Applied Computer Science, vol. 20, no. 2, pp. 157-174
doi: 10.35784/acs-2024-22

Submitted: 2024-03-06 | Revised: 2024-06-07 | Accepted: 2024-06-14

Keywords: Diophantine equations, coin bag problem, decomposition algorithm, branch-and-bound method, authentication

Krzysztof NIEMIEC[*], *Grzegorz BOCEWICZ*[*]

AUTHENTICATION METHOD BASED ON THE DIOPHANTINE MODEL OF THE COIN BAG PROBLEM

Abstract

The article presents the so-called coin bag problem, which is modeled by linear Diophantine equations. The problem in question involves assessing the contents of a set of coins based on its weight. Since this type of problem is undecidable, a special variant of the problem was proposed for which effective problem-solving algorithms can be developed. In this paper, an original heuristic is presented (an algorithm based on problem decomposition) which allows to solve the coin bag problem in fewer steps compared to a brute force algorithm. The proposed approach was verified in a series of computational experiments. Additionally, an authentication scheme making use of the approach was proposed as an example of potential practical use.

1. INTRODUCTION

The problem of solving Diophantine equations regards the assessment of the existence of a solution to a multivariable equation in the integer domain (Mordell, 1969). There is a long history of research devoted to finding an algorithm that would allow solving Diophantine equations of any form (Clausen & Fortenbacher, 1989; Mordell, 1969; Stark, 1973). In the past, this problem had become so important that in 1900 it was put forth as one of the 23 most important challenges for 20th century mathematics (Hilbert's tenth problem (Kane, 2006)).

The decidability of the problem of solving Diophantine equations has been determined for selected classes of those problems. There are known algorithms for solving equations such as (the Pell equation, (Barbeau, 2003)).

$$ax + by = c \text{ (Clausen \& Fortenbacher, 1989; Gilbert \& Pathria, 1990)} \quad (1)$$

$$x^2 - ny^2 = 1 \quad (2)$$

[*] Koszalin University of Technology, Faculty of Electronics and Computer Science, Poland u18896@s.tu.koszalin.pl, grzegorz.bocewicz@.tu.koszalin.pl

The equation related to Fermat's Last Theorem, $x^n + y^n = z^n$, has turned out to have no natural solutions for $n > 2$ (Darmon et al., 1995; Wiles, 1995). In general, however, the problem of assessing whether a given Diophantine equation has a solution is in the class of undecidable problems (Matiyasevich, 1993). It has been proven that there cannot exist an algorithm for solving all Diophantine equations (Matiyasievich's theorem, (Matiyasevich, 1993)).

A special case of the problem of solving Diophantine equations is a problem in which there is a guarantee that the equation has at least one solution. In such situations, a time-effective method of determining the solution is sought. One example is the Diophantine equation which models the problem of assessing the value of coins based on their weight – the so-called coin bag problem.

In this problem, a set of coins of different denominations is given. The denominations in which the coins can be found are known. Each denomination has a corresponding coin weight, and these values are also known. The goal is to find the answer to the following question: Can one determine the contents of a set of coins (the number of coins of each denomination) knowing the total weight of this set (weight M of all the coins comprising this set)? – Fig. 1.

Diophantine equations which model this type of problem usually have many solutions, only one of which is correct – the one that represents the actual contents of the set of coins. In this article, an algorithm has been proposed which can find the correct solution based on a repeated decomposition of the problem. The approach proposed here assumes that a Diophantine equation with multiple solutions can be reduced to a set of Diophantine equations with one solution each. Such a decomposition allows one to determine the solution much faster than the brute force algorithm dedicated to the considered problem.

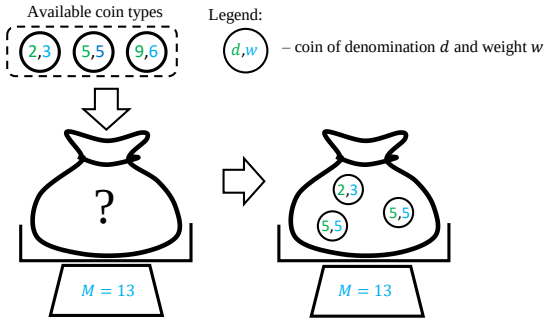


Fig.1. The coin bag problem

The problem considered in this work is one of the numerous "coin weighing" problems described in the literature. These problems are used as representations of special cases of combinatorial problems. They include:

- The counterfeit coin problem (Schell, 1945) – Given is a set of seemingly identical coins, one of which is fake and differs in weight from the others. The goal is to find the counterfeit coin using a minimum number of weighings on an ordinary balance scale with no weights.
- The Frobenius problem (Tripathi, 1978) – Given is a set of denominations. The goal is to determine the largest monetary amount that *cannot* be spent using coins of the

given denominations (the problem is also known in pop culture as the *Chicken McNugget Problem* (Picciotto, 1987).
- The coin change problem (Wright, 1975) – Given is a set of denominations. The goal is to find the smallest number of coins needed to make change for the given amount of money.
- The defective coin problem (Brody & Verbin, 2010) – Given is one unbalanced coin which produces an uneven number of heads/tails results when flipped. The problem involves finding the probability of the coin landing either heads up or tails up in as few flips as possible.
- The coin weighing problem (Karimi et al., 2018) – Given is a set of coins with unknown weights that take values within a given range. Any subset of the coins can be weighed, and the goal is to determine the weight of each coin in a minimum number of steps.

The above problems concern the topic of "coin weighing", however, due to different assumptions, they require the development of dedicated algorithms. There are already many time-efficient solutions for each of them. Due to the undecidability of the considered *coin bag problem*, it is still open. For this reason, the aim of the research was to develop an algorithm to solve this type of problem.

The present paper is organized as follows: Section 2 provides a formulation of the problem. It discusses the rules of decomposition and shows that for each value of M, it is possible to precisely determine the contents of the examined set of coins. Section 3 describes the algorithm based on problem decomposition. Section 4 presents the results of a comparison of the proposed algorithm with a brute force algorithm. A practical application example is presented in Section 5, and the key conclusions and the main directions of future research are stated in the Section 6.

2. PROBLEM STATEMENT

Typical Diophantine equations can be formulated in the following form [3]:

$$f(x_1, x_2, \dots, x_n) = 0,\ x_i \in \mathbb{C} \tag{3}$$

This is a multivariable equation in the integer domain for which it is assessed the existence of a solution. Many practical problems can be reduced to systems of Diophantine equations - among them is also the *coin bag problem*. Formally, the *coin bag problem* considered here is stated as follows: Given is a finite set of coins $C = \{c_1, \dots, c_i, \dots, c_k\}$. Each coin c_i is given by two values: weight $w(c_i) \in W = \{w_1, w_2, \dots, w_j, \dots, w_n\}$ and denomination $d(c_i) \in D = \{d_1, d_2, \dots d_j, \dots d_n,\}$, where W is the set of admissible weights of the coins used and D is the set of their denominations.

