Data Hiding Techniques Using Prime and Natural Numbers

Sandipan Dey,
Cognizant Technology Solutions,
Kolkata, India
sandipan.dey@gmail.com

Ajith Abraham,
Centre for Quantifiable Quality of Service in Communication Systems,
Norwegian University of Science and Technology
O.S. Bragstads plass 2E, N-7491 Trondheim, Norway
ajith.abraham@ieee.org

Bijoy Bandyopadhyay,
Department of Radio Physics and Electronics,
University of Calcutta
Kolkata, India
b_bandyopadhyay@yahoo.com

Sugata Sanyal,
School of Technology and Computer Science
Tata Institute of Fundamental Research,
Homi Bhabha Road, Mumbai - 400005, India
sanyal@tifr.res.in

Abstract

In this paper, a few novel data hiding techniques are proposed. These techniques are improvements over the classical LSB data hiding technique and the Fibonacci LSB data-hiding technique proposed by Battisti et al. [1]. The classical LSB technique is the simplest, but using this technique it is possible to embed only in first few bit-planes, since image quality becomes drastically distorted when embedding in higher bit-planes. Battisti et al. [1] proposed an improvement over this by using Fibonacci decomposition technique and generating a different set of virtual bit-planes all together, thereby increasing the number of bit-planes. In this paper, first we mathematically model and generalize this particular approach of virtual bit-plane generation. Then we propose two novel
embedding techniques, both of which are special-cases of our generalized model. The first embedding scheme is based on decomposition of a number (pixel-value) in sum of prime numbers, while the second one is based on decomposition in sum of natural numbers. Each of these particular representations generates a different set of (virtual) bit-planes altogether, suitable for embedding purposes. They not only allow one to embed secret message in higher bit-planes but also do it without much distortion, with a much better stego-image quality, in a reliable and secured manner, guaranteeing efficient retrieval of secret message. A comparative performance study between the classical Least Significant Bit (LSB) method, the data hiding technique using Fibonacci -p-Sequence decomposition and our proposed schemes has been done. Theoretical analysis indicates that image quality of the stego-image hidden by the technique using Fibonacci decomposition improves against simple LSB substitution method, while the same using the prime decomposition method improves drastically against that using Fibonacci decomposition technique, and finally the natural number decomposition method is a further improvement against that using prime decomposition technique. Also, optimality for the last technique is proved. For both of our data-hiding techniques, the experimental results show that, the stego-image is visually indistinguishable from the original cover image.

Keywords

1 Introduction

Data hiding technique is a new kind of secret communication technology. It has been a hot research topic in recent years, and it is mainly used to convey messages secretly by concealing the presence of communication. While cryptography scrambles the message so that it cannot be understood, steganography hides the data so that it cannot be observed. The main objectives of the steganographic algorithms are to provide confidentiality, data integrity and authentication.

Most steganographic techniques proceed in such a way that the data which has to be hidden inside an image or any other medium like audio, video etc., is broken down into smaller pieces and they are inserted into appropriate locations in the medium in order to hide them. The aim is to make them un-perceivable and to leave no doubts in minds of the hackers who 'step into' media-files to uncover 'useful' information from them. To achieve this goal the critical data has to be hidden in such a way that there is no major difference between the original image and the 'corrupted' image. Only the authorized person knows about the presence of data. The algorithms can make use of the various properties of the image to embed the data without causing easily detectable changes in
them. Data embedding or water marking algorithms [3], [6], [7], [8], [14], [20] necessarily have to guarantee the following:

- Presence of embedded data is not visible.
- Ordinary users of the document/image are not affected by the watermark, i.e., a normal user does not see any ambiguity in the clarity of the document/image.
- The watermark can be made visible/retrievable by the creator (and possibly the authorized recipients) when needed; this implies that only the creator has the mechanism to capture the data embedded inside the document/image.
- The watermark is difficult for the other eavesdropper to comprehend and to extract them from the channels.

In this paper, we mainly discuss about using some new decomposition methods in a classical Image Domain Technique, namely LSB technique (Least Significant Bit coding, [18], [19], in order to make the technique more secure and hence less predictable. We basically generate an entirely new set of bit planes and embed data bit in these bit planes, using our novel decomposition techniques [40], [41].

For convenience of description, here, the LSB is called the 0\textsuperscript{th} bit, the second LSB is called the 1\textsuperscript{st} bit, and so on. We call the newly-generated set of bit-planes 'virtual', since we do not get these bit-planes in classical binary decomposition of pixels.

Rest of the paper is organized as follows: Sections 2 and 3 describe the embedding technique in classical LSB and Fibonacci decomposition technique with our modification. Section 4 describes a generalized approach that we follow in our novel data-hiding techniques using prime/natural number decomposition. Section 5 illustrates the embedding technique using the prime decomposition, while the experimental results obtained using this technique are reported in Section 6. In Section 7, we propose the natural number based embedding technique and the experimental results obtained are reported in Section 8. Finally, in Section 9 we draw our conclusions.

2 The Classical LSB Technique - Data Hiding by Simple LSB Substitution

Among many different data hiding techniques proposed to embed secret message within images, the LSB data hiding technique is one of the simplest methods for inserting data into digital signals in noise free environments, which merely embeds secret message-bits in a subset of the LSB planes of the image. Probability of changing an LSB in one pixel is not going to affect the probability of changing the LSB of the adjacent or any other pixel in the image. Data
hiding tools, such as Steganos, StegoDos, HideBSeek etc are based on the LSB replacement in the spatial domain [2]. But the LSB technique has the following major disadvantages:

- It is more predictable and hence less secure, since there is an obvious statistical difference between the modified and unmodified part of the stego-image.

- Also, as soon as we go from LSB to MSB for selection of bit-planes for our message embedding, the distortion in stego-image is likely to increase exponentially, so it becomes impossible (without noticeable distortion and with exponentially increasing distance from cover-image and stego-image) to use higher bit-planes for embedding without any further processing.

The workarounds may be: Through the random LSB replacement (in stead of sequential), secret messages can be randomly scattered in stego-images, so the security can be improved.

Also, using the approaches given by variable depth LSB algorithm [21], or by the optimal substitution process based on genetic algorithm and local pixel adjustment [4], one is able to hide data to some extent in higher bit-planes as well.

We propose two novel new data-hiding schemes by increasing the available number of bit-planes using new decomposition techniques. Similar approach was given using Fibonacci-p-sequence decomposition technique [1], [12], but we show the proposed decomposition techniques to be more efficient in terms of generating more virtual bit-planes and maintaining higher quality of stego-image after embedding.

3 Generalized Fibonacci LSB Data Hiding Technique

This particular technique, proposed by Battisti et al. [1], investigates a different bit-planes decomposition, based on the Fibonacci-p-sequences, given by,

\[
F_p(0) = F_p(1) = \ldots = F_p(p) = 1 \\
F_p(n) = F_p(n - 1) + F_p(n - p - 1), \forall n \geq p + 1, \; n, p \in \mathbb{N}
\]  

(1)

This technique basically uses Fibonacci-p-sequence decomposition, rather than classical binary decomposition (LSB technique) to obtain different set of bit-planes, embed a secret message-bit into a pixel if it passes the Zeckendorf condition, then while extraction, follow the reverse procedure.

We shall slightly modify the above technique, but before that let us first generalize our approach, put forward a mathematical model and then propose our new data-hiding techniques as special-cases of the generalized model.

For the proposed data hiding techniques our aim will be
• To expand the set of bit-planes and obtain a new different set of virtual bit-planes.
• To embed secret message in higher bit-planes of the cover-image as well, maintaining high image quality, i.e., without much distortion.
• To extract the secret message from the embedded cover-image efficiently and without error.

4 A Generalized LSB Data Hiding Technique

If we have k-bit cover image, there are only k available bit-planes where secret data can be embedded. Hence we try to find a function \( f \) that increases the number of bit-planes from \( k \) to \( n \), \( n \geq k \), by converting the k-bit 8-4-2-1 standard binary pixel representation to some other binary number system with different weights. We also have to ensure less distortion in stego-image with increasing bit plane. As is obvious, in case of classical binary decomposition, the mapping \( f \) is identity mapping. But, our job is to find a non-identity mapping that satisfies our end. Figure-1 presents our generalized model, while Figure-2 explains the process of embedding.

4.1 The Number System

We define a number system by defining the following:

• Base (radix) \( r \) (digits of the number system \( \in \{0, \ldots, r-1\} \))
• Weight function \( W(.) \), where \( W(i) \) denotes the weight corresponding to \( i^{th} \) digit (e.g., for 8-4-2-1 binary system, \( W(0) = 1, W(1) = 2, W(2) = 4, W(4) = 8 \)).

Hence, the pair \( (r, W(.) ) \), defines a number system completely. Obviously, our decimal system can be denoted in this notation as \((10, 10^1)\).

A number having representation \( d_{k-1}d_{k-2} \ldots d_1d_0 \) in number system \((r, W(.) )\) will have the following value (in decimal), \( D = \sum_{i=0}^{k-1} d_i W(i) \), \( d_i \in \{0, 1, \ldots, r-1\} \). This number system may have some redundancy if \( \exists \) more than one representation for the same value, e.g., the same (decimal) value \( D \) may be represented as \( d_{k-1}d_{k-2} \ldots d_1d_0 \) and \( d'_{k-1}d'_{k-2} \ldots d'_1d'_0 \), i.e., \( D = \sum_{i=0}^{k-1} d_i W(i) = \sum_{i=0}^{k-1} d'_i W(i) \), where \( d_i, d'_i \in \{0, 1, \ldots, r-1\} \). Here \( d_i \neq d'_i \) for at least 2 different \( i \)s.

To eliminate this redundancy and to ensure uniqueness, we should be able to represent one number uniquely in our number system. To achieve this, we must develop some technique, so that for number(s) having multiple (more than one, non-unique) representation in our number system, we can discard all representations but one. One way of doing this may be: from the multiple representations choose the one that has lexicographical highest (or lowest) value, discard all others. We shall use this shortly in case of our prime number system.
Figure 1: Basic block-diagram for generalized data-hiding technique
As shown in Figure-2, for classical binary number system (8-4-2-1), we use the weight function $W(.)$ defined by, $W(.) = 2^i \Rightarrow W : i \rightarrow 2^i \Rightarrow W(i) = 2^i$, $\forall i \in Z^+ \cup \{0\}$, corresponding to $i^{th}$ bit-plane (LSB = 0th bit), so that a k-bit number (k-bit pixel-value) $p_k$ is represented as $p_k = \sum_{i=0}^{k-1} b_{iC}2^i$, where $b_{iC} \in \{0, 1\}$ - this is our well-known binary decomposition.

