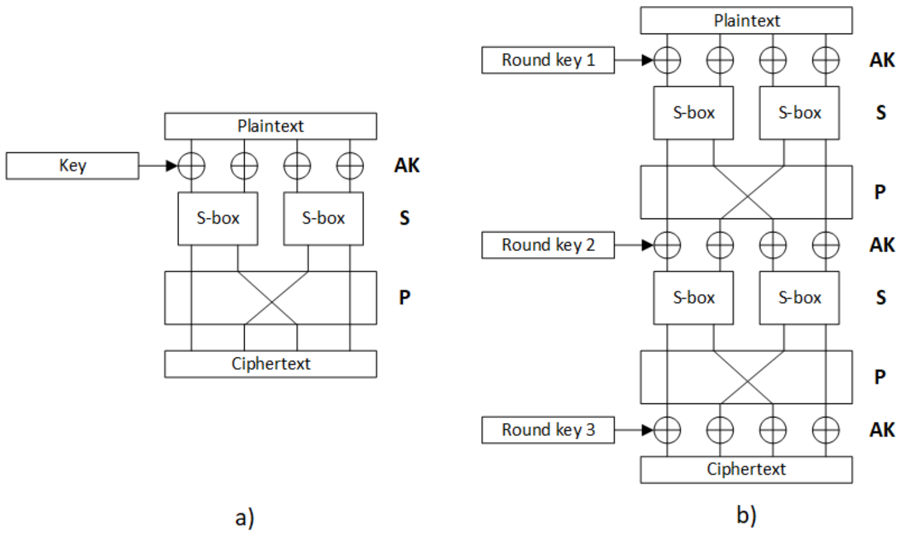

Fig. 1. Structure of the SmallCipher, a) one-round cipher, b) two-round cipher.

The encryption algorithm consists of three separate functions: add key (AK), substitution (S), and permutation (P). Thus, encryption can be presented as a superposition of the following functions:

$$P \circ S \circ AK$$

The key addition function is the bitwise XOR of 4-bit plaintext with a 4-bit round key.

The substitution function consists of two S-boxes, mapping two input bits to two output bits, according to the table:

IN	0	1	2	3
OUT	2	0	3	1

The permutation is a linear layer of this cipher, which is implemented according to the relationship: $(1, 3, 2, 4)$.

5.1 Overview of our approach

To obtain implicit equations of degree two or less, a list of Boolean polynomials satisfying the S-box is derived. There are seven polynomials for the S-box of SmallCipher: $x_1y_0, x_0 + y_1, x_0y_1 + x_0, x_1 + y_0 + 1, x_0x_1 + x_1y_1, x_0x_1 + x_0 + y_0y_1, x_0x_1 + x_0y_0 + x_0,$
where x_i is the input bits and y_i is the output bits of the S-box.

One-round SmallCipher The first example will be for a one-round SmallCipher with one round key. Figure 2 shows the variables that are needed to generate the equations. The number of variables is twelve.

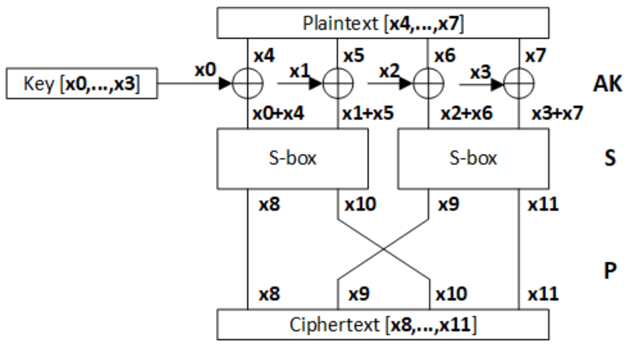

Fig. 2. Necessary variables for a one-round SmallCipher.

Since S-box satisfies seven polynomials, the number of equations (n) for one round of SmallCipher is 14, and the number of monomials in all equations is 68. The resulting equations are:

