KaB (Key as Bitmap) Encoding Algorithm

Introduction

This document describes an encoding algorithm that hides information within a sequence of bytes. The method relies on a key that dictates specific bit positions where secret data is embedded. By carefully modifying only selected bits, the original carrier data remains as intact as the key allows it to be while containing hidden information.

Encoding Process

This algorithm works for a sequence of data with any length, as long as the length of each key element is equal to or less than the length of the data block. The encoding process involves three primary components:

1. **Secret Data** - The information to be embedded.
2. **Key** - A sequence that determines which bits in the carrier will be modified.
3. **Carrier** - The byte sequence that will store the encoded information.

Steps of Encoding:

1. **Bit Selection Based on Key**: Each carrier byte is processed alongside a corresponding key byte. The key acts as a bitmap, indicating which bit positions are allowed to hold secret data.

2. **Bit Substitution**: Where the key has a `1`, a corresponding bit from the secret data replaces the original bit in the carrier.
3. **End of Data Indication**: To indicate the end of data, 0 will be encoded where the data would've been, therefore a long sequence of zeros should be visible to a decoder.

Example 1

Consider the following parameters:

Secret Data:

```
10010010
11100111
```

Key:

```
00001001
00010110
00110100
```

Carrier Before Encoding:

```
10010010
01010011
11010001
01010010
10011011
10101101
01010100
```

The following is a step by step log of how this encoding is done by an encoder (bit pos is counter from left to right, 0 to 7).

```
encoded 1 to carrier byte 0 and bit pos of 4, used 0th key byte
encoded 0 to carrier byte 0 and bit pos of 7, used 0th key byte
encoded 0 to carrier byte 1 and bit pos of 3, used 1th key byte
encoded 1 to carrier byte 1 and bit pos of 5, used 1th key byte
encoded 0 to carrier byte 1 and bit pos of 6, used 1th key byte
encoded 0 to carrier byte 2 and bit pos of 2, used 2th key byte
encoded 1 to carrier byte 2 and bit pos of 3, used 2th key byte
encoded 0 to carrier byte 2 and bit pos of 5, used 2th key byte
encoded 1 to carrier byte 3 and bit pos of 4, used 0th key byte
encoded 1 to carrier byte 3 and bit pos of 7, used 0th key byte
```

```
encoded 1 to carrier byte 4 and bit pos of 3, used 1th key byte
encoded 0 to carrier byte 4 and bit pos of 5, used 1th key byte
encoded 0 to carrier byte 4 and bit pos of 6, used 1th key byte
encoded 1 to carrier byte 5 and bit pos of 2, used 2th key byte
encoded 1 to carrier byte 5 and bit pos of 3, used 2th key byte
encoded 1 to carrier byte 5 and bit pos of 5, used 2th key byte
encoded 0 to carrier byte 6 and bit pos of 4, used 0th key byte
encoded 0 to carrier byte 6 and bit pos of 7, used 0th key byte
```

And the carrier will be:

```
10011010
01000101
11010001
01011011
10011001
10111101
01010100
```

Example 2

Consider the following parameters:

Secret:

```
10010010
```

Key:

```
00001001
```

Carrier before encoding:

```
10010010
10011000
11001110
00001100
```

The following is a step by step log of how this encoding is done by an encoder (bit pos is counter from left to right, 0 to 7).

```
encoded 1 to carrier byte 0 and bit pos of 4, used 0th key byte
encoded 0 to carrier byte 0 and bit pos of 7, used 0th key byte
encoded 0 to carrier byte 1 and bit pos of 4, used 0th key byte
encoded 1 to carrier byte 1 and bit pos of 7, used 0th key byte
encoded 0 to carrier byte 2 and bit pos of 4, used 0th key byte
encoded 0 to carrier byte 2 and bit pos of 7, used 0th key byte
encoded 1 to carrier byte 3 and bit pos of 4, used 0th key byte
encoded 0 to carrier byte 3 and bit pos of 7, used 0th key byte
```

And the carrier will be:

```
10011010
10010001
11000110
00001100
```

Note that the carrier in this case needed to be at least 4 bytes, because the key has 2 set bits and the secret is 8 bits long.