Now, our $f$ converts this $p_k$ to some virtual pixel representation $p'_n$ (in a different binary number system) with $n$ (virtual) bit-planes, obviously we need to have $n \geq k$ to expand number of bit planes. But finding such $f$ is equivalent to finding a new weight function $W(.)$, so that $W(i)$ denotes the weight of $i^{th}$ (virtual) bit plane in our new binary number system, $\forall i \in Z^+ \cup \{0\}$. Mathematically, $p'_n = \sum_{i=0}^{n-1} b_{iC}.W(i)$, where $b_{iC} \in \{0, 1\}$ - this is our new decomposition, with the obvious condition that $(p_k)(2^i) = (p'_n)(2^W(i))$

Also, $W(i)$ must have less abrupt changes with respect to $i$, ($i^{th}$ bit plane, virtual), than that in the case of $2^i$, in order to have less distortion while embedding data in higher (virtual) bit planes. We call these expanded set of bit planes as virtual bit planes, since these were not available in the original cover image pixel data.

But, at the same time we must ensure the fact that the function $f$ that we use must be injective, i.e., invertible, unless otherwise we shall not be able to extract the embedded message precisely.
4.2 The Number System Using Fibonacci $p$-Sequence Decomposition

Function $f$ proposed by Battisti et al.[1] converts the pixel in binary decomposition to pixel in Fibonacci decomposition using generalized Fibonacci $p$-sequence, where corresponding weights are $F_p(n)$, $\forall n \in \mathbb{N}$, i.e., $W(.) = Fib_p(.)$, i.e., the number system proposed by them to model virtual bitplanes is $(2,F_p(.))$.

Since this number system too has redundancy (we can easily see it by applying pigeon-hole principle), for uniqueness and to make the transformation invertible, Zeckendorf’s theorem, has been used.

4.2.1 Modification to ensure uniqueness

Instead of Zeckendorf’s theorem, we use our lexicographically higher property. Hence, if a number has more than one representation using Fibonacci $p$-sequence decomposition, only the one lexicographically highest will be valid. Using this technique we prevent some redundancy also, since numbers in the range $[0, \sum_{i=0}^{n} F_p(i)]$ can be represented using $n$-bit Fibonacci-$p$-sequence decomposition. For an 8-bit image, the set of all possible pixel-values in the range $[0,255]$ has the corresponding classical Fibonacci ($p = 1$, Fibonacci-1-sequence, Fibonacci series ([10], [11], [13]) ) decomposition as shown in Table-1. One may use this map to have a constant-time Fibonacci decomposition from pixel values into 12 virtual bit-planes.

5 The Prime Decomposition Technique

5.1 The Prime Number System and Prime Decomposition

We define a new number system, and as before we denote it as $(2,P(.) )$, where the weight function $P(.)$ is defined as,

\[ P(0) = 1, \]
\[ P(i) = p_i, \forall i \in \mathbb{Z}^+, \] \hspace{1cm} (2)
\[ p_i = i^{th} \text{ Prime}, \]
\[ p_1 = 2, p_2 = 3, p_3 = 5, \ldots \]
\[ p_9 = 1 \]

Since the weight function here is composed of prime numbers, we name this number system as prime number system and the decomposition as prime decomposition.

As we have discussed earlier, if a number has more than one representation in our number system, we always choose the lexicographically highest of them as valid, e.g., ‘3’ has two different representations in 3-bit prime number system, namely, 100 and 011, since we have,
Table 1: Fibonacci (1-sequence) decomposition for 8-bit image yielding 12 virtual bit-planes
1.\( P(2) + 0.\ P(1) + 0.\ P(0) = 1.p2 + 0.p1 + 0.1 = 1.3 + 0.2 + 0.1 = 3 \)
\[ 
0.\ P(2) + 1.\ P(1) + 1.\ P(0) = 0.p2 + 1.p1 + 1.1 = 0.3 + 1.2 + 1.1 = 3
\]

100 being lexicographically (from left to right) higher than 011, we choose 100 to be valid representation for 3 in our prime number system and hence discard 011, which is no longer a valid representation in our number system.

\[ 3 \equiv \max_{\text{lexicographic}} (100, 011) \equiv 100. \]

Hence, for our 3-bit example, the valid representations are: 000 \( \equiv 0 \), 001 \( \equiv 1 \), 100 \( \equiv 2 \), 101 \( \equiv 3 \), 110 \( \equiv 4 \), 111 \( \equiv 5 \), 100 \( \equiv 6 \). Numbers in the range [0, 6] can be decomposed using our 3-bit prime number system uniquely, with only the representation 011 avoided.

Now, let us proceed with this very simplified example to see how the secret data bit is going to be embedded. We shall embed a secret data bit into a (virtual) bit-plane by just simply replacing the corresponding bit by our data bit, if we find that after embedding, the resulting representation is a valid representation in our number system, otherwise we do not embed, just skip. This is only to guarantee the existence of the inverse function and proper extraction of our secret embedded message bit.

Again, let us elucidate by our previous 3-bit example. Let the 3-bit pixel within which we want to embed secret data be of value 2, use prime decomposition to get 010, and we want to embed in the LSB bit-plane, let our secret message bit to be embedded be 1. So, we just replace the pixel LSB 0 by data bit 1 and immediately see that after embedding the pixel, it will become 011, which is not a valid representation, hence we skip this pixel without embedding our secret data bit.

Had we used this pixel value for embedding and after embedding ended up with pixel value 011 (value 3), we might get erroneous result while extraction of the secret bit. Because during extraction decomposition of embedded pixel value 3 would wrongly give 100 instead of 011, and extraction of LSB virtual bit-plane would wrongly give the embedded bit as 0 instead of its true value 1. Figure-3 explains this error pictorially.

Hence, embed secret data bit only to those pixels, where after embedding, we get a valid representation in the number system.

### 5.2 Embedding algorithm

- First we find the set of all prime numbers that are required to decompose a pixel value in a k-bit cover-image, i.e., we need to find a number \( n \in \mathbb{N} \) such that all possible pixel values in the range \([0, 2^k - 1]\) can be represented using first \( n \) primes in our \( n \)-bit prime number system, so that we get \( n \) virtual bit-planes after decomposition. We can use Sieve method, for example, to find primes. (To find the \( n \) is quite easy, since we see, using Goldbach conjecture etc, that all pixel-values in the range \([0, \sum_{i=0}^{m-1} p_i]\)
can be represented in our m-bit prime number system, so all we need to do is to find an n such that \( \sum_{i=0}^{n-1} p_i \geq 2^k - 1 \), since the highest number that can be represented in n-bit prime number system is \( \sum_{i=0}^{n-1} p_i \).

- After finding the primes, we create a map of k-bit (classical binary decomposition) to n-bit numbers (prime decomposition), \( n > k \), marking all the valid representations (as discussed in previous section) in our prime number system. For an 8-bit image the set of all possible pixel-values in the range \([0, 255]\) has the corresponding prime decomposition as shown in Table-2. As one may notice, the size of the map to be stored has been increased in this case, indicating a slightly greater space complexity.

- Next, for each pixel of the cover image, we choose a (virtual) bit plane, say \( p^{th} \) bit-plane and embed the secret data bit into that particular bit plane, by replacing the corresponding bit by the data bit, if and only if we find that after embedding the data bit, the resulting sequence is a valid representation in n-bit prime number system, i.e., exists in the map otherwise discard that particular pixel for data hiding.

- After embedding the secret message bit, we convert the resultant sequence in prime number system back to its value (in classical 8-4-2-1 binary number system) and we get our stego-image. This reverse conversion is easy, since we need to calculate \( \sum_{i=0}^{n-1} b_i.p_i \) only, where \( b_i \in \{0, 1\}, \forall i \in \{0, n - 1\} \)

