Two Number-guessing Problems Plus Applications in Cryptography

Xingbo Wang
(Corresponding author: Xingbo Wang)
Department of Mechatronic Engineering,Foshan University
Guangdong Engineering Center of Information Security for Intelligent Manufacturing System
xbwang@fosu.edu.cn; dr.xbwang@qq.com
(Received Jan. 22, 2018; Revised and Accepted May 7, 2018; First Online Feb. 13, 2019)

Abstract

The article first puts forward two number-guessing problems that is to guess if an odd integer or one of its divisors is a divisor of another odd integer that is contained in an given odd sequence consisting in consecutive odd integers, then solves the two problems through an investigation on the global intrinsic properties of the odd sequence. Several criteria are proved to determine if a special term is contained in an odd sequence. Based on the proved criteria, algorithms are designed to detect if a interval contains a divisor-host that has a common divisor with a given number and to find out the divisor-host by means of probabilistic search. The theory and the algorithms are helpful in cracking a password or some encryption codes.

Keywords: Cryptography; Number Guessing; Password Cracking; Probabilistic Algorithm

1 Introduction

It is mandatory for a researcher of information security to guess the key of encrypted information. For example, it is necessary to simulate a crack on an encrypted system by guessing the password to see if the designed password is sure to be save enough, as J Lopez and H C Chou introduced in their articles [4] and [2]. In fact, such guessing games have been a primary approach in cryptography. In order to increase the probability of a successful encryption or decryption, kinds of mathematical methods are applied to the guessing process, as introduced and overviewed in articles [5] to [3]. Among the historical guessing approaches, the Pollard's rho method of factoring integers was a remarkably successful one because it was essentially a probabilistic algorithm called Monte Carol's approach, as analyzed in Bach's article [1].

A recent study comes across a problem that requires knowing an odd integer N with a nontrivial divisor that can divide another odd integer o that is hidden in a very large odd interval I_{odd}, which consists in finite consecutive odd integers. For example, suppose N is an odd composite integer; according to theorems proved in [7] and [8], one of N's divisor p can be found uniquely in an odd integer o lying in a *divisor-interval*, as the five odd composite integers and their respective divisor-intervals list in Table 1.

Due to a vast number of the odd integers contained in I_{odd}, a one-by-one sequential check is impossible to perform the detection of o, called a *divisor-host*, even with the fastest computer in the world. Hence the interval I_{odd} is subdivided into finite subintervals to be separately searched in parallel computation, as introduced in article [8]. Since there might be only one divisor-host o contained in one of the subdivided subintervals, solutions for the following two problems are necessary.

(P1). Is there a way to know which subinterval o lies in?

(P2). Is there a simple and fast way to locate o's position if the subinterval in which o lies is known?

The problems P1 and P2 are sure the kind of guessing-number problems if their background stated above is ignored. Moreover, they can be described in the following more general questions Q1 and Q2.

(Q1). Is there a way to know if an odd integer o itself or one of its divisorscan divide one of the odd integers in an odd interval I_{odd}?

(Q2). If an odd integer o itself or one of its divisors does divide one of the odd integers in an odd interval I_{odd}, is there a simple and fast way to detect the *divisor-host* that is divisible by o or one of o's divisors?

This article intends to answer the two questions Q1 and Q2 and explores their applications. By analyzing the intrinsic traits of consecutive odd integers that contain a special term, the article proves several criteria that can determine if o or one of o's divisors is contained in an odd interval or in an arithmetic progression. Based on the proved criteria, a fast way is developed to detect the

Table 1: Big integers and their divisor-intervals

Composite integer N	N**'s Divisor-interval**
$N_1 = 1123877887715932507$	[323935746324954482071652928163131137,323935746324954482071652929223262207]
$N_2 = 1129367102454866881$	[325517904753935056870231790741633615,325517904753935056870231791804350463]
$N_3 = 35249679931198483$	[317500890806149478235110120566155,317500890806149478235110308315135]
$N_4 = 208127655734009353$	[1499717812494687035718622130777530 9,1499717812494687035718622176398540 7]
$N_5 = 331432537700013787$	[477644625050956786725043756492054 87,477644625050956786725043762249072 63]

interval that contains o or one of o's divisors and a probabilistic approach is proposed to find out the integer that has a common divisor with o.