$f_1 = x_0 + x_4 + x_{10}$,
$f_2 = x_1 + x_5 + x_8 + 1$,
$f_3 = x_0x_1 + x_0x_5 + x_0x_8 + x_0 + x_1x_4 + x_4x_5 + x_4x_8 + x_4$,
$f_4 = x_0x_{10} + x_0 + x_4x_{10} + x_4$,
$f_5 = x_1x_8 + x_5x_8$,
$f_6 = x_0x_1 + x_0x_5 + x_1x_4 + x_1x_{10} + x_4x_5 + x_5x_{10}$,
$f_7 = x_0x_1 + x_0x_5 + x_0 + x_1x_4 + x_4x_5 + x_4 + x_8x_{10}$,
$f_8 = x_2 + x_6 + x_{11}$,
$f_9 = x_3 + x_7 + x_9 + 1$,
$f_{10} = x_2x_3 + x_2x_7 + x_2x_9 + x_2 + x_3x_6 + x_6x_7 + x_6x_9 + x_6$,
$f_{11} = x_2x_{11} + x_2 + x_6x_{11} + x_6$,
$f_{12} = x_3x_9 + x_7x_9$,
$f_{13} = x_2x_3 + x_2x_7 + x_3x_6 + x_3x_{11} + x_6x_7 + x_7x_{11}$,
$f_{14} = x_2x_3 + x_2x_7 + x_2 + x_3x_6 + x_6x_7 + x_6 + x_9x_{11}$.

During linearization, the number of variables increases, but the number of monomials does not change. Therefore, after transformation into an equation of Boolean variables with integer coefficients (f_i'), the next step is to perform the linearization of the above equations, during which the penalty is determined by the formula (4). The determined penalty is multiplied by a large integer constant c. After linearization, the number of variables increased to 38. The linearized equations are as follows:

$f'_{lin_1} = x_0 + x_4 + x_{10},$
$f'_{lin_2} = x_1 + x_5 + x_8 + 1,$
$f'_{lin_3} = x_0 + x_4 + x_{12} + x_{13} + x_{14} + x_{15} + x_{16} + x_{17},$
$f'_{lin_4} = x_0 + x_4 + x_{18} + x_{19},$
$f'_{lin_5} = x_{20} + x_{21},$
$f'_{lin_6} = x_{12} + x_{13} + x_{14} + x_{15} + x_{22} + x_{23},$
$f'_{lin_7} = x_0 + x_4 + x_{12} + x_{13} + x_{14} + x_{15} + x_{24},$
$f'_{lin_8} = x_2 + x_6 + x_{11},$
$f'_{lin_9} = x_3 + x_7 + x_9 + 1,$
$f'_{lin_{10}} = x_2 + x_6 + x_{25} + x_{26} + x_{27} + x_{28} + x_{29} + x_{30},$
$f'_{lin_{11}} = x_2 + x_6 + x_{31} + x_{32},$
$f'_{lin_{12}} = x_{33} + x_{34},$
$f'_{lin_{13}} = x_{25} + x_{26} + x_{27} + x_{28} + x_{35} + x_{36},$
$f'_{lin_{14}} = x_2 + x_6 + x_{25} + x_{26} + x_{27} + x_{28} + x_{37}.$
and the penalty is equal to:
$Pen = c \cdot (x_0x_1 + x_2x_3 + x_1x_4 + x_0x_5 + x_4x_5 + x_3x_6 + x_2x_7 + x_6x_7 + x_0x_8 + x_1x_8 + x_4x_8 + x_5x_8 + x_2x_9 + x_3x_9 + x_6x_9 + x_7x_9 + x_0x_{10} + x_1x_{10} + x_4x_{10} + x_5x_{10} + x_8x_{10} + x_2x_{11} + x_3x_{11} + x_6x_{11} + x_7x_{11} + x_9x_{11} - 2x_0x_{12} - 2x_1x_{12} - 2x_1x_{13} - 2x_4x_{13} - 2x_0x_{14} - 2x_5x_{14} - 2x_4x_{15} - 2x_5x_{15} - 2x_0x_{16} - 2x_8x_{16} - 2x_4x_{17} - 2x_8x_{17} - 2x_0x_{18} - 2x_{10}x_{18} - 2x_4x_{19} - 2x_{10}x_{19} - 2x_1x_{20} - 2x_8x_{20} - 2x_5x_{21} - 2x_8x_{21} - 2x_1x_{22} - 2x_{10}x_{22} - 2x_5x_{23} - 2x_{10}x_{23} - 2x_8x_{24} - 2x_{10}x_{24} - 2x_2x_{25} - 2x_3x_{25} - 2x_3x_{26} - 2x_6x_{26} - 2x_2x_{27} - 2x_7x_{27} - 2x_6x_{28} - 2x_7x_{28} - 2x_2x_{29} - 2x_9x_{29} - 2x_6x_{30} - 2x_9x_{30} - 2x_2x_{31} - 2x_{11}x_{31} - 2x_6x_{32} - 2x_{11}x_{32} - 2x_3x_{33} - 2x_9x_{33} - 2x_7x_{34} - 2x_9x_{34} - 2x_3x_{35} - 2x_{11}x_{35} - 2x_7x_{36} - 2x_{11}x_{36} - 2x_9x_{37} - 2x_{11}x_{37} + 3x_{12} + 3x_{13} + 3x_{14} + 3x_{15} + 3x_{16} + 3x_{17} + 3x_{18} + 3x_{19} + 3x_{20} + 3x_{21} + 3x_{22} + 3x_{23} + 3x_{24} + 3x_{25} + 3x_{26} + 3x_{27} + 3x_{28} + 3x_{29} + 3x_{30} + 3x_{31} + 3x_{32} + 3x_{33} + 3x_{34} + 3x_{35} + 3x_{36} + 3x_{37}).$