### 5.3 Extraction algorithm

The extraction algorithm is exactly the reverse. From the stego-image, we convert each pixel with embedded data bit to its corresponding prime decomposition and from the \( p^{th} \) bit-plane extract the secret message bit. Combine all the bits to get the secret message. Since, for efficient implementation, we shall have a hash-map for this conversion, the bit extraction is constant-time, so the secret message extraction will be polynomial (liner) in the length of the message embedded.
<table>
<thead>
<tr>
<th>N</th>
<th>Prime Decomp</th>
<th>N</th>
<th>Prime Decomp</th>
<th>N</th>
<th>Prime Decomp</th>
<th>N</th>
<th>Prime Decomp</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100000000000001</td>
<td>92</td>
<td>111111111000010</td>
<td>83</td>
<td>111111111000000</td>
<td>84</td>
<td>111111111001010</td>
</tr>
<tr>
<td>26</td>
<td>111111111000010</td>
<td>130</td>
<td>111111111000000</td>
<td>130</td>
<td>111111111000000</td>
<td>131</td>
<td>111111111001010</td>
</tr>
<tr>
<td>24</td>
<td>111111111000000</td>
<td>132</td>
<td>111111111000000</td>
<td>132</td>
<td>111111111000000</td>
<td>133</td>
<td>111111111001010</td>
</tr>
<tr>
<td>25</td>
<td>111111111000000</td>
<td>134</td>
<td>111111111000000</td>
<td>134</td>
<td>111111111000000</td>
<td>135</td>
<td>111111111001010</td>
</tr>
<tr>
<td>28</td>
<td>111111111000000</td>
<td>136</td>
<td>111111111000000</td>
<td>136</td>
<td>111111111000000</td>
<td>137</td>
<td>111111111001010</td>
</tr>
<tr>
<td>10</td>
<td>1010101010101010</td>
<td>153</td>
<td>111111111000000</td>
<td>153</td>
<td>111111111000000</td>
<td>154</td>
<td>111111111001010</td>
</tr>
<tr>
<td>11</td>
<td>1010101010101010</td>
<td>155</td>
<td>111111111000000</td>
<td>155</td>
<td>111111111000000</td>
<td>156</td>
<td>111111111001010</td>
</tr>
<tr>
<td>12</td>
<td>1010101010101010</td>
<td>157</td>
<td>111111111000000</td>
<td>157</td>
<td>111111111000000</td>
<td>158</td>
<td>111111111001010</td>
</tr>
<tr>
<td>13</td>
<td>1010101010101010</td>
<td>159</td>
<td>111111111000000</td>
<td>159</td>
<td>111111111000000</td>
<td>160</td>
<td>111111111001010</td>
</tr>
<tr>
<td>14</td>
<td>1010101010101010</td>
<td>161</td>
<td>111111111000000</td>
<td>161</td>
<td>111111111000000</td>
<td>162</td>
<td>111111111001010</td>
</tr>
<tr>
<td>15</td>
<td>1010101010101010</td>
<td>163</td>
<td>111111111000000</td>
<td>163</td>
<td>111111111000000</td>
<td>164</td>
<td>111111111001010</td>
</tr>
<tr>
<td>16</td>
<td>1010101010101010</td>
<td>165</td>
<td>111111111000000</td>
<td>165</td>
<td>111111111000000</td>
<td>166</td>
<td>111111111001010</td>
</tr>
<tr>
<td>17</td>
<td>1010101010101010</td>
<td>167</td>
<td>111111111000000</td>
<td>167</td>
<td>111111111000000</td>
<td>168</td>
<td>111111111001010</td>
</tr>
<tr>
<td>18</td>
<td>1010101010101010</td>
<td>169</td>
<td>111111111000000</td>
<td>169</td>
<td>111111111000000</td>
<td>170</td>
<td>111111111001010</td>
</tr>
<tr>
<td>19</td>
<td>1010101010101010</td>
<td>171</td>
<td>111111111000000</td>
<td>171</td>
<td>111111111000000</td>
<td>172</td>
<td>111111111001010</td>
</tr>
<tr>
<td>20</td>
<td>1010101010101010</td>
<td>173</td>
<td>111111111000000</td>
<td>173</td>
<td>111111111000000</td>
<td>174</td>
<td>111111111001010</td>
</tr>
<tr>
<td>21</td>
<td>1010101010101010</td>
<td>175</td>
<td>111111111000000</td>
<td>175</td>
<td>111111111000000</td>
<td>176</td>
<td>111111111001010</td>
</tr>
<tr>
<td>22</td>
<td>1010101010101010</td>
<td>177</td>
<td>111111111000000</td>
<td>177</td>
<td>111111111000000</td>
<td>178</td>
<td>111111111001010</td>
</tr>
<tr>
<td>23</td>
<td>1010101010101010</td>
<td>179</td>
<td>111111111000000</td>
<td>179</td>
<td>111111111000000</td>
<td>180</td>
<td>111111111001010</td>
</tr>
<tr>
<td>24</td>
<td>1010101010101010</td>
<td>181</td>
<td>111111111000000</td>
<td>181</td>
<td>111111111000000</td>
<td>182</td>
<td>111111111001010</td>
</tr>
<tr>
<td>25</td>
<td>1010101010101010</td>
<td>183</td>
<td>111111111000000</td>
<td>183</td>
<td>111111111000000</td>
<td>184</td>
<td>111111111001010</td>
</tr>
<tr>
<td>26</td>
<td>1010101010101010</td>
<td>185</td>
<td>111111111000000</td>
<td>185</td>
<td>111111111000000</td>
<td>186</td>
<td>111111111001010</td>
</tr>
<tr>
<td>27</td>
<td>1010101010101010</td>
<td>187</td>
<td>111111111000000</td>
<td>187</td>
<td>111111111000000</td>
<td>188</td>
<td>111111111001010</td>
</tr>
<tr>
<td>28</td>
<td>1010101010101010</td>
<td>189</td>
<td>111111111000000</td>
<td>189</td>
<td>111111111000000</td>
<td>190</td>
<td>111111111001010</td>
</tr>
</tbody>
</table>

Table 2: Prime decomposition for 8-bit image yielding 15 virtual bit-planes
5.4 The performance analysis: Comparison between classical Binary, Fibonacci and Prime Decomposition

In this section, we do a comparative study between the different decompositions and its effect upon higher-bit-plane data-hiding. We basically try to prove our following two claims, by means of the following theorems from Number Theory [39]:

5.4.1 The Prime Number Theorem: A Polynomial tight bound for Primes

By Tchebychef theorem, \(0.92 < \frac{\pi(x) \ln(x)}{x} < 1.105\), \(\forall x \geq 2\), where \(\pi(x)\) denotes number of primes not exceeding \(x\), i.e., \(\pi(x) = \theta\left(\frac{x}{\ln(x)}\right)\). This leads to famous Prime Number theorem \(\lim_{n \to \infty} \left(\frac{\pi(n)}{(n/\ln(n))}\right) = 1\). From this one can show [1] that, if \(p_n\) be the \(n^{th}\) prime, \(\exists L_1, L_2 \in \mathbb{R}\), such that \(L_1 < \left(\frac{p_n}{(n/\ln(n))}\right) < L_2\), \(\forall n \geq 2\), \(n \in \mathbb{Z}^+\), i.e., \(\lim_{n \to \infty} \left(\frac{p_n}{(n/\ln(n))}\right) = 1\).

\[
\pi(n) = \theta(n \ln(n))
\]

5.4.2 A lower bound for the Fibonacci-p-Sequence

The Fibonacci-p-sequence, for \(p \geq 1\), \(p \in \mathbb{N}\), is given by,

\[
F_p(0) = F_p(1) = \ldots = F_p(p) = 1, \\
F_p(n) = F_p(n-1) + F_p(n-p-1), \forall n \geq p + 1, n \in \mathbb{N}
\]

We prove the following lemmas and find

**Lemma-1:** If the ratio of two consecutive numbers in Fibonacci p-sequence converges to limit \(\alpha_p \in \mathbb{R}^+\), \(\alpha_p\) satisfies the equation \(x^{p+1} - x^p - 1 = 0\), \(\forall p \in \mathbb{R}\).

**Proof:**

\[
\alpha_p = \lim_{n \to \infty} \left(\frac{f_{n+p}}{f_{n+p-1}}\right) = \lim_{n \to \infty} \left(\frac{f_{n+p-1}}{f_n}\right) = \ldots = \lim_{n \to \infty} \left(\frac{f_n}{f_{n-1}}\right) = \ldots,
\]

\(f_n = n^{th}\) number in the Fibonacci \(-p\) Sequence, \(f_{n+p} = f_{n+p-1} + f_{n-1}\)

\[
\Rightarrow \alpha_p = \lim_{n \to \infty} \left(\frac{f_{n+p-1} + f_{n-1}}{f_{n+p-1}}\right) = \lim_{n \to \infty} \left(\frac{f_{n}}{f_{n-1}}\right),
\]

\[
\Rightarrow \alpha_p = 1 + \lim_{n \to \infty} \prod_{k=n-1}^{k=n+p-2} \left(\frac{f_k}{f_{k+1}}\right) = \lim_{n \to \infty} \left(\frac{f_n}{f_{n-1}}\right)
\]

\[
\Rightarrow \alpha_p = 1 + \prod_{k=1}^{k=p} \left(\frac{1}{\alpha_p}\right) \Rightarrow \alpha_p = 1 + \frac{1}{\alpha_p}
\]
\[ \Rightarrow \alpha_p^{p+1} - \alpha_p^p - 1 = 0 \]

**Lemma-2:** If \( \alpha_p \) be a +ve root of the equation \( x^{p+1} - x^p - 1 = 0 \), we have \( 1 < \alpha_p < 2, \forall p \in \mathbb{N} \).

**Proof:** We have,

\[ \alpha_p^{p+1} - \alpha_p^p - 1 = 0 \text{ also, } 2^{p+1} - 2^p - 1 = 2^p - 1 > 0, \forall p \in \mathbb{Z}^+ \]
\[ \Rightarrow 2^p - 1 > \alpha_p^{p+1} - \alpha_p^p - 1 \Rightarrow (2^p - \alpha_p^p) > \alpha_p^p(\alpha_p - 2) \quad (4) \]

Also,
\[ -1 < 0 = \alpha_p^{p+1} - \alpha_p^p - 1 \Rightarrow \alpha_p^p(\alpha_p - 1) > 0 \Rightarrow \alpha_p > 1 \text{ (since positive)} \quad (5) \]

From (4), we immediately see the following:

- \( \alpha_p > 0 \) according to our assumption, hence we can not have \( \alpha_p = 2 \) (LHS & RHS both becomes 0, that does not satisfy inequality (4)).
- If \( \alpha_p > 2 \), we have LHS < 0 while RHS > 0 which again does not satisfy inequality (4).
- Hence we have \( \alpha_p < 2, \forall p \in \mathbb{N} \)

From (5), we have, \( \alpha_p > 1 \). Combining, we get, \( 1 < \alpha_p < 2, \forall p \in \mathbb{N} \)

**Lemma-3:** If \( \alpha_p \) be a +ve root of the equation \( x^{p+1} - x^p - 1 = 0 \), where \( p \in \mathbb{N} \), we have,

- \( \alpha_k > \alpha_{k+1} \)
- \( \alpha_{k+1} > \frac{1+\alpha_k}{2} \)
- \( \alpha_k^k < (k + 1), \forall k \in \mathbb{N} \)

**Proof:** We have,

For \( p = k \), \( \alpha_k^{k+1} - \alpha_k^k - 1 = 0 \)
For \( p = k + 1 \), \( \alpha_{k+1}^{k+2} - \alpha_{k+1}^{k+1} - 1 = 0 \)
\[ \Rightarrow \alpha_{k+1}^{k+1}(\alpha_{k+1} - 1) = \alpha_k^k(\alpha_k - 1) \]
\[ \Rightarrow \left( \frac{\alpha_k}{\alpha_{k+1}} \right)^k = \left( \frac{\alpha_{k+1} - 1}{\alpha_k - 1} \right).\alpha_{k+1} \quad (6) \]

From (6) we can argue,

- \( \alpha_k \neq \alpha_{k+1} \), since neither of them is 0 or 1 (from lemma-2).
• If \( \alpha_k < \alpha_{k+1} \), we have LHS of inequality (6) < 1, but RHS > 1, since both the terms in RHS will be greater than 1 (by our assumption and by lemma-2), a contradiction.

• Hence, we must have

\[
\alpha_k > \alpha_{k+1}, \quad \forall k \in \mathbb{N}
\]  

Again, from (6) we have,

\[
(k + 1) \frac{\alpha_k - 1}{\alpha_{k+1} - 1} \alpha_{k+1} > 1, \quad \text{from (7)}
\]

\[
\Rightarrow 2 > \alpha_{k+1} > \left( \frac{\alpha_k - 1}{\alpha_{k+1} - 1} \right), \quad \text{(from lemma-2)}
\]

\[
\Rightarrow \alpha_{k+1} > \frac{1 + \alpha_k}{2}
\]

Now, let us induct on \( p \) to prove \( \alpha_p < p+1 \).