The contents of set C are unknown, i.e. it is not known how many coins there are in set C and what coins they are. Coin set C is being weighed (it is assumed that the scale used for weighing is perfect). The result of this operation is the total weight M of all the coins in set C. The goal is to find an answer to the following question:

Is it possible to determine the contents of set C given only the total weight M of the coins in this set?

One way to model a decision problem formulated in this way is to use the Diophantine equation given by:

$$\sum_{i=1}^{n} w_i \times a_i = M \tag{4}$$

where:

w_i – weight of a coin of denomination d_i, $d_i \in \mathbb{N}$,

a_i – number of coins in set C having denomination d_i, $a_i \in \mathbb{N}$,

M – weight of all the coins in set C, $M \in \mathbb{N}$.

Example. Let $D = \{d_1 = 2, d_2 = 5, d_3 = 9\}$ be a set of denominations and let $W = \{w_1 = 3, w_2 = 5, w_3 = 6\}$ be a set of their weights. Given is a certain set of coins C. The total weight of the coins found in this set is $M = 13$. The goal is to find values a_1, a_2, a_3 which satisfy the equation:

$$3 \times a_1 + 5 \times a_2 + 6 \times a_3 = 13 \tag{5}$$

The solution to the above equation is the sequence $A = (1,2,0)$, which means that set C consist of: 1 coin of denomination 2, 2 coins of denomination 5 and 0 coins of denomination 9 – see Fig. 2.

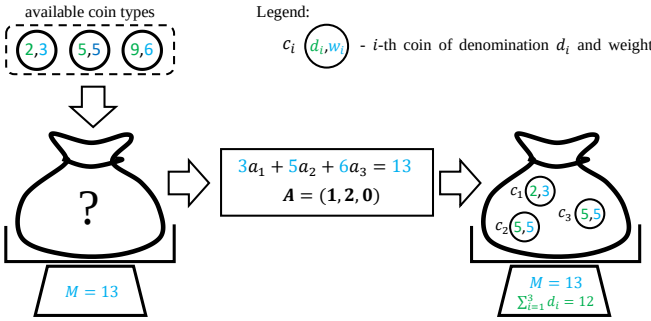


Fig. 2. An example of a solution to equation (2)

Under this model, the problem can be solved by answering the following question:

Does equation (2) have one solution in the set of natural numbers?

The solution to equation (2) is the sequence $A = (a_1, a_2 \ldots, a_i, \ldots a_n)$, wherein a_i denotes the number of coins of denomination d_i found in set C.

It turns out that for most values of M, the answer to this question is negative.

Example. Let D and W be sets describing currently circulating Polish coins, then:
$D = \{d_1 = 10, d_2 = 20, d_3 = 50, d_4 = 100, d_5 = 200, d_6 = 500\}$, (coin denominations are given in $grosz$[†])
$W = \{w_1 = 251, w_2 = 322, w_3 = 394, w_4 = 500, w_5 = 521, w_6 = 654\}$ (coin weights are given in $10^{-2} \cdot g$).

[†] „grosz" is a monetary unit equal to one hundredth of a zloty, the primary Polish monetary unit.

Consider two sets of coins being weighed: $C_1 = \{c_{1,1}, c_{1,2}, c_{1,3}, c_{1,4}\}$, $C_2 = \{c_{2,1}, c_{2,2}, c_{2,3}\}$. Set C_1 contains four twenty-grosz coins, and set C_2 contains two fifty-grosz coins and one one-zloty coin:

$$d(c_{1,1}) = d(c_{1,2}) = d(c_{1,3}) = d(c_{1,4}) = d_2 = 20,$$
$$d(c_{2,1}) = d(c_{2,2}) = d_3 = 50,\ d(c_{2,3}) = d_4 = 100.$$

The weights of the sets have the following values:

$$M_1 = w(c_{1,1}) + w(c_{1,2}) + w(c_{1,3}) + w(c_{1,4}) = 4 \times w_2 = 4 \times 322 = 1288$$
$$M_2 = w(c_{2,1}) + w(c_{2,2}) + w(c_{2,3}) = 2 \times w_3 + w_4 = 2 \times 394 + 500 = 1288$$

As one can see, the weights of the sets are identical, even though $C_1 \neq C_2$. In this example, equation (2) has two solutions – $A_1 = (0,4,0,0,0,0)$ and $A_2 = (0,0,2,1,0,0)$.

The example shows that it is not always possible to unequivocally determine the contents of a set of coins C, given solely the total weight M of the coins in the set.

It can be observed that there are infinitely many values of M, which do not provide a basis for unambiguously determining the contents of set C (different sets may have the same weight). Let $p(M)$ be a function describing the number of solutions to equation (2) for a given set $W = \{w_1, \ldots, w_i, \ldots, w_n\}$ and weight M. It can be observed that the function $p(M)$ always takes values greater than 1 for suitably large arguments M (Fig. 3).

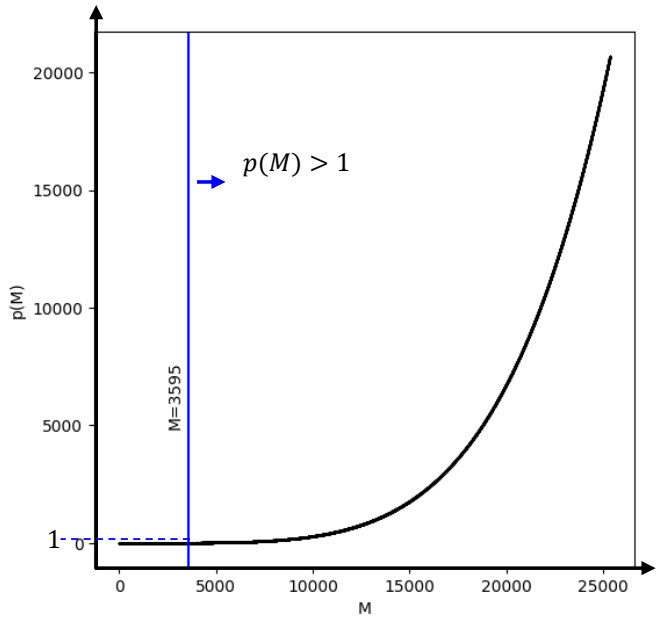


Fig. 3. Graph of $p(M)$ for the set $W = \{w_1 = 251, w_2 = 322, w_3 = 394, w_4 = 500, w_5 = 521, w_6 = 654\}$

For the set W used in the previous example (weights of Polish coins), the number of solutions $p(M)$ always takes a value greater than 1 for every $M > 3595$. This means that if the weight M of set C is greater than 3595, it is impossible to determine its contents. In other words, there is only a finite set of values of M for which the answer to the question stated in the problem statement is positive.

This observation leads to the formulation of a new variant of the problem which makes it possible to partition set C into smaller portions and weighing these portions rather than the whole set.

The operation of partitioning set C involves creating any number of non-empty, mutually disjoint subsets of this set, whose union is equal to set C (these subsets constitute a partition of set C) .