2 Symbols and Notations

In this whole article, an odd interval $[a, b]$, also called an odd sequence, is a set of consecutive odd numbers that take a as the lower bound and b as the upper bound, for example, $[3, 11] = \{3, 5, 7, 9, 11\}$. Symbol $\Sigma[a, b]$ is to express the sum of all terms on the close odd interval $[a, b]$, for example, $\Sigma[3, 11] = 3 + 5 + 7 + 9 + 11 = 35$. Symbol $\frac{\Sigma S}{n}$ is the arithmetic average value on odd interval S that consists in n odd integers, and symbol $\frac{\Sigma S}{\rho n} = \frac{1}{\rho} \times \frac{\Sigma S}{n}$ is called a ρ-enhanced arithmetic average value on S. Symbol $]a, b[$ is to express a set whose elements are either smaller than a or bigger than b, and thus $x \in]a, b[$ is equivalent to $x \leq a$ or $x \geq b$. Symbol $a|b$ means b is divisible by a and symbol $a \nmid b$ means b is not divisible by a. Symbol $A \Rightarrow B$ means conclusion B can be derived from condition A; $A \Leftrightarrow B$ means B holds if and only if A holds; the term 'if and only if' is also briefly written by notation *iff*, which also means 'the necessary and sufficient condition.

3 Main Results and Proofs

Theorem 1. *Suppose N is a composite odd integer with a divisor p bigger than 1, m and n are positive integers with $1 \leq m \leq n < p$; let $S = \{s_1, s_2, ..., s_n\}$ be a set that consists in n consecutive odd integers in which s_m is the unique term such that $p|s_m$ and $N \nmid s_m$; then there are at least $(n-m)(n-m+1)+1$ possible ways to compute $GCD(N, s_m)$ if $m \leq n < 2m-1$, and there are at least $m(m-1)+1$ possible ways to compute $GCD(N, s_m)$ if $n \geq 2m-1$.*

Proof. The given conditions immediately yield $GCD(s_m, N) = p$. Since S consists in n consecutive odd integers, when $n \geq 2m-1$ it knows that, there are $m-1$ terms on the left side of s_m and there are at least $m-1$ terms on its right side. This time, it yields the following facts.

1) There are $2(m-1)$ ways to obtain $2s_m$ by choosing s_{m-j} or s_{m+j} that fits $s_{m-j} + s_{m+j} = 2s_m$ with $j = 1, 2, ..., m-1$;

2) There are $2(m-2)$ ways to obtain $4s_m$ by choosing two consecutive terms, s_{m-j} and s_{m-j-1}, or two terms, s_{m+j} and s_{m+j+1}, that fit $s_{m-j} + s_{m-j-1} + s_{m+j} + s_{m+j+1} = 4s_m$ with $j = 1, 3, ..., m-2$. Considering that arbitrary symmetrically-distributed four terms, say s_{m-l}, s_{m-k},s_{m+l} and s_{m+k}, fit $s_{m-l} + s_{m-k} + s_{m+l} + s_{m+k} = 4s_m$, one knows there are at least $2(m-2)$ ways to obtain $4s_m$.

3) Likewise, there are at least $2(m-k)$ ways to obtain by choosing k consecutive terms, say s_{m-j-k}, $\cdots$, s_{m-j-1}, s_{m-j} or s_{m+j}, s_{m+j+1}, $\cdots$, s_{m+j+k} that fit $s_{m-j-k} + \cdots + s_{m-j-1} + s_{m-j} + s_{m+j} + s_{m+j+1} + \cdots + s_{m+j+k} = 2ks_m$ with $j = 1, \cdots, m-k-1$;

4) There is one way to obtain s_m, that is, to choose s_m itself.