Base case: for \( p = 1 \), \( \alpha_1 < 2 \), by lemma-2

Let us assume the inequality holds \( \forall p \leq k \Rightarrow \alpha_p < p+1 \) \( \forall p \leq k \)

Induction Step: for \( p = k+1 \), \( \alpha_{k+1} = \alpha_k \left( \frac{\alpha_k - 1}{\alpha_{k+1} - 1} \right) \), by (6)

\[
\Rightarrow \alpha_{k+1} < (k + 1). \left( \frac{\alpha_k - 1}{\alpha_{k+1} - 1} \right), \quad \text{by induction hypothesis}
\]

\[
\Rightarrow \alpha_{k+1} < (k + 1). \left( 1 + \frac{\alpha_k - \alpha_{k+1}}{\alpha_{k+1} - 1} \right)
\]

\[
\Rightarrow \alpha_{k+1} < (k + 1) + \left( \frac{\alpha_k - \alpha_{k+1}}{\alpha_{k+1} - 1} \right)
\]

\[
\Rightarrow \alpha_{k+1} < (k + 1) + 1, \quad \text{(from (8), we have,} \frac{\alpha_k - \alpha_{k+1}}{\alpha_{k+1} - 1} < 1)
\]

\[
\Rightarrow \alpha_{k+1} < (k + 2)
\]

\[
\Rightarrow \alpha_p < (p+1), \quad \forall p \in \mathbb{N}
\]

**Lemma-4**  The following inequalities always hold:

• \( (k+1)^{\frac{1}{k}} < k^{\frac{1}{k-1}} < \ldots < 4^{\frac{1}{3}} < 3^{\frac{1}{2}} < 2 \)

• \( \alpha_p < p+1 \Rightarrow \alpha_p < p \Rightarrow \ldots \alpha_3 < 4 \Rightarrow \alpha_2 < 3 \Rightarrow \alpha_p < 2 \)

**Proof:** By Binomial Theorem, we have,

\[
(k+1)^{k-1} = \sum_{n=0}^{k-1} \frac{(k-1)!}{n!(k-1-n)!} k^n = 1 + \sum_{n=1}^{k-1} \frac{1}{n!} \prod_{r=1}^{n} (1 - \frac{r}{k}). k^{k-1}
\]

\[
< \left( 1 + 1 + \ldots + 1 \right) k^{k-1} = k^k < (k+1)^{\frac{1}{k}}
\]

(10)
Hence, we have, \((k + 1)^\frac{1}{2} < k^{\frac{1}{k}} < \ldots < 4^{\frac{1}{4}} < 3^{\frac{1}{3}} < 2\)

Also, from (9) we have, \(\alpha_k < (k + 1)^{\frac{1}{2}}\).

Combining, we get,

\[\alpha_k < (k + 1)^{\frac{1}{2}} < k^{\frac{1}{k}} < \ldots < 4^{\frac{1}{4}} < 3^{\frac{1}{3}} < 2\]

\[\alpha_k < (k + 1)^{\frac{1}{2}} \Rightarrow \alpha_k^{k-1} < k \Rightarrow \alpha_k^4 < 5 \Rightarrow \alpha_k^3 < 4 \Rightarrow \alpha_k^2 < 3 \Rightarrow \alpha_k < 2 \quad (11)\]

**Lemma-5** The following inequality gives us the lower bound,

\[F_p(n) > \alpha_p^{n-p}, \forall n > p, \ n \in \mathbb{R}\]

where \(\alpha_p\) is the +ve root of the equation \(x^p+1 - x^p - 1 = 0\).

**Proof:** We induct on \(n\) to show the result.

\(F_p(0) = F_p(1) = \ldots = F_p(p) = 1\), (By definition of Fibonacci-p-Sequence).

Base case:

\(n = p + 1, \ F_p(p + 1) = F_p(p) + F_p(0) = 1 + 1 = 2 > \alpha_p\), (From Lemma-4)

\(n = p + 2, \ F_p(p + 2) = F_p(p + 1) + F_p(1) = 2 + 1 = 3 > \alpha_p^2\), (From Lemma-4)

\(n = p + 3, \ F_p(p + 3) = F_p(p + 2) + F_p(2) = 3 + 1 = 4 > \alpha_p^3\), (From Lemma-4)

\(\ldots\)

\(n = p + (p + 1), \ F_p(p + p + 1) = F_p(p + p) + F_p(p) = (p + 1) + 1 = p + 2 > \alpha_p^{p+1}\), (From Lemma-4)

Induction Step:

Let’s assume the above result is true \(\forall m < n, \ m, n \in \mathbb{R}\), for \(m > 2p + 1\) as well. Then we have,

\[F_p(n) = F_p(n - 1) + F_p(n - p - 1) > \alpha_p^{n-p-1} + \alpha_p^{n-p-2p-1} \quad (hypothesis)\]

\[\Rightarrow F_p(n) > \alpha_p^{n-2p-1}(1 + \alpha_p^p) = \alpha_p^{n-2p-1}\alpha_p^{p+1} = \alpha_p^{n-p}\]

\[\Rightarrow F_p(n) > \alpha_p^{n-p}, \ \forall n > p, \ n \in \mathbb{R}\]

Hence, we have the following inequality,

\[F_p(n) > (\alpha_p)^{n-p}, \quad \alpha_p \in \mathbb{R}^+;\]

\[\alpha_1 = \frac{1 + \sqrt{5}}{2} \approx 1.618034;\]

\[\alpha_2 \approx 1.465575;\]

\[\alpha_3 \approx 1.380278;\]

\[\alpha_4 \approx 1.324718;\]

\[\alpha_p > \alpha_{p+1}, \ \forall p \in \mathbb{Z}^+\]
\[
Fib(n) = \alpha^n - \beta^n
\]
\[
\beta = \frac{-1}{\phi}
\]
\[
\phi = \frac{1 + \sqrt{5}}{2}
\]

<table>
<thead>
<tr>
<th>$Fib(n)$</th>
<th>$\alpha^n$</th>
<th>$Fib(n)$</th>
<th>$\beta^n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>7</td>
<td>13</td>
<td>1</td>
<td>-1</td>
</tr>
</tbody>
</table>

Table 3: $\alpha$ is a $+ve$ Root of $x^2 - x - 1 = 0$, i.e., $\alpha \approx 1.618034$

The sequence $\alpha_p$ is decreasing in $p$.

The empirical results illustrated in Tables 3 and 4 also depict the same:

### 5.4.3 Measures

As we know, Security, embedding distortion and embedding rate can be used as schemes to evaluate the performance of the data hiding schemes. The following are the popular parameters,

- **Entropy** - A steganographic system is perfectly secure when the statistics of the cover-data and stego-data are identical, which means that the relative entropy between the cover data and the stego-data is zero. Entropy considers the information to be modeled as a probabilistic process that can be measured in a manner that agrees with intuition [38]. The information theoretic approach to steganography holds capacity of the system to be modeled as the ability to transfer information ([22], [23], [37]).

- **Mean Squared Error and SNR** - The (weighted) mean squared error between the cover image and the stego-image (embedding distortion) can be used as one of the measures to assess the relative perceptibility of the embedded text. Imperceptibility takes advantage of human psycho visual redundancy, which is very difficult to quantify. Mean square error (MSE) and Peak Signal to Noise Ratio (PSNR) can also be used as metrics to
**Table 4:** $\alpha_2$ is a +ve Root of $x^3 - x^2 - 1 = 0$, i.e., $\alpha_2 \approx 1.465571$

measure the degree of imperceptibility:

$$MSE = \sum_{i=1}^{M} \sum_{j=1}^{N} (f_{ij} - g_{ij})^2 MN$$

$$PSNR = 10 \log_{10} \left( \frac{L^2}{MSE} \right)$$

where $M$ and $N$ are the number of rows and number of columns respectively of the cover image, $f_{ij}$ is the pixel value from the cover image, $g_{ij}$ is the pixel value from the stego-image, and $L$ is the peak signal value of the cover image (for 8-bit images, $L = 255$. In general, for k-bit grayscale image, we have $L_k = 2^k - 1$). Signal to noise ratio quantifies the imperceptibility, by regarding the image as the signal and the message as the noise.

Here, we use a slightly different test-statistic, namely, Worst-case-Mean-Square-Error (WMSE) and the corresponding PSNR (per pixel) as our test-statistics. We define WMSE as follows:

If the secret data-bit is embedded in the $i^{th}$ bitplane of a pixel, the worst-case error-square-per-pixel will be $WSE = |W(i)(1 - 0)|^2 = (W(i))^2$, corresponding to when the corresponding bit in cover-image toggles in stego-image, after embedding the secret data-bit. For example, worst-case error-square-per-pixel for embedding a secret data-bit in the $i^{th}$ bit plane in case of a pixel in classical
binary decomposition is \( i = (2^i)^2 = 4^i \), where \( i \in \mathbb{Z}_+ \cup \{0\} \). If the original k-bit grayscale cover-image has size \( w \times h \), we define, \( WMSE = w \times h \times (W(i))^2 = w \times h \times WSE \). Here, we try to minimize this WMSE (hence WSE) and maximize the corresponding PSNR. We use the results (3) and (12) to prove our following claims:

### 5.4.4 The proposed Prime Decomposition generates more (virtual) bit-planes

Using Classical binary decomposition, for a k-bit cover image, we get only k bit-planes per pixel, where we can embed our secret data bit. From (3) and (12), we get,

- \( p_n = \theta(n, \ln n) \)
- \( \exists \alpha_p \in \mathbb{R}^+ : F_p(n) > (\alpha_p)^{n-1}, \alpha_p > \alpha_{p+1}, \forall p \in \mathbb{Z}_+, \alpha_1 \approx 1.618 \)

Since \( n, \ln n = o(\alpha_p^n) \), it directly implies that \( p_n = o(F_p(n)) \). The maximum (highest) number that can be represented in n-bit number system using our prime decomposition is \( \sum_{i=0}^{n-1} p_i \), and in case of n-bit number system using Fibonacci p-sequence decomposition is \( \sum_{i=0}^{n-1} F_p(i) \). Now, it is easy to prove that \( \exists n_0 \in \mathbb{N} : \forall n \geq n_0 \) we have, \( \sum_{i=0}^{n-1} F_p(i) > \sum_{i=0}^{n-1} p_i \).