It can be proven that the operation of partitioning allows one to determine the contents of any given set C, as described by the following theorem.

Theorem 1. The contents of any set C containing k coins can be determined in no more than $2 \times k - 1$ weighings.

Proof: Let $k = 1$. Let $F(k)$ denote the number of weighings needed to determine the contents of set C containing k coins. The result of partitioning set C (for the given k) is the same set C, which guarantees that there is one coin in it. After the weighing operation, an unambiguous solution to equation (2) is obtained. Hence, $F(1) = 1$.

Let $k > 1$. Let us consider set $C' = C \cup \{x\}$, where C is a set containing $k - 1$ coins and x is any coin. We start by weighing set C' to obtain its weight (1 step), and then we decompose it into sets C and $\{x\}$. A solution for C is obtained in $F(k - 1)$ steps. A solution for $\{x\}$ is obtained in one step ($F(1) = 1$). If so, then a solution for C' is obtained in $F(k - 1) + 2$ steps.

We obtain a recursive relationship:

$$\begin{cases} F(1) = 1 \\ F(k) = F(k-1) + 2, \ \ k > 1 \end{cases} \tag{6}$$

the only solution to which is the function:

$$F(k) = 2 \times k - 1 \tag{7}$$

This means that, for any set C, it is enough to perform $2 \times k - 1$ weighings to determine its contents.

In this context, the contents of set C can always be determined by partitioning it into single-element subsets (containing one coin each) and weighing each of these subsets. Such an algorithm will hereinafter be referred to as the brute force algorithm.

3. THE UNIFORM DECOMPOSITION ALGORITHM

In this section, an algorithm is presented that can determine the contents of set C in fewer steps than the brute force algorithm. The algorithm in question involves partitioning set C, in each step, into two parts of similar weight (as opposed to the brute force algorithm, which partitions the set into a single-element subset and the remaining elements).

Let J be a set of values of M for which equation (2) has one solution: $J = \{M: p(M) = 1\}$. The idea of the proposed algorithm is to decompose set C until the weights M of all its subsets belong to set J. By meeting this condition, one can determine the contents of set C (sequence $A = (a_1, a_2, \dots, a_n)$) as the sum of the solutions for each of its subsets. The algorithm consists of the following steps:

1. Perform the weighing operation on C.
2. If $M \in J$, record the sequence $(a_1, a_2, \ldots, a_n)$ which is the solution.
3. If $M \notin J$, partition set C into subsets C_1 and C_2 with weights M_1 and M_2 so that $|M_1 - M_2|$ is as small as possible. Repeat the procedure for C_1 and C_2.
4. Add up the recorded sequences – the result of this operation is the content of C. The sum of sequences $A_1 = (a_{1,1}, a_{1,2}, \ldots, a_{1,n})$ and $A_2 = (a_{2,1}, a_{2,2}, \ldots, a_{2,n})$ is defined as sequence $(a_{1,1} + a_{2,1}, a_{1,2} + a_{2,2}, \ldots, a_{1,n} + a_{2,n})$.

To be able to use this algorithm, one must identify the set J. One way to determine this set is to use the generating functions method and analyze the coefficients of the following polynomial (Pólya, 1956).

$$T(x) = \prod_{i=0}^{n} \frac{1}{1-x^{w_i}} = (1 + x^{w_1} + \cdots + \ldots) \times (1 + x^{w_2} + x^{2\times w_2} + \ldots) \times (\ldots) \times (1 + x^{w_n} + x^{2\times w_n} + \ldots) = \sum_{M\in\mathbb{N}} p(M) \times x^M \quad (8)$$

Polynomial $T(x)$ describes all possible combinations of coins for the given set W. Each of the factors given by $(1 + x^{w_i} + x^{2\times w_i} + \ldots)$ specifies the possible number of occurrences in set C of a coin weighing w_i. Set C may contain zero coins of weight w_i (as represented by the free term 1), one coin weighing w_i (as represented by the term x^{w_i}), two coins weighing w_i (as represented by the term $x^{2\times w_i}$), etc. In general, k coins of weight w_i are represented by the term $x^{k\times w_i}$. The product of the factors $(1 + x^{w_i} + x^{2\times w_i} + \ldots) \times (1 + x^{w_j} + x^{2\times w_j} + \ldots)$ for two different coin weights (w_i and w_j), represents the admissible combinations of the occurrences of these coins in set C. For instance, zero coins of either weight are represented by the product of the terms (1×1); one coin of weight w_i and zero coins of weight w_j are represented by the product $(x^{w_i} \times 1)$; and two coins, each of which is a different weight, are represented by the product $(x^{w_i} \times x^{w_j} = x^{w_i + w_j})$. The exponent in each term of such a product represents the value M of the total weight of this combination.

The expression obtained after simplifying $T(x)$ contains information about the number of solutions to equation (4) for a given M. The coefficient at x^M is equal in value to the function $p(M)$. To determine the set J (which is a condition for using the proposed algorithm), it is enough to test all the values of $M \in \mathbb{N}$ and choose those, for which $p(M) = 1$. Because there are potentially infinitely many different values of M, the problem of determining set J is undecidable in the general case.

An example of a function $p(M)$ is shown in Fig. 4. Such a function grows exponentially in approximation (Matiyasevich, 1993). Numerical determination of the curve $p(M)$ is subject to some inaccuracy. The number of the terms of each of the factors $(1 + x^{w_i} + x^{2\times w_i} + \ldots)$ must be limited to a certain finite value (which means some possible combinations of weights w_i are ignored). As a result of this simplification, "distortions" start to appear on the curve as the value of M increases – some combinations of coin weights are not taken into account when the expression $T(x)$ is simplified, and thus the value of $p(M)$ is lowered. In such a case, the simplified curve takes the shape of a bell curve, which is different from the theoretical shape of the exponential curve (Fig. 4). The accuracy of the results of the discussed algorithm depends on the correct determination of set J, whose contents depend on the accurate determination of the value of $p(M)$.

It is worth emphasizing that at suitably low values of $M \leq M_{max}$, the value of $p(M)$ can be determined without error. It can therefore be assumed that set J is accurate for a set of coins whose weight does not exceed a certain value M_{max}.