The next step is to find the integer k_i, which is determined for each linearized equation f'_{lin_i} and depends on the number of monomials in the equation. The k_i values presented by the following variables are as follows:

$$\begin{array}{lll}
k_1 = x_{38}, & k_6 = x_{47} + 2x_{48}, & k_{11} = x_{57} + x_{58}, \\
k_2 = x_{39} + x_{40}, & k_7 = x_{49} + 2x_{50}, & k_{12} = x_{59}, \\
k_3 = x_{41} + 2x_{42} + x_{43}, & k_8 = x_{51}, & k_{13} = x_{60} + 2x_{61}, \\
k_4 = x_{44} + x_{45}, & k_9 = x_{52} + x_{53}, & k_{14} = x_{62} + 2x_{63}, \\
k_5 = x_{46}, & k_{10} = x_{54} + 2x_{55} + x_{56}, &
\end{array}$$

the number of variables has increased to 64.

The last step is to find the sum of the squares $(f'_{lin_1} - 2k_1)^2 + (f'_{lin_2} - 2k_2)^2 + \cdots + (f'_{lin_{14}} - 2k_{14})^2 + Pen$. Additionally, since variables are binary, then $x_i^2 = x_i$. Finally, the problem considered in the form of QUBO consists of 375 monomials and 64 variables.

Two-round SmallCipher The second example will be for a two-round SmallCipher with three round key. Figure 3 shows the variables that are needed to generate the equations. For the number of rounds greater than one, each intermediate state needs separate variables.

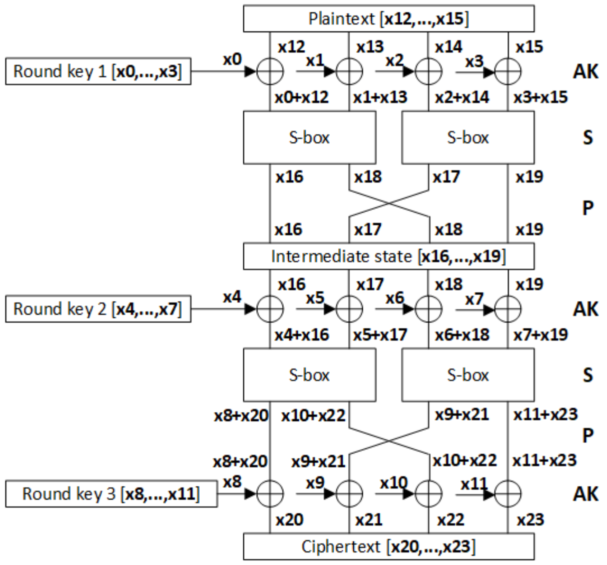

Fig. 3. Necessary variables for a two-round SmallCipher.

The number of Boolean equations generated for the two-round SmallCipher is 28, and these equations contain 24 variables and 162 monomials.
Assuming for now that the variables of the round keys are independent, linearization was performed for the Boolean equations. Since there are 74 monomials of degree two in the equations, the number of variables after linearization increased to 98, and the penalty includes 296 monomials. The next 60 variables were used to represent 28 values of k. Finally, after determining the sum of squares and adding a penalty, we obtain a polynomial of degree two, with $1,091$ monomials and 158 binary variables.

Block ciphers use round keys generated with the key generation algorithm, therefore, for this example, a very simple round key generation algorithm was designed, based on the AES key expansion. A 4-bit key is grouped into two 2-bit words and then expanded to the form of six words, as shown in figure 4. The first two words are a copy of the primary key, and the next four are calculated according to the following formulas:
$w_2 = w_0 \oplus S(swap(w_1))$,
$w_3 = w_1 \oplus w_2$,
$w_4 = w_2 \oplus S(swap(w_3))$,
$w_5 = w_3 \oplus w_4$,
where S is a substitution with S-box and $swap$ is a bit-swap function.