Hence, using same number of bits it is possible to represent more numbers in case of the number system using Fibonacci-p-sequence decomposition, than that in case of the number system using prime decomposition, when number of bits is greater than some threshold. This in turn implies that number of virtual bit-planes generated in case of prime decomposition will be eventually (after some \( n \)) more than the corresponding number of virtual bit-planes generated by Fibonacci p-Sequence decomposition.

From the bar-chart shown in Figure-6, we see, for instance, to represent the pixel value 131, prime number system requires at least 12 bits, while for its Fibonacci counterpart 10 bits suffice. So, at the time of decomposition the same pixel value will generate 12 virtual bit-planes in case of prime decomposition and 10 for the later one, thereby increasing the space for embedding.

### 5.4.5 Prime Decomposition gives less distortion in higher bit-planes

Here, we assume the secret message length (in bits) is same as image size, for evaluation of our test statistics. For message with different length, the same can similarly be derived in a straightforward manner.

In case of our Prime Decomposition, WMSE for embedding secret message bit only in \( l^\text{th} \) (virtual) bitplane of each pixel (after expressing a pixel in our prime number system, using prime decomposition technique) = \( p_i^2 \), because change in \( l^\text{th} \) bit plane of a pixel simply implies changing of the pixel value by at most \( p_i \) prime number.

From the above discussion and using equation (3), also treating image-size as constant we can immediately conclude, (for \( l > 0 \))
Figure 4: Maximum number that can be represented in different decomposition techniques
\[(WMSE_{l^{th\ \text{bitplane}}}^{\text{prime-Decomposition}} = w \times h \times p_l^2 = \theta(l^2 \log^2(l)) \quad (13)\]

whereas WMSE in case of classical (traditional) binary (LSB) data hiding technique is given by,
\[(WMSE_{l^{th\ \text{bitplane}}}^{\text{Classical-Binary-Decomposition}} = \theta(4^l). \quad (14)\]

The above result implies that the distortion in case of prime decomposition is much less (since polynomial) than in case of classical binary decomposition (in which case it is exponential).

Now, let us calculate the WMSE for the embedding technique using Fibonacci p-sequence decomposition. In this case, WMSE for embedding secret message bit only in \(l^{th}\) (virtual) bit-plane of each pixel (after expressing it using Fibonacci-1-sequence decomposition) = \((F_p(l))^2\), because change in \(l^{th}\) plane of a pixel simply implies changing of the pixel value by at most \(l^{th}\) Fibonacci number.

From inequality (12), we immediately get that in case of \(p = 1\), i.e., for the Fibonacci-1-sequence decomposition, we have,
\[(WMSE_{l^{th\ \text{bitplane}}}^{\text{Fibonacci-1-Sequence\ Decomposition}} = (F(l))^2 = \theta ((2.618)^l)\]

Similarly, for other values of \(p\), one can easily derive (by induction) some exponential lower-bounds, which are definitely better than the exponential bound obtained in case of classical binary decomposition, but still they are exponential in nature, even if the base of the exponential lower bound will decrease gradually with increasing \(p\). So, we can generalize the above result by the following,
\[(WMSE_{l^{th\ \text{bitplane}}}^{\text{Fibonacci-p-Sequence\ Decomposition}} > \theta \left((\alpha_p^2)^l\right), \quad (15)\]

\[\alpha_p \in \mathbb{R}^+, \quad \alpha_1 = \frac{1 + \sqrt{5}}{2}, \quad \alpha_p > \alpha_{p+1}^2, \quad \forall p \in \mathbb{Z}^+.\]

The sequence \(\alpha_p^2\) is decreasing in \(p\). Obviously, Fibonacci-p-sequence decomposition, despite being better than classical binary decomposition, is still exponential and causes much-more distortion in the higher bit-planes, than our prime decomposition, in which case WMSE is polynomial (and not exponential!) in nature. The plot shown in Figure-5 proves our claim, it vindicates the polynomial nature of the weight function in case of prime decomposition against the exponential nature of classical binary and Fibonacci decomposition.

So from all above discussion, we conclude that Prime Decomposition gives less distortion than its competitors (namely classical binary and Fibonacci Decomposition) while embedding secret message in higher bit-planes.

At a glance, results obtained for test-statistic WMSE, for our k-bit cover image,
Figure 5: Weight functions for different decomposition techniques

\[
(WMSE_{l^{th} \text{bitplane}})_{\text{Classical Binary Decomposition}} = \theta(4^l).
\]

\[
(WMSE_{l^{th} \text{bitplane}})_{\text{Prime Decomposition}} = \theta(l^2 \log^2(l)).
\]

\[
(WMSE_{l^{th} \text{bitplane}})_{\text{Fibonacci} - p \text{ Decomposition}} = \theta \left( (\alpha_p)^l \right), \quad \alpha_p \in \mathbb{R}^+, 2.618 > \alpha_p > \alpha_{p+1}, \forall p \in \mathbb{Z}^+,
\]

with

\[
\text{Fibonacci} - 1 \text{ Decomposition} = \theta \left( (2.618)^l \right) \quad (15)
\]

Also, results for our test-statistic PSNR\(_\text{worst}\),

\[
\left( (\text{PSNR}_{\text{worst}})_{l^{th} \text{bitplane}} \right)_{\text{Binary Decomposition}} = 10 \log_{10} \left( \frac{(2^k - 1)^2}{(2^l)^2} \right),
\]

\[
\left( (\text{PSNR}_{\text{worst}})_{l^{th} \text{bitplane}} \right)_{\text{Prime Decomposition}} = 10 \log_{10} \left( \frac{(2^k - 1)^2}{c l^2 \log^2(l)} \right), \quad c \in \mathbb{R}^+. \quad (16)
\]

\[
\left( (\text{PSNR}_{\text{worst}})_{l^{th} \text{bitplane}} \right)_{\text{Fibonacci} - p \text{ Decomposition}} = 10 \log_{10} \left( \frac{(2^k - 1)^2}{(\alpha_p)^l} \right),
\]

\[
\alpha_p \in \mathbb{R}^+, 2.618 > \alpha_p > \alpha_{p+1}, \forall p \in \mathbb{Z}^+,
\]

with

\[
\left( (\text{PSNR}_{\text{worst}})_{l^{th} \text{bitplane}} \right)_{\text{Fibonacci} - 1 \text{ Decomposition}} = 10 \log_{10} \left( \frac{(2^k - 1)^2}{(2.618)^l} \right) \quad (16)
\]

6 Experimental Results for data-hiding technique using Prime decomposition

We have, as input:
- Cover Image: 8-bit (256 color) gray-level standard image of Lena.

- Secret message length = cover image size, (message string "sandipan" repeated multiple times to fill the cover image size).

- The secret message bits are embedded into one (selected) bit-plane per pixel only, the bitplane is indicated by the variable $p$.

- The test message is hidden into the chosen bitplane using different decomposition techniques, namely, the classical (traditional) binary (LSB) decomposition, Fibonacci 1-sequence decomposition and Prime decomposition separately and compared.

We get, as output:

- As was obvious from the above theoretical discussions, our experiment supported the fact that was proved mathematically.

- As obvious, as the relative entropy between the cover-image and the stego-image tends to be more and more positive (i.e., increases), we get more and more visible distortions in image rather than invisible watermark.

- Figure 6 illustrates gray level $[0 \ldots 255]$ vs. frequency plot of the cover image and stego image in case of classical LSB data-hiding technique. As seen from the figure, we get only 8 bit-planes and the frequency distribution (as shown in histograms) and hence the probability mass function [27] corresponding to gray-level values changes abruptly, resulting in an increasing relative entropy between cover-image and stego-image, implying visible distortions, as we move towards higher bit-planes for embedding data bits.

- Figure 7 shows gray level $[0 \ldots 255]$ vs. frequency plot of the cover image and stego image in case of data-hiding technique based on Fibonacci decomposition. This figure shows that, we get 12 bit-planes and the probability mass function corresponding to gray-level values changes less abruptly, resulting in a much less relative entropy between cover-image and stego-image, implying less visible distortions, as we move towards higher bit-planes for embedding data bits.

- Figure 8 again depicts gray level $[0 \ldots 255]$ versus frequency plot of the cover image and stego image in case of data-hiding technique based on Prime decomposition. This figure shows that, we get 15 bit-planes and the change of frequency distribution (and hence probability mass function) corresponding to gray-level values is least when compared to the other two techniques, eventually resulting in a still less relative entropy between the cover image and stego-image, implying least visible distortions, as we move towards higher bitplanes for embedding data bits.
• Data-hiding technique using the prime decomposition has a better performance than that of Fibonacci decomposition, the later being more efficient than classical binary decomposition, when judged in terms of embedding secret data bit into higher bit-planes causing least distortion and thereby having least chance of being detected, since one of principal ends of data-hiding is to go as long as possible without being detected.

• Using classical binary decomposition, we get here only 8 bit planes (since an 8-bit image), using Fibonacci 1-sequence decomposition we have 12 (virtual) bit-planes, and using prime decomposition we have still higher, namely 15 (virtual) bit-planes.

• As evident in Figure 9, distortion is highest in case of classical binary decomposition, less prominent in case of Fibonacci, and least for prime.

This technique can be enhanced by embedding into more than one (virtual) bit-plane, following the variable-depth data-hiding technique [21].

7 The Natural Number Decomposition Technique

For further improvement in the same line, we introduce a new number system and use transformation into that in order to get more (virtual) bit-planes, and
Figure 7: Frequency distribution of pixel gray-levels in different bit-planes before and after data-hiding in case of Fibonacci (1-sequence) decomposition technique.
Figure 8: Frequency distribution of pixel gray-levels in different bit-planes before and after data-hiding in case of Prime decomposition technique.
Figure 9: Result of embedding secret data in different bit-planes using different data-hiding techniques
also to have better image quality after embedding data into higher (virtual) bit-planes.

### 7.1 The Proposed Decomposition in Natural Numbers

We define yet another new number system, and as before we denote it as \((2, N(\cdot))\), where the weight function \(N(\cdot)\) is defined as, \(W(i) = N(i) = i+1, \forall i \in \mathbb{Z}^+ \cup \{0\}\)

Since the weight function here is composed of natural numbers, we name this number system as natural number system and the decomposition as natural number decomposition.