Let $p_k(M)$ be a function corresponding to a function $p(M)$ for a polynomial $T(x)$ limited to k terms in each factor, i.e.:

$$T_k(x) = \left(1 + x^{w_1} + x^{2\times w_1} + \ldots + x^{(k-1)\times w_1}\right) \times (\ldots) \times \left(1 + x^{w_n} + x^{2\times w_n} + \ldots + x^{(k-1)\times w_n}\right) = \sum_{M\in\mathbb{N}} p_k(M) x^M \quad (9)$$

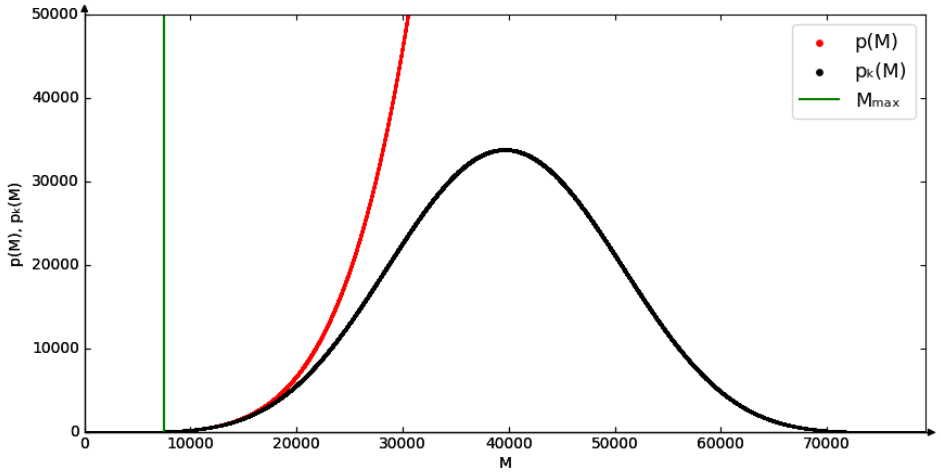


Fig. 4. Curves of functions $p(M)$ and $p_k(M)$ for $W = \{w_1 = 251, w_2 = 322, w_3 = 394, w_4 = 500, w_5 = 521, w_6 = 654\}$ and $k = 30$

It is worth noting that the polynomial $T_k(x)$ contains all representations of the sets containing fewer than k coins (the product of the factors given by $\left(1 + x^{w_i} + x^{2\times w_i} + \ldots + x^{(k-1)\times w_i}\right)$ takes into consideration all combinations of coins the total number of which is less than k). This fact can be used to obtain a given degree of curve accuracy, as described by Theorem 2:

Theorem 2. Let $w_{min} = min(w_1, w_2, w_3, \ldots, w_n)$. For each $M < k \times w_{min}$, functions $p_k(M)$ and $p(M)$ are equal: $p_k(M) = p(M)$.

Proof. Let C be a set with a weight M for which the above Theorem is false. This means that $M < k \times w_{min}$, $p_k(M) \neq p(M)$. Since $p_k(M)$ considers all sets containing fewer than k coins, it follows that $|C| \geq k$ – the value of M described in the above assumption must be the weight of a set containing k or more coins.

Let C_z denote any set containing exactly z coins, and let M_z denote the weight of this set. Note that $M_z \geq z \times w_{min}$ (the weight of the set must be greater or equal to the weight of z lightest coins).

$C = C_k \cup C_l, l \geq 0$ and $k + l = |C|$. Weight M_k satisfies the inequality $M_\mathrm{k} \geq k \times w_{min}$. Weight M_l satisfies the inequality $M_l \geq l \times w_{min} \geq 0 \times w_{min} \geq 0$. The weight of the set C therefore satisfies the following condition:

$$M = M_\mathrm{k} + M_l \geq k \times w_{min} + 0 \geq k \times w_{min} \quad (10)$$

This condition contradicts the previously accepted assumption. Since a contradiction was obtained, it can be concluded that there does not exist a value of $M < k \times w_{min}$ for which $p_k(M) \neq p(M)$, which concludes the proof.

The theorem can be used to determine such a value of weight M_{max} for which set J will consist of correct elements (weights M, for which $p(M) = 1$). To ensure that $p_k(M) = p(M), M \in \{0, 1, \dots, M_{max}\}$, one should find a k for which the inequality $M_{max} < k \times w_{min}$ is satisfied. After transformation, the inequality is given by $k > \frac{M_{max}}{w_{min}}$. It is equivalent to inequality $k > \left\lfloor \frac{M_{max}}{w_{min}} \right\rfloor$, and since k is a natural number, the smallest number that satisfies this inequality is $k \geq \left\lfloor \frac{M_{max}}{w_{min}} \right\rfloor + 1$.

Example

Let $W = \{w_1 = 2, w_2 = 3, w_3 = 5\}$ and let the weight of the set of coins C not exceed $M_{max} = 11$. To obtain accurate results under this constraint, the factors of the polynomial $T(x)$ must contain at least $k_{max} = \left\lfloor \frac{M_{max}}{w_{min}} \right\rfloor + 1 = \lfloor \frac{11}{2} \rfloor + 1 = 6$ terms. The polynomial $T_6(x)$ is then given by:

$$T_6(x) = (1 + x^2 + x^4 + x^6 + x^8 + x^{10}) \times (1 + x^3 + x^6 + x^9 + x^{12} + x^{15}) \times (1 + x^5 + x^{10} + x^{15} + x^{20} + x^{25}) = 1x^0 + 0x^1 + 1x^2 + 1x^3 + 1x^4 + 2x^5 + 2x^6 + 2x^7 + 3x^8 + 3x^9 + 4x^{10} + 4x^{11} + \cdots$$

The values of $p(M)$ obtained from the polynomial $T_6(x)$ are summarized in Table 1:

Tab. 1. Values of $p(M)x^M$ obtained from the polynomial $T_6(x)$

M	0	1	2	3	4	5	6	7	8	9	10	11	…
$p(M)$	1	0	1	1	1	2	2	2	3	3	4	4	…

The set of values M for which equation (4) has one solution ($p(M) = 1$) is, in this case, $J = \{0,2,3,4\}$.

Once set J has been determined, one can determine the contents of set C weighing $M < M_{max}$. For example, let $C = \{c_1, c_2, c_3, c_4, c_5\}$ and $w(c_1) = w(c_2) = w(c_3) = w(c_4) = 2, w(c_5) = 3$. The steps of the algorithm are listed below and shown in graphical form in Fig. 5:

Step 1 The result of the weighing operation C is 11. Since $11 \notin J$ (there is more than one solution), C has to be partitioned. The decomposition yields sets $C_1' = \{c_1, c_2, c_3\}$ and $C_2' = \{c_4, c_5\}$.

Step 2 The result of the operation of weighing C_1' is 6. $6 \notin J$, so the set needs to be partitioned. The decompositions yields sets $C_{1,1}' = \{c_1, c_2\}$ and $C_{1,2}' = \{c_3\}$.

Step 3 The result of the operation of weighing $C_{1,1}'$ is 4. $4 \in J$ – there exists only one combination with a weight of 4. The solution, in this case, is the sequence $A_{1,1}' = (2,0,0)$.

Step 4 The result of the operation of weighing $C_{1,2}'$ is 2. $2 \in J$ – this value has only one solution, which is the sequence $A_{1,2}' = (1,0,0)$.

Step 5 The result of the operation of weighing C_2' is 5. $5 \notin J$, so the set is subjected to decomposition. As a result of decomposition, sets $C_{2,1}' = \{c_4\}$ and $C_{2,2}' = \{c_5\}$ are obtained.

Step 6 The result of the operation of weighing $C'_{2,1}$ is 2 – this value has only one solution, which is the sequence $A'_{2,1} = (1,0,0)$.