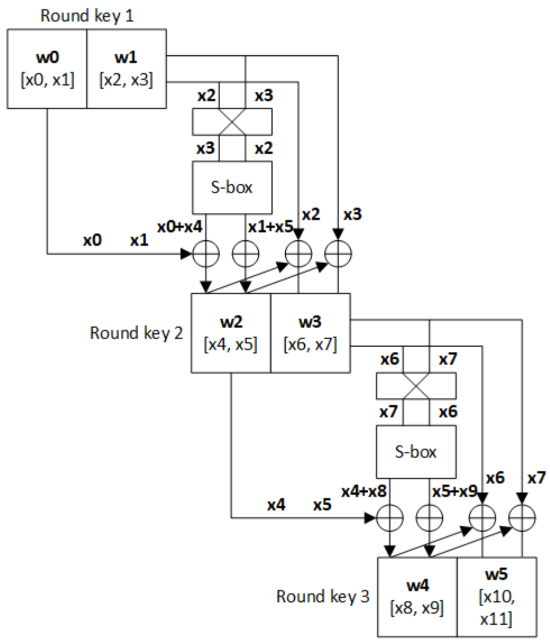

Fig. 4. Necessary variables for a round key generation algorithm.

In the presented algorithm for generating round keys, an S-box is used, therefore, to obtain equations of degree two or less, polynomials satisfying the S-box are used. For the bits of the word passing through the S-box, seven equations are generated, in addition, for the remaining words that are only bitwise XOR, the number of generated equations is equal to their bit-length. Therefore, in this example, the number of generated equations is 18, including 12 variables. These equations will be appended to the equations generated for the cipher. The generated Boolean equations for the round keys are as follows:

$f_{29} = x_1 + x_3 + x_5,$
$f_{30} = x_0 + x_2 + x_4 + 1,$
$f_{31} = x_0x_3 + x_2x_3 + x_3x_4 + x_3,$
$f_{32} = x_1x_3 + x_3x_5 + x_3,$
$f_{33} = x_0x_2 + x_2x_4,$
$f_{34} = x_1x_2 + x_2x_3 + x_2x_5,$
$f_{35} = x_0x_1 + x_2x_3 + x_1x_4 + x_0x_5+$
$+x_4x_5 + x_3,$
$f_{36} = x_2 + x_4 + x_6,$
$f_{37} = x_3 + x_5 + x_7,$
$f_{38} = x_5 + x_7 + x_9,$
$f_{39} = x_4 + x_6 + x_8 + 1,$
$f_{40} = x_4x_7 + x_6x_7 + x_7x_8 + x_7,$
$f_{41} = x_5x_7 + x_7x_9 + x_7,$
$f_{42} = x_4x_6 + x_6x_8,$
$f_{43} = x_5x_6 + x_6x_7 + x_6x_9,$
$f_{44} = x_4x_5 + x_6x_7 + x_5x_8 + x_4x_9+$
$+x_8x_9 + x_7,$
$f_{45} = x_6 + x_8 + x_{10},$
$f_{46} = x_7 + x_9 + x_{11}.$

After joining the round key equations to the cipher equations, there are 92 monomials of degree two among the 224 monomials. After linearization, the number of variables increased to 116, and the determined penalty is a polynomial consisting of 368 monomials. Then 46 k values were determined using the next 84 variables. Finally, after computing the sum of the squares and adding the penalty, the computed polynomial consists of $1,324$ monomials and 200 binary variables.

5.2 An attack example

For a one-round SmallCipher, for an example of a plaintext - ciphertext pair, we made an attack using a simple D-Wave computer simulator. We randomly selected plaintext and a key: $P = [1,0,0,1], K = [1,0,1,1]$, and a ciphertext $C = [1,1,0,1]$ was generated for the selected parameters.