This technique also involves a lot of redundancy. Proving this is again very easy by using pigeonhole principle. Using \(n\) bits, we can have \(2^n\) different binary combinations. But, as is obvious and we shall prove shortly that using \(n\) bits, all (and only) the numbers in the range \([0, n(n + 1)/2]\), i.e., total \(\frac{n(n+1)}{2} + 1\) different numbers can be represented using our natural number decomposition. Since by induction one can easily show, \(2^n > 2\left(\frac{n(n+1)}{2}\right) + 1\), \(\forall n \geq 2, n \in \mathbb{N}\), we conclude, by Pigeonhole principle that, at least 2 representations out of \(2^n\) binary representations will represent the same value. Hence, we have redundancy.

As we need to make our transform one-to-one, what we do is exactly the same that we did in case of prime decomposition: if a number has more than one representation in our number system, we always take the lexicographically highest of them. (e.g., the number 3 has 2 different representations in 3-bit natural number system, namely, 100 and 011, since we have, \(1.3 + 0.2 + 0.1 = 3\) and \(0.3 + 1.2 + 1.1 = 3\). But, since 100 is lexicographically (from left to right) higher than 011, we choose 100 to be valid representation for 3 in our natural number system and thus discard 011, which is no longer a valid representation in our number system. \(3 \equiv \text{max}_{\text{lexicographic}}(100, 011) \equiv 100\) So, in our 3-bit example, the valid representations are: \(000 \leftrightarrow 0, 001 \leftrightarrow 1, 010 \leftrightarrow 2, 100 \leftrightarrow 3, 101 \leftrightarrow 4, 110 \leftrightarrow 5, 111 \leftrightarrow 6\) Also, to avoid loss of message, we embed secret data bit to only those pixels, where, after embedding we get a valid representation in the number system. It is worth noticing that, up-to 3-bits, the prime number system and the natural number system are identical, after that they are different.

#### 7.2 Embedding algorithm

- First, we need to find a number \(n \in \mathbb{N}\) such that all possible pixel values in the range \([0, 2^k - 1]\) can be represented using first \(n\) natural numbers in our \(n\)-bit prime number system, so that we get \(n\) virtual bit-planes after decomposition. To find the \(n\) is quite easy, since we see, and we shall prove shortly that, in \(n\)-bit Natural Number System, all the numbers in the range \([0, n(n + 1)/2]\) can be represented. So, our job reduces to finding an \(n\) such that \(\frac{n(n+1)}{2} \geq 2^k - 1\), i.e., solving the following quadratic in-equality

\[
n^2 + n - 2^{k+1} + 2 \geq 0,
\]
\[ n \geq -1 + \sqrt{2^{k+3} + 9}, \quad n \in \mathbb{Z}^+ \]  

(17)

- After finding \( n \), we create a map of \( k \)-bit (classical binary decomposition) to \( n \)-bit numbers (natural number decomposition), \( n > k \), marking all the valid representations (as discussed in previous section) in our natural number system. For an 8-bit image the set of all possible pixel-values in the range \([0, 255]\) has the corresponding natural number decomposition as shown in Table-5.

For \( k = 8 \), we get,

\[
n \geq -1 + \sqrt{2^8 + 3} + 9 = 22.675 \Rightarrow n = 23
\]

Hence, for an 8-bit image, we get 23 (virtual) bit-planes.

If we recapitulate our earlier result, as we see from the map shown in Table-2, in case of prime decomposition, it yields much less numbers of (virtual) bit planes (namely 15). Again it is noteworthy that the space to store the map is still increased. Although this computation of the map (one-time computation for a fixed value of \( k \)) is slightly more expensive and takes more space to store in case of our natural number decomposition than in case of prime decomposition, the first outperforms the later one when compared in terms of steganographic efficiency, i.e., in terms of embedded image quality, security (since number of virtual bit-planes will be more in case of the first) etc, as will be explained shortly.

- Next, for each pixel of the cover image, we choose a (virtual) bit plane, say \( p^{th} \) bit-plane and embed the secret data bit into that particular bit plane, by replacing the corresponding bit by the data bit, if and only if we find that after embedding the data bit, the resulting sequence is a valid representation in \( n \)-bit prime number system, i.e., exists in the map otherwise discard that particular pixel for data hiding.

- After embedding the secret message bit, we convert the resultant sequence in prime number system back to its value (in classical 8-4-2-1 binary number system) and we get our stego-image. This reverse conversion is easy, since we need to calculate \( \sum_{i=0}^{n-1} b_i(i+1) \) only, \( b_i \in \{0, 1\}, \forall i \in \{0, n - 1\} \).

### 7.3 Extracting algorithm

The extraction algorithm is exactly the reverse. From the stego-image, we convert each pixel with embedded data bit to its corresponding natural decomposition and from the \( p^{th} \) bit-plane extract the secret message bit. Combine all the bits to get the secret message. Since, for efficient implementation, we shall have a hash-map for this conversion, the bit extraction is constant-time, so the secret message extraction will be polynomial (linear) in the length of the message embedded.
<table>
<thead>
<tr>
<th>N</th>
<th>Natural Decomp</th>
<th>N</th>
<th>Natural Decomp</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00000000000000000000000000000000</td>
<td>94</td>
<td>11101111111111111111111111111111</td>
</tr>
<tr>
<td>1</td>
<td>00000000000000000000000000000001</td>
<td>95</td>
<td>11101111111111111111111111111111</td>
</tr>
<tr>
<td>2</td>
<td>00000000000000000000000000000010</td>
<td>96</td>
<td>11101111111111111111111111111111</td>
</tr>
<tr>
<td>3</td>
<td>00000000000000000000000000000011</td>
<td>97</td>
<td>11101111111111111111111111111111</td>
</tr>
<tr>
<td>4</td>
<td>00000000000000000000000000000100</td>
<td>98</td>
<td>11101111111111111111111111111111</td>
</tr>
<tr>
<td>5</td>
<td>00000000000000000000000000000101</td>
<td>99</td>
<td>11101111111111111111111111111111</td>
</tr>
<tr>
<td>6</td>
<td>00000000000000000000000000000110</td>
<td>100</td>
<td>11101111111111111111111111111111</td>
</tr>
<tr>
<td>7</td>
<td>00000000000000000000000000000111</td>
<td>101</td>
<td>11101111111111111111111111111111</td>
</tr>
<tr>
<td>8</td>
<td>000000000000000000000000000001000</td>
<td>102</td>
<td>11101111111111111111111111111111</td>
</tr>
<tr>
<td>9</td>
<td>000000000000000000000000000001001</td>
<td>103</td>
<td>11101111111111111111111111111111</td>
</tr>
<tr>
<td>10</td>
<td>000000000000000000000000000001010</td>
<td>104</td>
<td>11101111111111111111111111111111</td>
</tr>
<tr>
<td>11</td>
<td>000000000000000000000000000001011</td>
<td>105</td>
<td>11101111111111111111111111111111</td>
</tr>
<tr>
<td>12</td>
<td>000000000000000000000000000001100</td>
<td>106</td>
<td>11101111111111111111111111111111</td>
</tr>
<tr>
<td>13</td>
<td>000000000000000000000000000001101</td>
<td>107</td>
<td>11101111111111111111111111111111</td>
</tr>
<tr>
<td>14</td>
<td>000000000000000000000000000001110</td>
<td>108</td>
<td>11101111111111111111111111111111</td>
</tr>
<tr>
<td>15</td>
<td>000000000000000000000000000001111</td>
<td>109</td>
<td>11101111111111111111111111111111</td>
</tr>
<tr>
<td>16</td>
<td>000000000000000000000000000000001</td>
<td>110</td>
<td>11101111111111111111111111111111</td>
</tr>
<tr>
<td>17</td>
<td>000000000000000000000000000000010</td>
<td>111</td>
<td>11101111111111111111111111111111</td>
</tr>
<tr>
<td>18</td>
<td>000000000000000000000000000000011</td>
<td>112</td>
<td>11101111111111111111111111111111</td>
</tr>
<tr>
<td>19</td>
<td>000000000000000000000000000000001</td>
<td>113</td>
<td>11101111111111111111111111111111</td>
</tr>
<tr>
<td>20</td>
<td>0000000000000000000000000000000001</td>
<td>114</td>
<td>11101111111111111111111111111111</td>
</tr>
<tr>
<td>21</td>
<td>00000000000000000000000000000000001</td>
<td>115</td>
<td>11101111111111111111111111111111</td>
</tr>
<tr>
<td>22</td>
<td>000000000000000000000000000000000001</td>
<td>116</td>
<td>11101111111111111111111111111111</td>
</tr>
<tr>
<td>23</td>
<td>0000000000000000000000000000000000001</td>
<td>117</td>
<td>11101111111111111111111111111111</td>
</tr>
<tr>
<td>24</td>
<td>00000000000000000000000000000000000001</td>
<td>118</td>
<td>11101111111111111111111111111111</td>
</tr>
<tr>
<td>25</td>
<td>000000000000000000000000000000000000001</td>
<td>119</td>
<td>11101111111111111111111111111111</td>
</tr>
<tr>
<td>26</td>
<td>0000000000000000000000000000000000000001</td>
<td>120</td>
<td>11101111111111111111111111111111</td>
</tr>
<tr>
<td>27</td>
<td>00000000000000000000000000000000000000001</td>
<td>121</td>
<td>11101111111111111111111111111111</td>
</tr>
<tr>
<td>28</td>
<td>000000000000000000000000000000000000000001</td>
<td>122</td>
<td>11101111111111111111111111111111</td>
</tr>
<tr>
<td>29</td>
<td>0000000000000000000000000000000000000000001</td>
<td>123</td>
<td>11101111111111111111111111111111</td>
</tr>
<tr>
<td>30</td>
<td>00000000000000000000000000000000000000000001</td>
<td>124</td>
<td>11101111111111111111111111111111</td>
</tr>
<tr>
<td>31</td>
<td>000000000000000000000000000000000000000000001</td>
<td>125</td>
<td>11101111111111111111111111111111</td>
</tr>
<tr>
<td>32</td>
<td>0000000000000000000000000000000000000000000001</td>
<td>126</td>
<td>11101111111111111111111111111111</td>
</tr>
</tbody>
</table>

Table 5: Natural Number decomposition yielding 23 virtual bit-planes

30
7.4 The performance analysis: Comparison between Prime Decomposition and Natural Number Decomposition

In this section, we do a comparative study between the different decompositions and its effect upon higher-bit-plane data-hiding. We basically try to prove our following claims,

7.4.1 In k-bit Natural Number System, all the numbers in the range \([0, k(k + 1)/2]\) can be represented and only these numbers can be represented

Proof by Induction on \(k\):

Basis: \(k = 1\), we can represent only 2 numbers, namely 0 and 1, but we have, \(\frac{k(k+1)}{2} = 1\), i.e., all the numbers (and only these numbers) in the range \([0, 1]\), i.e., \([0, k(k+1)/2]\) can be represented for \(k = 1\).