Consequently, from 1 to k the minimal ways, denoted by Λ_k, which can produce multiples of s_m are given by

$$\begin{aligned}\Lambda_k &= 2(m-1) + 2(m-2) + \cdots + 2(m-k) + 1 \\ &= k(2m-k-1) + 1.\end{aligned}$$

And when $k = m-1$, it yields

$$\Lambda_{m-1} = m(m-1) + 1.$$

When $m \leq n < 2m-1$, there are $m-1$ terms on the left side of s_m while there are merely $n-m < m-1$ terms on its right side. Hence Λ_{n-m} is the biggest number for the case and consequently it yields

$$\begin{aligned}\Lambda_{n-m} &= 2(n-m) + ... + 2 + 1 \\ &= 2(\frac{(n-m+1)(n-m)}{2}) + 1 \\ &= (n-m)(n-m+1) + 1\end{aligned}$$

□

Corollary 1. *Suppose p is an odd prime number, m and n are positive integers with $1 \leq m < n < p$ and $n = 2m-1$; let $S = \{s_1, s_2, ..., s_n\}$ be a set consisting in n consecutive odd integers; then the necessary and sufficient condition of $p|s_m$ is $p|\Sigma S$, namely, $p|s_m \Leftrightarrow p|\Sigma S$.*

Proof. $n = 2m-1$ means s_m is the mid-term of S. $p|s_m$ yields $s_m = ps$ for some odd integer $s > 1$ and $s_{m-j} + s_{m+j} = 2s_m$ with $j = 1, 2, ..., m-1$. Consequently,

$$p|s_m \Rightarrow p|\Sigma S.$$

Since S consists in n consecutive odd integers, it knows

$$\begin{aligned}\Sigma S &= \Sigma[s_1, s_{m-1}] + s_m + \Sigma[s_{m+1}, s_{2m-1}] \\ &= (2m-1)s_m.\end{aligned} \tag{1}$$

Note that the condition that $1 \le m < n < p$ and $n = 2m-1$ yields $p > 2m-1$; hence $p \nmid 2m-1$ because of p's primality and as a result,

$$p|\Sigma S \Rightarrow p|s_m.$$

□

Corollary 2. *Suppose p is an odd prime number, m and n are positive integers with $1 \le m < n < p$ and $n = 2m-1$; Let $S = \{s_1, s_2, ..., s_n\}$ be an arithmetic integer sequence that consists in n consecutive terms; then the necessary and sufficient condition of $p|s_m$ is $p|\Sigma S$, namely, $p|s_m \Leftrightarrow p|\Sigma S$.*

Proof. (Omitted) □

Theorem 2. *Let S be an odd sequence that consists in n consecutive odd integers and s be an odd integer; then S contains s iff $ns-n(n-1) \le \Sigma S \le ns+n(n-1)$, whereas S does not contain s iff $\Sigma S \in]ns-n(n+1), ns+n(n+1)[$.*

Proof. The necessity. When containing s, S is given by

$$S = \{\underbrace{s-2k, ..., s-2, s, s+2, ..., s+2l}_{n \quad terms}\}$$

with $l, k \ge 0$ and $l+k+1 = n$; consequently it yields

$$\Sigma S = ns + (l-k)n.$$

Since $k+l+1 = n \Rightarrow k+l = n-1 \Rightarrow \begin{cases} l = n-1-k \\ k = n-1-l \end{cases}$, it yields

$$\Sigma S = \begin{cases} ns + n(n-1) - 2nk \\ ns - n(n-1) + 2nl \end{cases} \tag{2}$$

Considering the following two particular cases given by

$$S = \{\underbrace{s, s+2, ..., s+2(n-1)}_{n \quad terms}\} \Rightarrow \Sigma S = ns + n(n-1)$$

and

$$S = \{\underbrace{s-2(n-1), ..., s-2, s}_{n \quad terms}\} \Rightarrow \Sigma S = ns - n(n-1),$$

it yields

$$ns - n(n-1) \le \Sigma S \le ns + n(n-1).$$

When not containing s, S fits one of the following two cases

1) $S = \{\underbrace{s+2k, s+2k+2, ..., s+2k+2(n-1)}_{n \quad terms}\}$ with $k \ge 1$ leads to

$$\Sigma S = ns + n(n-1) + 2kn \tag{3}$$

2) $S = \{\underbrace{s-2l-2(n-1), ..., s-2l-2, s-2l}_{n \quad terms}\}$ with $l \ge 1$ leads to

$$\Sigma S = ns - n(n-1) - 2ln. \tag{4}$$

Note that, Case (1) turns to be $S = \{\underbrace{s+2, s+4, ..., s+2n}_{n \quad terms}\} \Rightarrow \Sigma S = ns+n(n+1)$ when $k = 1$ and case (2) turns to be $S = \{\underbrace{s-2n, ..., s-4, s-2}_{n \quad terms}\} \Rightarrow \Sigma S = ns - n(n+1)$ when $l = 1$; it immediately knows $\Sigma S \in]ns - n(n+1), ns + n(n+1)$.