Step 7 The result of the operation of weighing $C'_{2,2}$ is 3 – this value also has only one solution, which is $A'_{2,2} = (0,1,0)$.

Since the solutions for the sets created as an effect of the decomposition of C are known, they can be added together to obtain the solution for C: $A = A'_{1,1} + A'_{1,2} + A'_{2,1} + A'_{2,2} = (2,0,0) + (1,0,0) + (1,0,0) + (0,1,0) = (4,1,0)$. The sequence obtained in this way reflects the contents of set C.

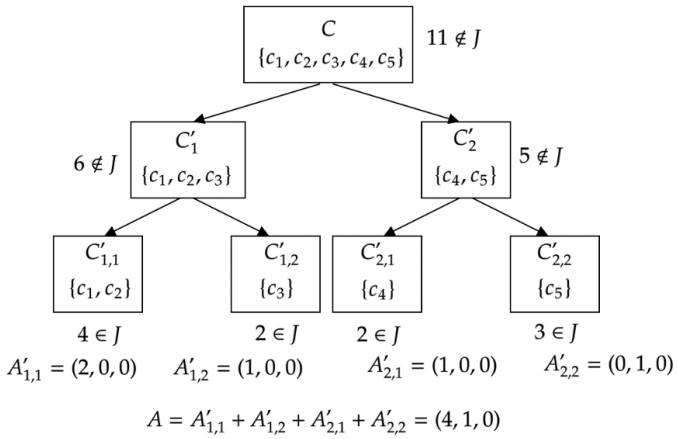


Fig. 5. A diagram showing an interpretation of the operation of the uniform decomposition algorithm based on the given example

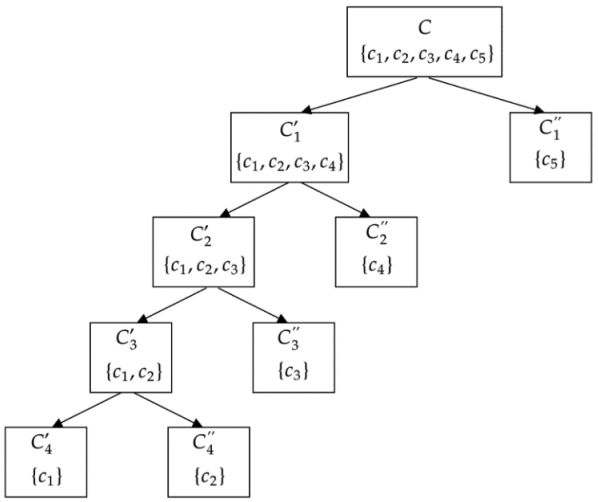


Fig. 6. A diagram showing an interpretation of the operation of the brute force algorithm based on the given example

It is worth noting that the decomposition tree for the brute force algorithm (Fig. 6) has more nodes than the uniform decomposition algorithm – the brute force algorithm will always partition set C into single elements, which is not always the case with the uniform decomposition algorithm (Fig. 5). According to equation (6), the number of steps of the brute force algorithm is $F(5) = 9$.

4. QUANTITATIVE EXPERIMENTS

The algorithm was tested by implementing it in the *Python* environment. During the experiments, the number of weighings (steps) performed by the brute force algorithm and the uniform decomposition algorithm were examined. Sets W were randomly generated. It was assumed that in the successive experiments, the elements of set W took values from the given set Q (the potential weight values were narrowed down to a certain range). Test cases (sets C to be subjected to decomposition) were also randomly generated. The successive sets C contained $k = 2, 3, \ldots, 99$ coins; for each value of k, 10 test cases were generated.

Fig. 7 shows a comparison of the results of the operation of the two algorithms for the case discussed in the previous examples - $W = \{251, 322, 394, 500, 521, 654\}$ (weights of currently circulating Polish coins). The black dots represent the number of weighings (number of algorithm steps) performed using the uniform decomposition algorithm, and the red dots indicate the number of weighings performed using the brute force algorithm. As it can easily be seen, the proposed decomposition algorithm requires fewer steps than the brute force algorithm. In an extreme case, for $M = 45000$ (the weight of the coins is $450\ g$), the number of weighings needed to identify the contents of set C totals 50 (the brute force algorithm requires 200 weighings to compute the same).

Figs. 8 through 10 show the results for the randomly generated sets W. Table 2 shows the experiment result for $k = 2, 3, \ldots, 29$ for ten sets W.

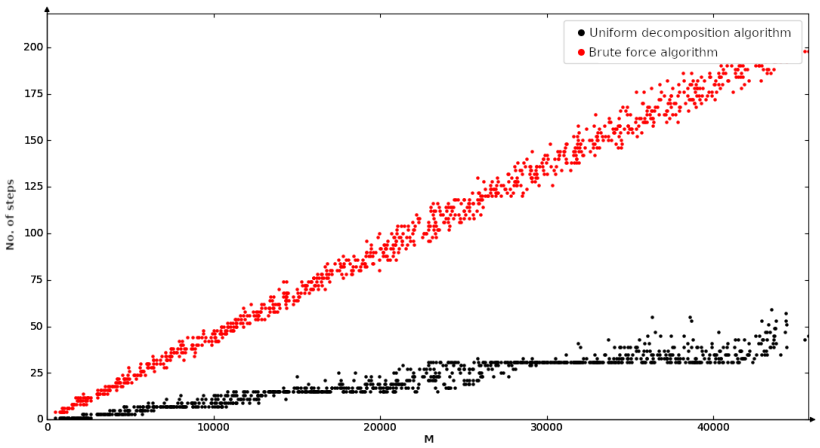


Fig. 7. Experiment result for $W = \{251, 322, 394, 500, 521, 654\}$

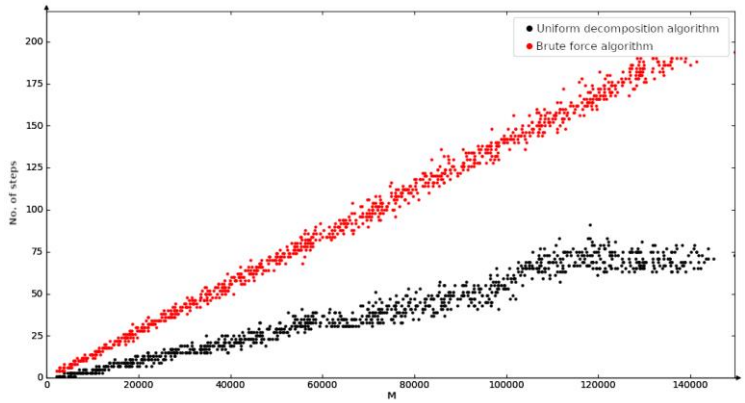


Fig. 8. Experiment result for $W = \{1261, 1077, 1814, 1792, 1662, 934, 1663, 1041, 989, 1997\}$, $Q = \{1000, 1001, \dots, 2000\}$