Induction hypothesis: Let us assume the above result holds \(\forall k \leq n, n \in \mathbb{N}\).

Now, let us prove the same for \(k = n + 1\).

From induction hypothesis, we know, using \(n\) bit Natural Number System, all (and only) the numbers in the range \([0, \frac{n(n+1)}{2}]\) can be represented. Let us list all the valid representations in \(n\) bit,

\[
0 \equiv b_{0,n-1}b_{0,n-2} \ldots b_{0,1}b_{0,0} \equiv 0000 \ldots 00 \\
1 \equiv b_{1,n-1}b_{1,n-2} \ldots b_{1,1}b_{1,0} \equiv 0000 \ldots 01 \\
\ldots \\
\ldots \\
n(n + 1)/2 \equiv b_{n(n+1)/2,n-1}b_{n(n+1)/2,n-2} \ldots b_{n(n+1)/2,1}b_{n(n+1)/2,0} \equiv 1111 \ldots 11
\]

Now, for \((n + 1)\) bit Natural Number System, we have the weight corresponding to the \(n^{th}\) significant Bit (MSB), \(W(n) = n + 1\).

So when the MSB is 0, we have all the numbers in the range \([0, \frac{n(n+1)}{2}]\)

\[
0b_{0,n-1}b_{0,n-2} \\
0b_{1,n-1}b_{1,n-2} \\
\ldots \\
\ldots \\
0b_{n(n+1)/2,n-1}
\]

and when the MSB is 1, we get a new set of \(\frac{n(n+1)}{2} + 1\) numbers

\[
n + 1 + 0, \\
n + 1 + 1, \\
n + 1 + 2, \\
\ldots \\
\ldots \\
n + 1 + \frac{n(n + 1)}{2},
\]

31
i.e., all the (consecutive) numbers in the range \([n + 1, \frac{(n+1)(n+2)}{2}]\).

\[
\begin{align*}
&1b_{0,n-1}b_{0,n-2} \\
&1b_{1,n-1}b_{1,n-2} \\
&\quad \ldots \\
&1b_{n(n+1)/2,n-1}
\end{align*}
\]

So, we get all the numbers in the range \([0, \frac{n(n+1)}{2}] \cup [n + 1, \frac{(n+1)(n+2)}{2}] = [0, \frac{n(n+1)(n+2)}{2}]\)

Also, the maximum number that can be represented (all 1’s) using \((n + 1)\) bit Natural Number System. = \((n + 1) + (n) + (n - 1) + \ldots + (3) + (2) + (1) = \frac{(n+1)(n+2)}{2}\), and minimum number that can be represented (all 0’s) is 0. Hence, only the numbers in this range can be represented.

Hence, we proved for \(k = n + 1\) also. \(\Rightarrow \forall k \in \mathbb{N}\) the above result holds.

7.4.2 The proposed Natural Number Decomposition generates more (virtual) bit-planes

Using Classical binary decomposition, for a k-bit cover image, we get only k bit-planes per pixel, where we can embed our secret data bit. From equation (3), we get, \(p_n = \theta (n,\ln(n))\). Since \(n + 1 = o(n,\ln(n))\), the weight corresponding to the \(n^{th}\) bit in our number system using natural number decomposition eventually becomes much higher than the weight corresponding to the \(n^{th}\) bit in the number system using prime decomposition. In n-bit Prime Number System, the numbers in the range \([0, \sum_{i=0}^{n-1} p_i]\) can be represented, while in our n-bit Natural Number System, the numbers in the range \([0, \sum_{i=0}^{n-1} (i + 1)] = [0, \sum_{i=1}^{n} i] = [0, \frac{n(n+1)}{2}]\) can be represented. Now, it is easy to prove that \(\exists n_0 \in \mathbb{N} : \forall n \geq n_0\), we have, \(\sum_{i=0}^{n-1} p_i > \frac{n(n+1)}{2}\).

Hence, using same number of bits, it is eventually possible to represent more numbers in case the number system using prime decomposition, than that in case of the number system using natural number decomposition. This in turn implies that number of virtual bit-planes generated in case of natural number decomposition will be eventually more than the corresponding number of virtual bit-planes generated by prime decomposition.

From The bar-chart shown in Figure-10, we see that, in order to represent the pixel value 92, Natural number system requires at least 14 bits, while for Prime number system 10 bits suffice. So, at the time of decomposition the same pixel value will generate 14 virtual bit-planes in case of natural number decomposition and 10 for the prime, thereby increasing the space for embedding.

7.4.3 Natural Number Decomposition gives less distortion in higher bit-planes

Here we assume the secret message length (in bits) is same as image size, for evaluation of our test statistics. For message with different length, the same
Figure 10: Maximum number that can be represented in prime and natural number decomposition techniques
can similarly be derived in a straight-forward manner.

In case of Prime Decomposition technique, WMSE for embedding secret message bit only in $l^{th}$ (virtual) bitplane of each pixel (after expressing a pixel in our prime number system, using prime decomposition technique) = $p_l^2$, because change in $l^{th}$ bit plane of a pixel simply implies changing of the pixel value by at most the $l^{th}$ prime number. From above, (treating image-size as constant) we can immediately conclude, from equation (3), for $l > 0$

$$(WMSE_{l^{th} \ bitplane})_{Prime \ Decomposition} = w \times h \times p_l^2 = \theta(l^2 \log^2(l))$$

In case of our Natural Decomposition, WMSE for embedding secret message bit only in $l^{th}$ (virtual) bit-plane of each pixel (after expressing a pixel in our natural number system, using natural number decomposition technique) = $(l + 1)^2$. From above, (treating image-size as constant again) we can immediately conclude,

$$(WMSE_{l^{th} \ bitplane})_{Natural \ Number \ Decomposition} = (l + 1)^2 = \theta(l^2).$$

Since $(l + 1)^2 = o(l^2 \log^2(l))$, eventually we have,

$$(WMSE_{l^{th} \ bitplane})_{Natural \ Decomposition} < (WMSE_{l^{th} \ bitplane})_{Prime \ Decomposition}$$

The above result implies that the distortion in case of natural number decomposition is much less than that in case of prime decomposition. The plot shown in Figure-11 buttresses our claim, it compares the nature of the weight function in case of prime decomposition against that of the natural number decomposition.

So, from all above discussion, we conclude that Natural Number Decomposition gives less distortion than Prime Decomposition technique, while embedding secret message in higher bit-planes.

At a glance, results obtained for test-statistic WMSE, in case of our k-bit cover image,
\[
\begin{align*}
(WMSE_{th\ bitplane})_{Classical\ Binary\ Decomposition} &= \theta(4^l). \\
(WMSE_{th\ bitplane})_{Prime\ Decomposition} &= \theta(l^2 \cdot \log^2(l)). \\
(WMSE_{th\ bitplane})_{Natural\ Number\ Decomposition} &= (l + 1)^2 = \theta(l^2). \quad (18)
\end{align*}
\]

Also, results for our test-statistic \(PSNR_{worst}\),

\[
\begin{align*}
\left(PSNR_{worst}\right)_{Classical\ Binary\ Decomposition} &= 10 \log_{10} \left( \frac{(2^k - 1)^2}{(2^l)^2} \right). \\
\left(PSNR_{worst}\right)_{Prime\ Decomposition} &= 10 \log_{10} \left( \frac{(2^k - 1)^2}{c \cdot l^2 \cdot \log^2(l)} \right). \\
\left(PSNR_{worst}\right)_{Natural\ Number\ Decomposition} &= 10 \log_{10} \left( \frac{(2^k - 1)^2}{(l + 1)^2} \right). \quad (19)
\end{align*}
\]

From equations (18) and (19), we see that, WMSE gradually decreased from Binary to Prime and then from Prime to Natural decomposition techniques (minimized in case of Natural number decomposition), ensuring lesser probability of distortion, while PSNR gradually increased along the same direction (maximized in case of Natural number decomposition), implying more imperceptibility in message hiding.

### 7.4.4 Natural Number Decomposition is Optimal

This particular decomposition technique is optimal in the sense that it generates maximum number of (virtual) bit-planes and also least distortion while embedding in higher bit-planes, when the weight function is strictly monotonically increasing. Since, among all monotonic strictly increasing sequences of positive integers, natural number sequence is the tightest, all others are subsequences of the natural number sequence. Our generalized model indicates that the optimality of our technique depends on which number system we choose, or more precisely, which weight function we define. Since weight function \(W : \mathbb{Z}^+ \cup \{0\} \rightarrow \mathbb{Z}^+\) (Since we are going to represent pixel-values, that are nothing but non-negative integers, the co-domain of our weight function is set of non-negative integers. Also, weight function is assumed to be one-one, otherwise there will be too much redundancy) is optimized when it is defined as \(W(i) = i + 1, \forall i \in \mathbb{Z}^+ \cup \{0\}\), i.e., in case of natural number decomposition.

Since we have, the weight function \(W : \mathbb{Z}^+ \cup \{0\} \rightarrow \mathbb{Z}^+\), that assigns a bit-plane (index) an integral weight, if we assume that weight corresponding to a bit-plane is unique and the weight is monotonically increasing, one of the simplest but yet optimal way to construct such an weight function is to assign consecutive natural number values to the weights corresponding to each bit-plane, i.e., \(W(i) = i + 1, \forall i \in \mathbb{Z}^+ \cup \{0\}\) (We defined \(W(i) = i + 1\) instead of \(W(i) = i\), since we want all-zero representation for the value 0, in this particular number system). Now, this particular decomposition in virtual bit-planes and
embedding technique gives us optimal result. We get optimal performance of any data-hiding technique by minimizing our test-statistic WMSE. For embedding data in \( l \)th virtual bit-plane, we have \((WMSE)_l = (W(l))^2\), so minimizing WMSE implies minimizing the weight function \( W(.) \), but having our weight function allowed to assume integral values only, and also assuming the values assigned by \( W \) are unique (\( W \) is injective, we discard the un-interesting case when weight-values corresponding to more than one bit-planes are equal), we can without loss of generality assume \( W \) to be monotonically increasing.