Hence the necessity is true. Next is the proof for the sufficiency. Without loss of generality, let $S = \{s_1, s_2, ..., s_n\}$; then $s_n = s_1 + 2(n-1)$ and $\Sigma S = ns_1 + n(n-1)$. The condition $ns-n(n-1) \le \Sigma S \le ns+n(n-1)$ yields

$$\frac{\Sigma S}{n} - (n-1) \le s \le \frac{\Sigma S}{n} + (n-1).$$

Namely,

$$s_1 \le s \le s_1 + 2(n-1).$$

Hence it is sure that S contains s.

Now consider $\Sigma S \in]ns-n(n+1), ns+n(n+1)[$, namely, $\Sigma S \le ns - n(n+1)$ or $\Sigma S \ge ns + n(n+1)$. If $\Sigma S \le ns - n(n+1)$, it holds

$$\frac{\Sigma S}{n} + (n+1) \le s.$$

Substituting ΣS by $ns_1 + n(n-1)$ yields, namely

$$s_n = s_1 + 2(n-1) < s$$

which says S does not contains s when $\Sigma S \le ns-n(n+1)$.

Similarly, $\Sigma S \ge ns+n(n+1)$ yields $s < s_n - 2(n-1) = s_1$, which indicates S does not contains s.

Hence the sufficiency is also true. □

Theorem 2 can be alternatively stated as the following Theorem 3 and Theorem 4.

Theorem 3. *Let S be an odd sequence that consists in n consecutive odd integers and s be an odd integer; then S contains s iff $|\frac{\Sigma S}{n} - s| \le n-1$, and S does not contain s iff $|\frac{\Sigma S}{n} - s| \ge n+1$.*

Theorem 4. *Let S be an odd sequence that consists in n consecutive odd integers and s be an odd integer; then S contains s iff $|\frac{\Sigma S}{ns} - 1| \le \frac{n-1}{s}$, and S does not contain s iff $|\frac{\Sigma S}{ns} - 1| \ge \frac{n+1}{s}$.*

These theorems directly derive out the following corollaries.

Corollary 3. *Let S be an arithmetic integer sequence that consists in n terms with common difference d; suppose s is an integer number; if S contains s, then $ns-\frac{1}{2}n(n-1)d \leq \Sigma S \leq ns+\frac{1}{2}n(n-1)d$; if S does not contains s, then $\Sigma S \in]ns-\frac{1}{2}n(n+1)d, ns+\frac{1}{2}n(n+1)d[$.*

Proof of Corollary 3. (Omitted) □

Corollary 4. *Let $S_1 = \{s_1, s_2, ..., s_n\}$ and $S_2 = \{s_n + 2, s_n + 4, ..., s_n + 2n\}$ be two adjacent odd sequences each of which consists in n terms; then $\Sigma S_2 = \Sigma S_1 + 2n^2$.*

Proof. $s_n = s_1 + 2(n-1) \Rightarrow \Sigma S_1 = ns_1 + n(n-1)$. $\Sigma S_2 = ns_n + n(n+1) = ns_1 + n(n-1) + 2n^2 = \Sigma S_1 + 2n^2$. □

Corollary 5. *Suppose S is an odd sequence that consists in n consecutive odd integers and $s = pq$ is an integer with p and q being odd integers and $1 < n \leq p \leq q$; then S contains s iff $|\frac{\Sigma S}{np} - q| \leq \frac{n-1}{p}$, and S does not contain s iff $|\frac{\Sigma S}{np} - q| \geq \frac{n+1}{p}$. This indicates $\frac{\Sigma S}{np}$ is more closer to q if S contains $s = pq$.*

Proof. (Omitted) □

Corollary 6. *Suppose S is an odd sequence that consists in n consecutive odd integers and p is an odd integer that fits $1 < n \leq p$; if there exists an odd integer q that satisfies $|q - \frac{\Sigma S}{np}| \leq \frac{n-1}{p}$ then S contain $s = pq$.*