After substituting bits of the plaintext and ciphertext for the appropriate variables and naming the variables with the key: $K = [x_0, x_1, x_2, x_3]$, we generated the following 13 equations (one equation was reduced to zero) with four variables and 28 monomials:

$$\begin{array}{lll}
f_1 = x_0 + 1, & f_6 = x_0x_1 + x_1, & f_{11} = x_3 + 1,\\
f_2 = x_1, & f_7 = x_0x_1 + x_0 + x_1 + 1, & f_{12} = x_2x_3 + x_2 + x_3 + 1,\\
f_3 = x_0x_1 + x_1, & f_8 = x_2 + 1, & f_{13} = x_2x_3 + 1.\\
f_4 = x_0 + 1, & f_9 = x_3 + 1, & \\
f_5 = x_1, & f_{10} = x_2x_3 + x_2, &
\end{array}$$

After linearization, the number of variables increased to 6 in the following equations:

$$\begin{array}{lll}
f'_{lin_1} = x_0 + 1, & f'_{lin_6} = x_1 + x_4, & f'_{lin_{11}} = x_3 + 1,\\
f'_{lin_2} = x_1, & f'_{lin_7} = x_0 + x_1 + x_4 + 1, & f'_{lin_{12}} = x_2 + x_3 + x_5 + 1,\\
f'_{lin_3} = x_1 + x_4, & f'_{lin_8} = x_2 + 1, & f'_{lin_{13}} = x_5 + 1,\\
f'_{lin_4} = x_0 + 1, & f'_{lin_9} = x_3 + 1, & \\
f'_{lin_5} = x_1, & f'_{lin_{10}} = x_2 + x_5, &
\end{array}$$

and with the following penalty:
$Pen : c \cdot (x_0x_1 + x_2x_3 - 2x_0x_4 - 2x_1x_4 - 2x_2x_5 - 2x_3x_5 + 3x_4 + 3x_5)$,
where $c = 1000$.

The k values for the above equations were presented using the next 13 variables: $k_1 = x_6, k_2 = 0, k_3 = x_7, k_4 = x_8, k_5 = 0, k_6 = x_9, k_7 = x_{10}+x_{11}, k_8 = x_{12}, k_9 = x_{13}, k_{10} = x_{14}, k_{11} = x_{15}, k_{12} = x_{16} + x_{17}, k_{13} = x_{18}$.

Finally, the problem in the QUBO form consist of 42 monomials and 19 variables:
$1002x_0x_1 + 1002x_2x_3 - 1998x_0x_4 - 1994x_1x_4 - 1996x_2x_5 - 1998x_3x_5 - 4x_0x_6 - 4x_1x_7 - 4x_4x_7 - 4x_0x_8 - 4x_1x_9 - 4x_4x_9 - 4x_0x_{10} - 4x_1x_{10} - 4x_4x_{10} - 4x_0x_{11} - 4x_1x_{11} - 4x_4x_{11} + 8x_{10}x_{11} - 4x_2x_{12} - 4x_3x_{13} - 4x_2x_{14} - 4x_5x_{14} - 4x_3x_{15} - 4x_2x_{16} - 4x_3x_{16} - 4x_5x_{16} - 4x_2x_{17} - 4x_3x_{17} - 4x_5x_{17} + 8x_{16}x_{17} - 4x_5x_{18} + 9x_0 + 7x_1 + 7x_2 + 9x_3 + 3005x_4 + 3007x_5 + 4x_7 + 4x_9 + 4x_{14} + 8.$

Using the D-Wave simulator, we recovered the values of nineteen binary variables, including the key bits:
$x_0 = 1, x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 0, x_5 = 1, x_6 = 1, x_7 = 0, x_8 = 1, x_9 = 0,$ $x_{10} = 1, x_{11} = 0, x_{12} = 1, x_{13} = 1, x_{14} = 1, x_{15} = 1, x_{16} = 1, x_{17} = 1, x_{18} = 1.$

6 The transformation of the AES-128 cipher equations into the QUBO problem

Another cipher we considered is the AES-128 standard, which requires 128 bits of plaintext and 128 bits of the master key. After 10 rounds, the plaintext block is encrypted and returns a 128-bit ciphertext.

6.1 One-round AES-128

First, we focused on one round of the AES-128 cipher, consisting of key addition, S-box byte substitution, shift rows, and mix columns.

The number of Boolean equations generated for one round of the AES-128 cipher is 624, consisting of a total of $283,568$ monomials of degree at most two. During the linearization of $16,576$ various monomials, the degree of two was replaced with successive binary variables. After linearization, the number of variables increased to $16,960$. During linearization, we also determined a penalty that consists of at most $66,304$ monomials. In the next step, we presented 624 integer k_i values with 5024 new binary variables.

Finally, after calculating the sum of squares and adding the penalty, one round of the AES-128 cipher was presented in the form of the QUBO problem using $21,984$ binary variables and at most $227,947,152$ monomials.