But, according to the above condition imposed on \( W \), we see that such strictly increasing \( W \) assigning minimum integral weight-values to different bit planes must be linear in bit-plane index. Put it in another way, for \( n \)-bit number system, we need \( n \) different weights that are to be assigned to weight-values corresponding to \( n \) bit-planes. But, the assigning must also guarantee that these weight values are minimum possible. Such \( n \) different positive integral values must be smallest \( n \) consecutive natural numbers, i.e., \( 1, 2, 3, \ldots, n \). But, our weight function \( W(i) = i + 1, \forall i \in Z^+ \cup \{0\} \) merely gives these values as weights only, hence this technique is optimal.

Using classical binary decomposition, we get \( k \) bit planes only corresponding to a \( k \)-bit image pixel value, but in case of natural number decomposition, we get, \( n \)-bit pixels, where \( n \) satisfies,

\[
\begin{align*}
    n^2 + n - 2^{k+1} + 2 & \geq 0 \\
    \Rightarrow n \geq \frac{-1 + \sqrt{2^{k+3} + 9}}{2}, \quad n \in Z^+ \\
    \Rightarrow n = \theta(2^k)
\end{align*}
\]

8 Experimental Results for Natural Number decomposition technique

We have, again, as input:

- Cover Image: 8-bit (256 color) gray-level standard image of Lena.
- Secret message length = cover image size, (message string "sandipan" repeated multiple times to fill the cover image size).
- The secret message bits are embedded into one (selected) bit-plane per pixel only, the bit-plane is indicated by the variable \( p \).
- The test message is hidden ([26]) into the chosen bit-plane using different decomposition techniques, namely, the classical (traditional) binary (LSB) decomposition, Fibonacci 1-sequence decomposition and Prime decomposition separately and compared.
We get, as output:

- As was obvious from the above theoretical discussions, our experiment supported the fact that was proved mathematically, i.e., we got more (virtual) bit-planes and less distortion after embedding secret message into the bit-planes in case of Natural and Prime decomposition technique than in case of Fibonacci technique and classical binary LSB data hiding technique. We could also capture the hidden message from the stego-image successfully using our decoding technique.

- As obvious, as the relative entropy between the cover-image and the stego-image tends to be more and more positive (i.e., increases), we get more and more visible distortions in image rather than invisible watermark.

- As recapitulation of our earlier experimental result, Figure-8 shows gray level (0…255) vs. frequency plot of the cover image and stego-image in case of data-hiding technique based on Prime decomposition. This figure shows that, we get 15 bit-planes and the change of frequency distribution (and hence probability mass function) corresponding to graylevel values is least when compared to the other two techniques, eventually resulting in a still less relative entropy between the cover-image and stego-image, implying least visible distortions, as we move towards higher bit-planes for embedding data bits.

- Figure-12 shows gray level (0…255) vs. frequency plot of the cover image and stego image in case of data-hiding technique based on Natural Number decomposition. We get 23 bit-planes and the change of frequency distribution (and hence probability mass function) corresponding to gray-level values is least when compared to the other two techniques, eventually resulting in a still less relative entropy between the cover-image and stego-image, implying least visible distortions, as we move towards higher bit-planes for embedding data bits.

- Data-hiding technique using the natural number decomposition has a better performance than that of prime decomposition, the later being more efficient than classical binary decomposition, when judged in terms of embedding secret data bit into higher bit-planes causing least distortion and thereby having least chance of being detected, since one of principle ends of data-hiding is to go as long as possible without being detected.

- Using classical binary decomposition, we get here only 8 bit planes (since an 8 bit image), using Fibonacci 1-sequence decomposition we have 12 (virtual) bit-planes, and using prime decomposition we have 15 (virtual) bit-planes, but using natural decomposition, we have the highest, namely, 23 (virtual) bit planes.

- As vindicated in the figures 8 and 9, distortion is much less for natural decomposition, than that in case of prime. This technique can also be
Figure 12: Result of embedding secret data in different bit-planes using Natural Number decomposition technique
Figures 13 and 14 show comparison of WMSE and PSNR values, respectively, obtained from experimental results. It clearly shows that even for higher bitplanes the secret data can be reliably hidden with quite high PSNR value. Hence, it will be difficult for the attacker to predict the secret embedding bitplane.

The experimental results were obtained by implementing the algorithms and data hiding techniques in C++ (open source gcc) and (gray-scale) Lena bitmap as input image file. Also the extraction algorithms that described for both the techniques run at linear time in length of message embeded.

9 Conclusions

This chapter presented very simple methods of data hiding technique using prime numbers / natural numbers. It is shown (both theoretically and experimentally) that the data-hiding technique using prime decomposition outperforms the famous LSB data hiding technique using classical binary decomposition and that using Fibonacci p-sequence decomposition. Also, the technique

Figure 13: Comparison of WMSE values for different data hiding techniques enhanced by embedding into more than one (virtual) bit-plane, following the variable-depth data-hiding technique [9].
using natural number decomposition outperforms the one using prime decomposition, when thought with respect to embedding secret data bits at higher bit-planes (since number of virtual bit-planes generated also increases) with less detectable distortion. We have shown all our experimental results using the famous Lena image, but since in all our theoretical derivation above we have shown our test-statistic value (WMSE, PSNR) independent of the probability mass function of the gray levels of the input image, the (worst-case) result will be similar if we use any gray-level image as input, instead of the Lena image.

References


10 Short CV of the Authors

Sandipan Dey  Sandipan Dey is a Technical Lead in Cognizant Technology Solutions, India. He has four years of experience in the software industry. He has C/C++ application development experience as well as research and development experience. He received his B.E. from Jadavpur University, India. His research interests include Computer Security, Steganography, Cryptography, Algorithm. He is also interested in Compilers, program analysis and evolutionary algorithms. He has published several papers in international journals and conferences.
Ajith Abraham  Dr. Ajith Abraham’s research and development experience includes over 17 years in the Industry and Academia spanning different continents in Australia, America, Asia and Europe. He works in a multidisciplinary environment involving computational intelligence, network security, sensor networks, e-commerce, Web intelligence, Web services, computational grids, data mining and applied to various real world problems. He has authored/co-authored over 350 refereed journal/conference papers and book chapters and some of the works have also won best paper awards at international conferences and also received several citations. Some of the articles are available in the ScienceDirect Top 25 hottest articles. His research interests in advanced computational intelligence include Nature Inspired Hybrid Intelligent Systems involving connectionist network learning, fuzzy inference systems, rough set, swarm intelligence, evolutionary computation, bacterial foraging, distributed artificial intelligence, multi-agent systems and other heuristics. He has given more than 20 plenary lectures and conference tutorials in these areas. Currently, he is working with the Norwegian University of Science and Technology (NTNU), Norway. Before joining NTNU, he was working under the Institute for Information Technology Advancement (IITA) Professorship Program funded by the South Korean Government. He was a Researcher at Rovira i Virgili University, Spain during 2005-2006. He also holds an Adjunct Professor appointment in Jinan University, China and Dalian Maritime University, China. He has held academic appointments in Monash University, Australia; Oklahoma State University, USA; Chung-Ang University, Seoul and Yonsei University, Seoul. Before turning into a full time academic, he was working with three international companies: Keppel Engineering, Singapore, Hyundai Engineering, South Korea and Ashok Leyland Ltd., India where he was involved in different industrial research and development projects for nearly 8 years. He received Ph.D. degree in Computer Science from Monash University, Australia and a Master of Science degree from Nanyang Technological University, Singapore. He serves the editorial board of over 30 reputed International journals and has also guest edited 28 special issues on various topics. He is actively involved in the Hybrid Intelligent Systems (HIS) ; Intelligent Systems Design and Applications (ISDA) and Information Assurance and Security (IAS) series of International conferences. He is a Senior Member of IEEE (USA), IEEE Computer Society (USA), IET (UK), IEAust (Australia) etc. In 2008, he is the General Chair/Co-chair of Tenth International Conference on Computer Modeling and Simulation, (UKSIM’08), Cambridge, UK; Second Asia International Conference on Modeling and Simulation (AMS’08), Kuala Lumpur, Malaysia; Eight International Conference on Intelligent Systems Design and Applications (ISDA’08), Kaohsuing, Taiwan; Fourth International Symposium on Information Assurance and Security (IAS’08), Naples, Italy; 2nd European Symposium on Computer Modeling and Simulation, (EMS’08), Liverpool, UK; Eighth International Conference on Hybrid Intelligent Systems (HIS’08), Barcelona, Spain; Fifth International Conference on Soft Computing as Transdisciplinary Science and Technology (CSTST’08), Paris, France Program Chair/Co-chair of Third International Conference on Digital Information Management (ICDIM’08), Lou-
Bijoy Bandopadhyay  Dr. Bijoy Bandyopadhyay is currently a faculty member of the Department of Radio Physics and Electronics, University of Calcutta. He received his PhD, M. Tech. B.Tech. and B.Sc. degrees from University of Calcutta, Kolkata. His current research interest includes Microwave Tomography, Ionosphere Tomography, Atmospheric Electricity Parameters, and Computer Security (Steganography). He has published numerous papers in International Journals and Conferences.

Sugata Sanyal  Dr. Sugata Sanyal is a Professor in the School of Technology and Computer Science at the Tata Institute of Fundamental Research, India. He received his Ph.D. degree from Mumbai University, India, M. Tech. from IIT, Kharagpur, India and B.E. from Jadavpur University, India. His current research interests include Multi-Factor Security Issues, Security in Wireless and Mobile Ad Hoc Networks, Distributed Processing, and Scheduling techniques. He has published numerous papers in national and international journals and attended many conferences. He is in the editorial board of four International Journals. He is co-recipient of Vividhlaxi Audyogik Samsodhan Vikas Kendra Award (VASVIK) for Electrical and Electronics Science and Technologies (combined) for the year 1985. He was a Visiting Professor in the Department of Electrical and Computer Engineering and Computer Science in the University of Cincinnati, Ohio, USA in 2003. He delivered a series of lectures and also interacted with the Research Scholars in the area of Network Security in USA, in University of Cincinnati, University of Iowa, Iowa State University and Oklahoma State University. He has been an Honorary Member of Technical Board in UTI (Unit Trust of India), SIDBI (Small Industries Development Bank of India) and Coal Mines Provident Funds Organization (CMPFO). He has also been acting as a consultant to a number of leading industrial houses in India. More information about his activities is available at http://www.tifr.res.in/ sanyal. Sugata Sanyal is in an honorary member of the Technical Board and has also served as a consultant: (Few significant ones): UTI (Unit Trust of India); SIDBI (Small Industries Development Bank of India); CMPFO (Coal Mines Provident Funds Organization); MAHAGENCO; Centre for Development of Telematics (CDOT); Crompton Greaves Limited; Tata Electronic Co. (Research and Development).