Proof. $|q - \frac{\Sigma S}{np}| \leq \frac{n-1}{p} \Rightarrow -\frac{n-1}{p} \leq q - \frac{\Sigma S}{np} \leq \frac{n-1}{p} \Rightarrow \frac{\Sigma S}{np} - \frac{n-1}{p} \leq q \leq \frac{\Sigma S}{np} + \frac{n-1}{p} \Rightarrow \Sigma S - n(n-1) \leq ns \leq \Sigma S + n(n-1)$. Since p and q are odd integers, $s = pq$ is an odd integer. By Theorem 2, it knows S contain $s = pq$. □

Corollary 5 and Corollary 6 result in the following Theorem 5.

Theorem 5. *Suppose S is an odd sequence that consists in n consecutive odd integers and p is an odd integer that fit $1 < n \leq p$; then there exists an odd integer q such that S contains $s = pq$ iff there exists an odd integer q that satisfies $|q - \frac{\Sigma S}{np}| \leq \frac{n-1}{p}$.*

Obviously, the computation of q is mandatory to take into consideration, as proposed by the following proposition.

Proposition 1. *Arbitrary integers p, q and n with $1 < n \leq p$ and $q \geq 1$, the inequality $|q - \frac{\alpha}{np}| \leq \frac{n-1}{p}$ yields with arbitrary real number α. Consequently, suppose S is an odd sequence consisting in n consecutive odd integers and p is an odd integer that fit $1 < n \leq p$; if there exists an odd integer q such that S contains $s = pq$ then $\left\lfloor \frac{\Sigma S}{np} \right\rfloor - 1 \leq q \leq \left\lfloor \frac{\Sigma S}{np} \right\rfloor + 1$.*

Proof. $\frac{\alpha}{np} - \frac{n-1}{p} \leq q \leq \frac{\alpha}{np} + \frac{n-1}{p} \Rightarrow \left\lfloor \frac{\alpha}{np} - \frac{n-1}{p} \right\rfloor \leq q \leq \left\lfloor \frac{\alpha}{np} + \frac{n-1}{p} \right\rfloor \Rightarrow \left\lfloor \frac{\alpha}{np} \right\rfloor - \left\lfloor \frac{n-1}{p} \right\rfloor - 1 \leq q \leq \left\lfloor \frac{\alpha}{np} \right\rfloor + \left\lfloor \frac{n-1}{p} \right\rfloor + 1$. Since $1 < n \leq p$ leads to $\left\lfloor \frac{n-1}{p} \right\rfloor = 0$, it knows

$$\left\lfloor \frac{\alpha}{np} \right\rfloor - 1 \leq q \leq \left\lfloor \frac{\alpha}{np} \right\rfloor + 1$$

□

4 Algorithms & Applications

By now, the questions Q1 and Q2 raised in the introductory part has been answered. For example, Theorem 2 shows how to know if an odd interval contains a special odd integer, Theorem 5 shows how to know if a divisor of an odd integer is a divisor of a term in an odd interval. Moreover, Theorem 1 indicates that, there is big probability to seek a unique term in a large odd interval, This provides a new applicable chance in factorization of odd composite integers by probabilistic approaches. Accordingly, this section presents a fast way to detect the divisor-interval that contains the divisor-host and introduces a probabilistic approach to search the divisor-host. Examples to solve the problems Q1 and Q2 are also given in this section.

4.1 Fast Detection of Divisor-interval

Let I_{odd} be a very large odd interval that contains the divisor host N_{host}; as is stated, a one-by-one check is difficult to locate N_{host}. Instead, a subdivision-based approach can reach an appreciated effect. The approach first subdivides I_{odd} into a proper number of subintervals, then calculates the arithmetic average value $\frac{\Sigma S}{n}$, s-enhanced arithmetic average value $\frac{\Sigma S}{ns}$ or p-enhanced arithmetic average value $\frac{\Sigma S}{np}$ on each subinterval. Since $n-1, \frac{n-1}{s}$ and $\frac{p-1}{p}$ are definitely known for a given n, s or p, it is easy to judge which subinterval contains N_{host}. Assuming L_{si} is set to be the biggest number of odd integers contained in each subinterval, the following algorithm can find the subinterval containing N_{host}. The process is described as a divisor-interval detecting algorithm (Algorithm 1).