6.2 Full AES-128

In the full AES-128 cipher, due to a large number of equations and monomials, the final transformation parameters of the equations to the QUBO problem will be estimated.

To generate Boolean equations of degree at most two, we need 128 variables for the plaintext, 128 variables for the ciphertext, $1,408$ variables for eleven round keys, and $1,152$ variables for intermediate states, and we need a total of $2,816$ binary variables. The number of Boolean equations of degree two or less needed to represent the encryption algorithm of the AES-128 cipher is $6,240$. Since separate variables represent the intermediate states of each round and the round keys, there are variables from two adjacent intermediate states in the single-round equations. So each round produces equations with the same number of monomials. Hence, considering the results from the previously described case of the one-round AES-128 cipher, the number of different monomials in all equations is $169,610$. Among these monomials, there are $165,760$ monomials of degree two, which new variables will replace during the linearization. Hence the number of variables increased to $168,576$. Since the number of monomials in the corresponding equations between rounds is the same, the values of integers k_i between rounds should also be the same. It means that the number of variables needed for k_i values in each round is the same. Therefore, the number of variables needed to represent the values of all k_i is $50,240$. Finally, $218,816$ binary variables are needed for the AES-128 cipher encryption algorithm.

Next, we considered the algorithm for generating round keys. The relationships between the variables of the round keys can be represented by $2,520$ Boolean equations, of degree two or less. These equations consist of $11,121$ different monomials, among which there are $9,712$ monomials of degree two. During linearization, these monomials were replaced with new variables. Then, we have determined k_i values for each equation that $9,387$ binary variables can represent.

In summary, $237,915$ binary variables are needed to present the AES-128 cipher equations as the QUBO problem.

7 Conclusion

In this paper approach of algebraic attacks using quantum annealing was presented. We showed the practical attack for SmallCipher, which was performed using a D-Wave simulator. We also estimated the total number of variables required to solve the full AES-128 cipher using the transformation of its algebraic equations to the QUBO problem. The total number of variables, in this case, is equal to $237,915$ and is much smaller than the total number of variables necessary for breaking RSA-3072, which ensures a similar level of security (approximately 128 bits). The total number of variables, in this case, is equal to approximately $2.36 \cdot 10^6$. In the discrete logarithm problem over the 3072-bits prime field, the total number of necessary variables is approximately $1.4 \cdot 10^{10}$. From this point of view, the presented approach of transformation of block ciphers to the QUBO problem and then solving this problem using D-Wave computer seems to be the most promising application of quantum annealing to cryptanalysis of cryptographic algorithms. Unfortunately, it is hard to estimate the time this process would require.

References

1. Gregory Bard. *Algebraic cryptanalysis.* Springer Science & Business Media, 2009.
2. Yu-Ao Chen and Xiao-Shan Gao. Quantum algorithms for boolean equation solving and quantum algebraic attack on cryptosystems. *arXiv preprint arXiv:1712.06239*, 2017.
3. Nicolas T Courtois and Gregory V Bard. Algebraic cryptanalysis of the data encryption standard. In *IMA International Conference on Cryptography and Coding*, pages 152–169. Springer, 2007.
4. Nicolas T Courtois and Willi Meier. Algebraic attacks on stream ciphers with linear feedback. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 345–359. Springer, 2003.

5. Nicolas T Courtois and Josef Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 267–287. Springer, 2002.
6. The Quantum Computing Company D-WAVE. Getting started with the d-wave system. *User manual*, 2020.
7. Raouf Dridi and Hedayat Alghassi. Prime factorization using quantum annealing and computational algebraic geometry. *Scientific reports*, 7:43048, 2017.
8. Tristan Greene. Google reclaims quantum computer crown with 72 qubit processor.
9. Shuxian Jiang, Keith A Britt, Alexander J McCaskey, Travis S Humble, and Sabre Kais. Quantum annealing for prime factorization. *Scientific reports*, 8(1):1–9, 2018.
10. Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.
11. Baonan Wang, Feng Hu, Haonan Yao, and Chao Wang. prime factorization algorithm based on parameter optimization of ising model. *Scientific Reports*, 10(1):1–10, 2020.
12. Michał Wroński. Solving discrete logarithm problem over prime fields using quantum annealing and $\frac{n^3}{2}$ logical qubits. Cryptology ePrint Archive, Report 2021/527, 2021. https://eprint.iacr.org/2021/527.