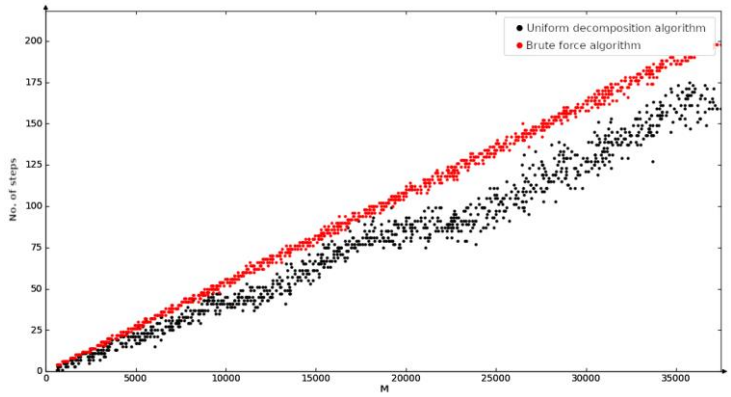


Fig. 9. Experiment result for $W = \{307, 339, 362, 333, 472, 313, 495, 322, 321, 374, 396, 393, 352, 464, 301, 403, 340, 345, 347, 443\}$, $Q = \{300, 301, \dots, 500\}$

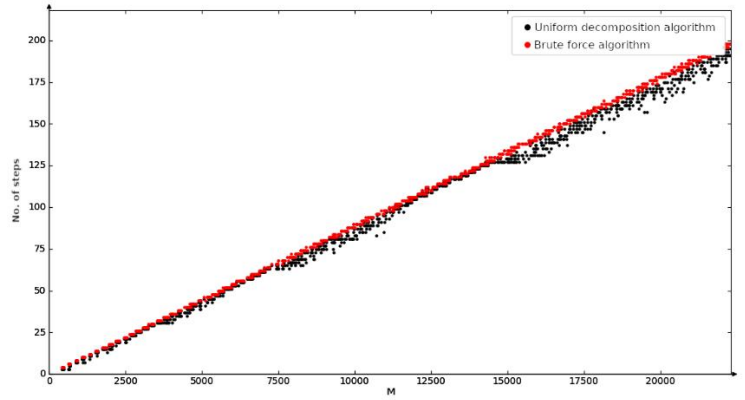


Fig. 10. Experiment result for W $\{218, 245, 231, 246, 234, 226, 212, 201, 223, 227, 225, 216, 215, 206, 222, 233, 249, 241, 224, 210\}$, $Q = \{200, 201, \dots, 300\}$

Tab. 2. Results of quantitative experiments

$\|W\|$	w_{min}	w_{max}	z/p	α_z	α_p	δ
19	230	1652	1.24737	1.89423	1.63935	0.13456
17	206	292	1.33367	1.89423	1.56272	0.17501
17	248	1284	1.51192	1.89423	1.40958	0.25586
14	207	496	1.84531	1.89423	1.14452	0.39579
16	410	1883	1.87511	1.89423	1.12072	0.40835
10	331	891	2.37705	1.89423	0.91151	0.5188
8	344	2064	2.52351	1.89423	0.91016	0.51951
9	228	693	2.53848	1.89423	0.84585	0.55346
7	398	1097	3.05425	1.89423	0.74172	0.60843
6	227	1365	3.25092	1.89423	0.73668	0.61109

$|W|$ – number of items in the set of weights W; w_{min} – the lowest weight in W; w_{max} – the highest weight in W; z/p – mean quotient of the numbers of weighings performed by the brute force algorithm and the uniform decomposition algorithm; α_z – mean number of weighings per coin performed by the brute force algorithm; α_p – mean number of weighings per coin performed by the uniform decomposition algorithm; δ – relative difference between α_z and α_p, calculated using the equation $\delta = \frac{\alpha_z - \alpha_p}{\alpha_z}$. The higher the value of δ, the greater the advantage of the uniform decomposition algorithm.

The experiments show that the uniform decomposition algorithm has an advantage over the brute force algorithm. For each examined set W, the novel approach proposed here turned out to be more efficient in determining the contents of set C. However, it can be observed that the degree of advantage of the uniform decomposition algorithm over the brute force algorithm varies depending on the size of set W. The degree of advantage is described numerically in Table 2 in column z/p (as a quotient of the number of steps performed by the tested algorithms) and column δ (as a difference in the number of steps per one coin). Both of these values increase as the size of set W becomes smaller. This means that the proposed algorithm is more efficient for smaller sets W. It is worth noting that the size of set W is not the only factor influencing the degree of advantage; the uniform decomposition algorithm is more efficient for sets W which contain values from a wider range (Table 2).

5. AN APPLICATION EXAMPLE

One of the potential applications of the algorithm is a step in an authentication procedure.

Let us assume that Alice and Bob share a password that has been securely generated using, for instance, the Diffie-Hellman method based on elliptic curves (Haakegaard & Lang, 2015). After some time, an attempt at communication is made, but Alice is not sure whether it is actually Bob who is trying to communicate with her. She can verify Bob's identity using the following procedure:

1. Give "Bob" instructions on how to convert the shared password into a set of weights W, without broadcasting the password itself (e.g. take groups of 4 bytes from the ASCII code representation of the password and treat each group as a weight).

2. Send "Bob" a solution A (in vector form) of an instance of the coin weighing problem with the previously established weights.
3. Both parties add up the solutions for A to a weight M using their weights and decompose it.
4. "Bob" says how many weighings were performed in his decomposition.
   a. If this number is not the same as Alice's, the procedure terminates: "Bob" has failed the authentication.
   b. If the number of weighings is the same, the test can be repeated, with Bob's credibility increasing with each test performed.