Algorithm 1 Divisor-interval Detecting Algorithm

1: Begin
2: Input: Large odd interval I_{odd}, L_{si}, s (or p);
3: Step 1. Calculate N, the number of odd integers in I_{odd};
4: Step 2. Subdivide I_{odd} into $m+1$ subintervals, among which m ones are of equal length L_{si} and one is of length R by $N = mL_{si} + R$.
5: Step 3. Compare on each subinterval $\frac{\Sigma S}{L_{si}} - s$ with $n-1$ or $\frac{\Sigma S}{sL_{si}} - 1$ with $\frac{L_{si}-1}{s}$ for objective s, (or compute q with Algorithm 2 and then compare $\frac{\Sigma S}{pL_{si}} - q$ with $\frac{L_{si}-1}{p}$ for objective p), where ΣS is the sum on the respective subinterval.
6: Step 4. Choose the host-interval by Theorem 5* (or Corollary 5 for p).
7: End

Remark 1. *If the calculated objective is a divisor p, it is mandatory to calculate q first. This can be done as the following subroutine (Algorithm 2) shows.*

Algorithm 2 Subroutine: q's Calculation

1: Begin
2: Input: p, L_{si}, ΣS;
3: Step 1. Calculate $\varepsilon = \frac{L_{si}-1}{p}$;
4: Step 2. Calculate $q_i^{-1} = \left\lfloor \frac{\Sigma S}{pL_{si}} \right\rfloor - 1, q_i^0 = \left\lfloor \frac{\Sigma S}{pL_{si}} \right\rfloor$ and $q_i^{+1} = \left\lfloor \frac{\Sigma S}{pL_{si}} \right\rfloor + 1$ on each subinterval;
5: Step 3. Choose q to be an odd one from q_i^{-1}, q_i^0 and q_i^{+1} that satisfies $|q - \frac{\Sigma S}{np}| \le \varepsilon$
6: End

4.2 Probabilistic Approach To Find Out Divisor-host

Now that a divisor-interval is obtained with Algorithm 1, one can search the divisor-host by *the brutal search*, which searches the objective number one by one in the divisor-interval. It is known that, the efficiency of the brutal search depends on the length of the divisor-interval and sometimes a probabilistic search is faster than the brutal search. Therefore here introduces a probabilistic approach. According to Theorem 3, the probability to find the GCD will increase a lot by adding two or more odd integers in the divisor-interval. Thus, the probabilistic approach of searching the divisor-host can be deigned as Algorithm 3 shows.

Algorithm 3 Probabilistic Algorithm

1: Begin
2: Input: odd composite integer N and N's divisor-interval I_{odd}
3: Begin loop
4: Step 1. Select an integer $a \in I_{odd}$ randomly;
Select another integer $b \in I_{odd}$ randomly;
5: Step 2. d_1=FindGCD(N, a);d_2=FindGCD(N, b);
d_3=FindGCD(N, $a+b$);
6: Step 3. If $d_1 > 1$ then stop loop and return d_1;
If $d_2 > 1$ then stop loop and return d_2;
If $d_3 > 1$ then stop loop and return d_3;
7: End loop
8: End

Remark 2. *Algorithm 3 can also be a algorithm to factorize an odd integer. It can be applied only on the divisor-interval. Otherwise, it will fail absolutely.*

Applying Algorithm 3, big odd integers list in Table 1 are very quickly factorized by $N_1 = 299155897 \times 3756830131$, $N_2 = 25869889 \times 43655660929$, $N_3 = 59138501 \times 596052983$, $N_4 = 430470917 \times 483488309$ and $N_5 = 114098219 \times 2904800273$.

4.3 Comments & Examples

One can see that, the purpose of the algorithm 1 is to subdivide a big interval to find out the host-interval. The time complexity of the algorithm is $O(\frac{N}{L_{si}})$ with N being the number of terms in I_{obj} and L_{si} that is set up in advance and according to the computational capability of the computer performing the computation. Algorithm 2 is an auxiliary process of Algorithm 1 and its time complexity is $O(1)$. Algorithm 3 can be applied following Algorithm 1. It needs to point out that, Algorithm 3 here is merely a framework of the probabilistic approach. It needs improving in the way that picks the numbers randomly. The related work will be shown in a future paper.