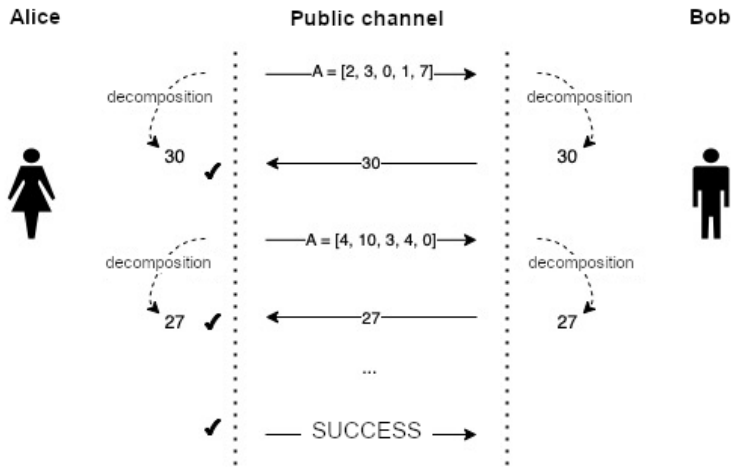


Fig. 11. Example of a successful authentication procedure

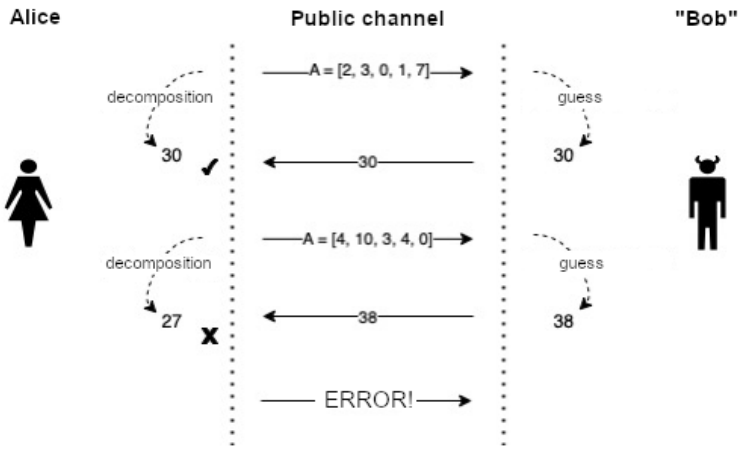


Fig. 12. Example of a failed authentication procedure

The test utilizes the low computational complexity of the decomposition procedure and the high computational complexity of the process of generating set J, which is necessary to perform a decomposition consistent with Alice's decomposition. If "Bob" is the real Bob, then he should know the correct set of weights and should be able to answer Alice's queries correctly every time. A potential attacker will have difficulty completing the procedure because:

- The set of weights is not provided in the public channel, and so, the attacker can, at best, try to deduce its contents,
- Set J can vary dramatically even if the changes made to the set of weights are small. Also, generating a new set J from scratch is computationally burdensome.
- Bob's response can in fact be any natural number, but only one response allows positive authentication, effectively defending the procedure against brute-force attacks. If the answer to the query is a natural number between 1 and k, and n queries have been sent, then, for $k = 50$ and $n = 3$, the probability of the successful authentication of a user who does not know the weights is $\frac{1}{k^n} = \frac{1}{125000}$.
- The number of decomposition steps as a function of weight M is highly susceptible to changes in set W (see Figs. 7–10), and the adjacent values of such a function differ from each other in an unpredictable way (which can be seen in Figs. 7–10 as a vertical "scatter" of values). This means that the knowledge of the number of steps for a certain M_i does not allow one to infer the number of steps for another M_j.

The above observations show the attractiveness of the potential use of the proposed procedure in authentication systems. The procedure is computationally simple for a party that knows the correct set of weights, and the chances of a potential attacker guessing the correct answers decrease dramatically with each repeated query.

An example of an implementation of the protocol described above may look as follows:

1. Let us consider two parties, “Alice” and “Bob”, analogous to the description at the beginning of the chapter.
2. First, both parties agree upon a secret (for example – a password). Let the password be the ASCII encoded string “di0ph4nt1n3”. This string is then converted to a set of integer values according to some rule – let us assume we take chunks from the password the size of a byte (8 digits of the binary representation of the password) and interpret them as an integer value (Fig. 13). The distinct obtained values form a set W.

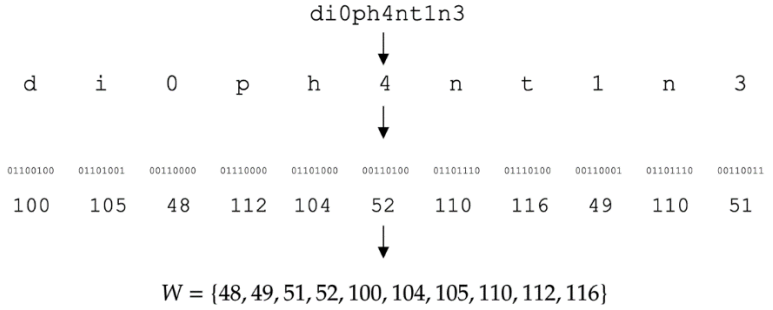


Fig. 13. Mapping a character string “di0ph4nt1n3” to a set W

3. Once the secret is established, both parties run a routine to generate a set J from the previously computed values in W. In the presented example, we obtain a 51-element set $J = \{48, 49, 51, 52, 96, 97, 98, 99, 101, 102, 103, \dots\}$ (Fig. 14).

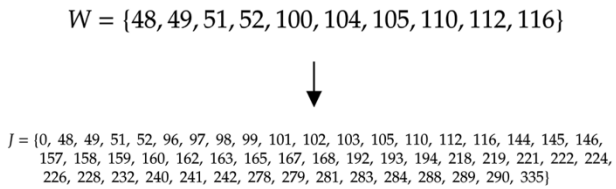
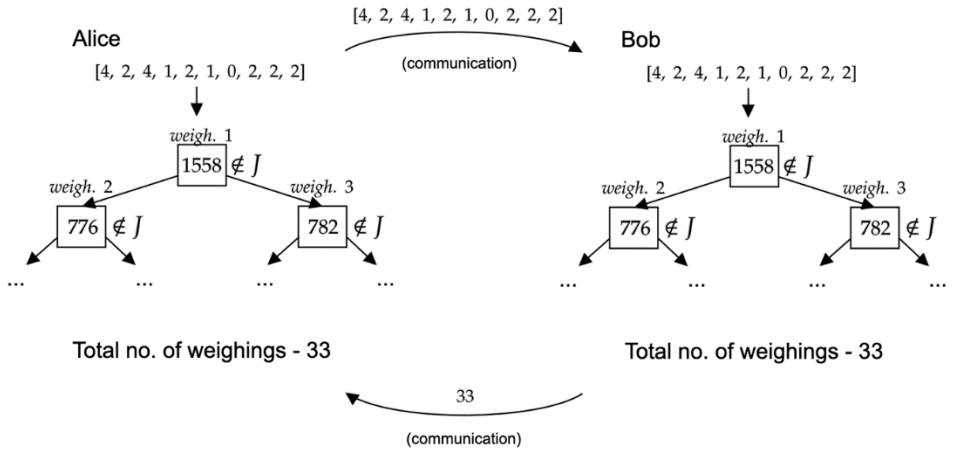


Fig. 14. The set W and its corresponding set J

4. Then, Alice may send queries to Bob consisting of a solution A for the coin bag problem. Both parties sum the solution to a weight M and compute its decomposition (Fig, 15). For example - query is the solution $[2, 6, 2, 0, 2, 3, 0, 3, 2, 0]$, corresponding to the weight $M = 1558$. Its uniform decomposition (as per the algorithm described in the article) has 33 weighings, and that is the response that Bob would have to send to Alice to successfully complete the authentication challenge.



The weighing trees (and consequently the number of weighings) match because the generated set J is the same. That number is shared, but the set J, W or the password are not

Fig. 15. The procedure of authentication between Alice and Bob after generating J

5. The number of weighings in the decomposition is then counted by both parties, and Bob sends the result to Alice.
6. The authentication challenge is successful, if and only if the results match. Then, Bob successfully proves to Alice that he possesses the password without revealing it (in plaintext or encrypted) over the network.