In order for readers to understand the theory and the algorithms 1 and 2, here presents 2 examples which are also examples of the problems Q1 and Q2 with their solutions.

Example 1. *Let S_1, S_2 and S_3 be three sets each of which consists in 5 odd consecutive integers; suppose the sum of all the odd integers in each set is given by 4045, 4095 and 4145 respectively, judge which of S_1, S_2 and S_3 contains the number 825.*

Solution. The number *825* is the objective and each of S_1, S_2 and S_3 is a possible host-interval. $n = 5$ yields $n-1 = 4$. The arithmetic average values of S_1, S_2 and S_3 are respectively $\frac{4045}{5} = 809$, $\frac{4095}{5} = 819$ and $\frac{4145}{5} = 829$. Since $|809 - 825| = 16$, $|819 - 825| = 6$ and $|829 - 825| = 4$, it knows S_3 contains the objective *825* by Theorem 5*.

Example 2. *Let S={1133, 1135, 1137, 1139, 1141, 1143, 1145, 1147, 1149, 1151, 1153, 1155, 1157, 1159, 1161, 1163, 1165, 1167, 1169, 1171}; detect if 31 can be divisible one of the terms in S.*

Solution. The objective is $p = 31$ and $I_{obj} = S$. $N = 20$ is the number of terms in S. Subdivide S into 4 sub-sequences, (and thus $L_{si} = 5$), by

$$S_1 = \{1133, 1135, 1137, 1139, 1141\},$$

$$S_2 = \{1143, 1145, 1147, 1149, 1151\},$$

$$S_3 = \{1153, 1155, 1157, 1159, 1161\},$$

$$S_4 = \{1163, 1165, 1167, 1169, 1171\}.$$

Let $\varepsilon = \frac{L_{si}-1}{p} = \frac{5-1}{31} \approx 0.129$ and $E = \frac{L_{si}+1}{p} = \frac{5+1}{31} \approx 0.1935$; by Corollary 4, it needs to determine a q and check $|\frac{\Sigma S_i}{5\times 31} - q| \le \varepsilon$ for $i = 1, 2, 3, 4$. Note that,

$$\Sigma S_1 = 5685 \Rightarrow \frac{\Sigma S_1}{5 \times 31} \approx 36.6774$$

$$\Sigma S_2 = 5735 \Rightarrow \frac{\Sigma S_2}{5 \times 31} = 37.0000$$

$$\Sigma S_3 = 5785 \Rightarrow \frac{\Sigma S_3}{5 \times 31} \approx 37.3226$$

$$\Sigma S_4 = 5835 \Rightarrow \frac{\Sigma S_4}{5 \times 31} \approx 37.6452$$

By Proposition 1, q is 37.0000; then by Corollary 4, it knows that S_2 contains a term that is divisible by 31. In fact, 1147=31×37.

As a comparison, subdividing S into 5 sub-sequences, which leads to $L_{si} = 4$, results in

$$S_1 = \{1133, 1135, 1137, 1139\},$$

$$S_2 = \{1141, 1143, 1145, 1147\},$$

$$S_3 = \{1149, 1151, 1153, 1155\},$$

$$S_4 = \{1157, 1159, 1161, 1163\},$$

$$S_5 = \{1165, 1167, 1169, 1171\},$$

Referring to Corollary 4, let $\varepsilon = \frac{L_{si}-1}{p} = \frac{4-1}{31} \approx 0.0968$ and $E = \frac{L_{si}+1}{p} = \frac{5+1}{31} \approx 0.1613$; then

$$\Sigma S_1 = 4544 \Rightarrow \frac{\Sigma S_1}{4 \times 31} \approx 36.6452$$

$$\Sigma S_2 = 4576 \Rightarrow \frac{\Sigma S_2}{4 \times 31} \approx 36.9032$$

$$\Sigma S_3 = 4608 \Rightarrow \frac{\Sigma S_3}{4 \times 31} \approx 37.1613$$

$$\Sigma S_4 = 4640 \Rightarrow \frac{\Sigma S_4}{4 \times 31} \approx 37.4194$$

$$\Sigma S_5 = 4672 \Rightarrow \frac{\Sigma S_5}{4 \times 31} \approx 37.6774$$

Obviously, by Proposition 1, $q = 36 + 1 = 37$ is a reasonable choice and this time S_2 contains $s = 31 \times 37$ because $|\frac{\Sigma S_2}{4\times 31} - q| \approx 0.09677 \leq \varepsilon$ while $|\frac{\Sigma S_3}{4\times 31} - q| \approx 0.1613 \approx E$ and $|\frac{\Sigma S_1}{4\times 31} - q| \approx 0.3548 > E$.

5 Conclusions and Future Work

Finding a hidden objective in a set is meaningful in both mathematics and engineering. The set of consecutive odd integers, as a special set in cryptography, hides many problems inside. It is worthy of a subtitle study of the set. This paper solves the problem to detect if a number itself or one of its divisors is a divisor of another number that hidden in a set. It is helpful in cracking a password or some encryption codes. The probabilistic approach raised in this paper is sure a new idea to factorize odd integers because its way of finding GCD by adding randomly-picked items is quite different from the present Pollard's rho method that finds GCD by subtracting randomly-picked items. However, as stated before, the algorithm presented in this paper needs a further investigation. This is part of our future work. Hope it is concerned by more colleagues and valuable results come into being.

Acknowledgments

The research work is supported by the State Key Laboratory of Mathematical Engineering and Advanced Computing under Open Project Program No.2017A01, Department of Guangdong Science and Technology under projects 2015A030401105 and 2015A010104011, Foshan Bureau of Science and Technology under projects 2016AG100311, Projects 2014SFKC30 and 2014QTLXXM42 from Guangdong Education Department. The author sincerely presents thanks to them all.

References

[1] E. Bach, "Toward a theory of Pollard's Rho method", *Information and Computation*, vol. 90, pp. 139-155, 1991.

[2] H. C. Chou, H. C. Lee, H. J. Yu, *et al.*, "Password cracking based on learned patterns from disclosed passwords", *International Journal of Innovative Computing Information & Control*, vol. 9, no. 2, pp. 821-839, 2013.

[3] S. Houshmand, S. Aggarwal, R. Flood, "Next gen PCFG password cracking", *IEEE Transactions on Information Forensics & Security*, vol. 10, no. 8, pp. 1776-1791, 2017.

[4] J. Lopez, L. F. Cranor, N. Christin, *et al.*, "Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms", *Security & Privacy*, vol. 12, no. 02, pp. 523-537, 2012.

[5] S. Marechal, "Advances in password cracking", *Journal in Computer Virology*, no. 4, pp. 73-81, 2008. DOI10.1007/s11416-007-0064-y

[6] V. Vijayan, J. P. Joy, M. S. Suchithra, "A review on password cracking strategies", *International Journal of Computer and Communication Technology*, vol. 3, no. 3, pp. 8-15, 2014.

[7] X. Wang, "Genetic traits of odd numbers with applications in factorization of integers", *Global Journal of Pure and Applied Mathematics*, vol. 13, no. 1, pp. 318-333, 2017.

[8] X. Wang, "Strategy for algorithm design in factoring RSA numbers", *IOSR Journal of Computer Engineering*, vol. 19, no. 3(ver.2), pp. 1-7, 2017.

[9] C. M. Weir, "Using probabilistic techniques to aid in password cracking attacks", *PhD Dissertations*, 2010. (`http://purl.flvc.org/fsu/fd/FSU_migr_etd-1213`)

[10] F. Yu, Y. Huang, "An overview of study of passowrd cracking", in *International Conference on Computer Science and Mechanical Automation*, pp. 25-29, 2015.

Biography

Dr. & Prof. Xingbo WANG was born in Hubei, China. He got his Master and Doctor degrees at National University of Defense Technology of China and had been a staff

in charge of researching and developing CAD/CAM/NC technologies in the university. Since 2010, he has been a professor in Foshan University with research interests in computer application and information security. He is now the chief of Guangdong engineering center of information security for intelligent manufacturing system. Prof. WANG was in charge of more than 40 projects including projects from the National Science Foundation Committee, published 8 books and over 90 papers related with mathematics, computer science and mechatronic engineering, and invented 30 more patents in the related fields.