The proposed method allows parties to authenticate over a connection without revealing the shared secret. The entire procedure flow is shown in Fig. 16. As described earlier, the procedure is not computationally intensive and is reasonably secure. With refinement, it may present a plausible alternative to other methods (Azrour et al., 2021; Mumtaz et al., 2019) concerning such authentication or, for example, key agreement. Due to its lower computational costs for each query, the presented method may prove useful, for example, on devices with limited computational power - especially compared to schemes based on modular exponentiation, which are taxing on the central processing unit (Jonsson & Tornkvist, 2017).

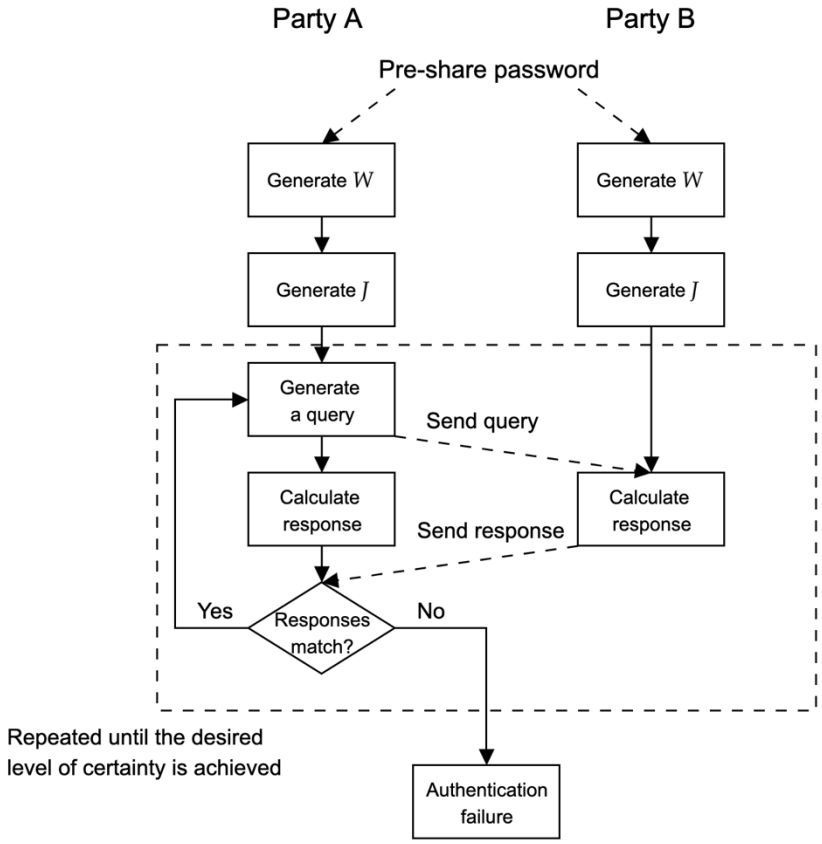


Fig. 16. The authentication procedure based on the coin bag problem, with details abstracted away

6. CONCLUSIONS

The article describes the so-called *coin bag problem*, which is analogous to the problem of solving linear Diophantine equations. In this article, a model of this problem was formulated, and it was shown that it is not always possible to determine a solution for the original version of the problem. Therefore, a variant of the problem was proposed, for which there is always a solution. The "uniform decomposition" heuristic was presented, which allows to determine the solution using fewer steps than the brute force algorithm. Tests were

also carried out which confirmed the advantage of the proposed algorithm over the brute force one. Additionally, a cryptographic procedure was presented that utilizes the properties of the examined problem for the purpose of authenticating parties in an IT system.

Prospects for further research include:

- The determination of an appropriate metric describing set W that would allow one to accurately predict the degree of advantage of the uniform decomposition algorithm over the brute force algorithm;
- Analysis of the impact of various ways of partitioning the set C on the efficiency of the algorithm (e.g. dividing the set into more than two parts).

REFERENCES

Azrour, M., Mabrouki, J., Guezzaz, A., & Farhaoui, Y. (2021). New enhanced authentication protocol for Internet of Things. *Big Data Mining and Analytics*, *4*(1), 1–9. https://doi.org/10.26599/BDMA.2020.9020010

Barbeau, A. (2003). *Pell's equation*. New York: Springer.

Brody, J., & Verbin, E. (2010). The Coin Problem and Pseudorandomness for Branching Programs. *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, 30–39. https://doi.org/10.1109/FOCS.2010.10

Clausen, M., & Fortenbacher, A. (1989). Efficient solution of linear diophantine equations. *Journal of Symbolic Computation*, *8*(1–2), 201–216. https://doi.org/10.1016/S0747-7171(89)80025-2

Darmon, H., Diamond, F., & Taylor, R. (1995). *Fermat's last theorem*. Current developments in mathematics.

Gilbert, W. J., & Pathria, A. (1990). *Linear Diophantine Equation*. Manuscript.

Haakegaard, R., & Lang, J. (2015). *The Elliptic Curve Diffie-Hellman (ECDH)*.

Jonsson, F., & Tornkvist, M. (2017). *RSA authentication in Internet of Things: Technical limitations and industry expectations*. Dissertation.

Kane, D. M. (2006). An elementary derivation of the asymptotics of partition functions. *The Ramanujan Journal*, *11*(1), 49–66. https://doi.org/10.1007/s11139-006-5307-x

Karimi, E., Kazemi, F., Heidarzadeh, A., & Sprintson, A. (2018). A Simple and Efficient Strategy for the Coin Weighing Problem with a Spring Scale. *2018 IEEE International Symposium on Information Theory (ISIT)*, 1730–1734. https://doi.org/10.1109/ISIT.2018.8437774

Matiyasevich, Y. (1993). *Hilbert's 10th Problem*. MIT press.

Mordell, L. J. (1969). *Diophantine Equations*. Academic Press.

Mumtaz, M., Akram, J., & Ping, L. (2019). An RSA Based Authentication System for Smart IoT Environment. *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, 758–765. https://doi.org/10.1109/HPCC/SmartCity/DSS.2019.00112

Picciotto, H. (1987). *Math McPuzzle*. *85*(52).

Pólya, G. (1956). On Picture-Writing. *The American Mathematical Monthly*, *63*(10), 689–697. https://doi.org/10.1080/00029890.1956.11988895

Schell, E. D. (1945). *Weighed and found wanting*. *52*(42).

Stark, H. (1973). *Effective estimates of solutions of some diophantine equations*. *24*(3), 251–259.

Tripathi, A. (1978). On a linear Diophantine problem of Frobenius. *Journal Für Die Reine Und Angewandte Mathematik (Crelles Journal)*, *1978*(301), 171–178. https://doi.org/10.1515/crll.1978.301.171

Wiles, A. (1995). *Modular elliptic curves and Fermat's last theorem*. Annals of mathematics.

Wright, J. W. (1975). The Change-Making Problem. *Journal of the ACM*, *22*(1), 125–128. https://doi.org/10.1145/321864.321874