Steganography

P2-project
Group A325a
Software
Aalborg University
May 24, 2016

AALBORG UNIVERSITY
STUDENT REPORT

Første Studieår v/ Det Teknisk-Naturvidenskabelige Fakultet
Byggeri og Anlæg
Strandvejen 12-14
9000 Aalborg
http://www.tnb.aau.dk

Title:

Steganography

Project:

P2-project

Project period:

February 2016 - May 2016

Project group:

A325a

Authors:

Anders L. Jakobsen
Andreas N. Jensen
Matias R. Jensen
Rasmus Jespersen
Simon N. Linnebjerg
Theis E. Jendal
Thomas B. Andersen

Supervisor:

Søren Enevoldsen

Pagecount: TODO
Appendix: TODO
Finished 24-05-2016

Abstract:

Synopsis

Preface

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam neque augue, tincidunt id augue at, mollis bibendum felis. Vestibulum ultrices nisi at tortor venenatis, nec ultrices justo pulvinar. Nam non iaculis metus, et consequat lectus. Sed vestibulum dui dolor, quis auctor orci posuere non. Aenean commodo pulvinar augue, eu aliquam sapien iaculis at. Sed eget vestibulum risus, vel viverra justo. Pellentesque ut lacinia dui, vel vestibulum nisi. Sed imperdiet consectetur turpis, id lobortis mi viverra sit amet. Ut quis aliquet nisi. Quisque dolor quam, efficitur quis aliquet eget, commodo consequat tellus. Morbi tincidunt ipsum sit amet velit pretium blandit varius non diam. Nam vitae purus tempus, auctor justo in, malesuada tortor. Donec nulla mauris, faucibus id diam et, finibus maximus orci. Aenean euismod maximus odio, a molestie urna cursus id. Maecenas congue ut sapien blandit tristique.

Duis est purus, lobortis eu ex scelerisque, viverra rhoncus risus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec dictum augue et arcu volutpat ullamcorper. Nulla suscipit eget sem sit amet porta. Praesent consectetur tellus in eros sagittis, id rutrum augue molestie. Nam lobortis massa eu dapibus malesuada. Etiam id velit commodo neque maximus sollicitudin. Maecenas finibus cursus orci, eu fermentum lectus pharetra at. Pellentesque ornare, orci sed ultrices maximus, risus nisi posuere orci, in finibus urna nisl sit amet mi. Ut id accumsan quam. Vivamus porta sem sit amet aliquam hendrerit. Vivamus sagittis magna porttitor hendrerit volutpat. Integer dolor massa, ullamcorper vel arcu euismod, viverra dapibus lectus. Sed sodales porta magna, non pretium risus aliquet vitae.

Contents

1 Project description

Part I

Theory

Introduction 2

Analysis of technologies 3

3.1 Least Significant Bit method

A commonly used method in digital steganography is the Least Significant Bit (LSB) method. The basic principle of this method is to make small changes to a large amount of data. LSB is useful in data formats where small change don't cause big differences. Embedding hidden messages in text would for example be a bad idea, as changes would be apparent, but on an image such changes may not be apparent. In a lossless image format like PNG (see **??**), the decompressed data is essentially a 2D matrix of pixels. Depending on the pixel format (sRGB, aRGB or perhaps even grayscale) each pixel has a set amount of color channels, where each value is a byte (8 bits), which determines the intensity of the color channel on that specific pixel. The purpose of the LSB method is to change the least significant bit in each of these bytes, and perhaps even the second least significant bit as well. This means that the decimal value may only change by 1 if modifying one bit and by 3 if modifying two least significant bits.

For example, say we want to embed the number '**42**' in a message. This can be represented as an 8-bit binary number:

$$00101010 \tag{3.1}$$

Since this message is 8 bits long, we would need 8 bytes when modifying one bit and 4 bytes when modifying two bits. In the former case, this means that we would need two aRGB pixels, as this gives us 2 * 4 bytes to hide our message in. For example, let's say we want to hide our image in the two pixels:

$$\begin{array}{cccc} 11111111 & 01001111 & 10110000 & 10000111 \\ 11111111 & 10010110 & 10001110 & 10000001 \end{array} \tag{3.2}$$

With the LSB method, we now need to look at the least significant bit in each of these bytes, and negate the bit if it does not correspond to the bit we are currently trying to insert. When the message is embedded in the original pixels, the outcome is:

$$\begin{array}{cccc} 1111111\mathbf{0} & 0100111\mathbf{0} & 1011000\mathbf{1} & 1000011\mathbf{0} \\ 1111111\mathbf{1} & 1001011\mathbf{0} & 1000111\mathbf{1} & 1000000\mathbf{0} \end{array} \tag{3.3}$$

The embedded message can then be extracted by simply combining each least significant bit of the 8 bytes. After doing this, we end up with the message seen in Equation 3.1.

When comparing the pixel values of a modified image (now referred to as a stego-image) and the original image, the change is clear on the bit/byte level, but not be apparent to a

human comparing the two images. This means that LSB is prone to steganalysis (see **??**). With the stego-image and original image in possession, it is trivial for a piece of software to detect that one of the images have been edited. In addition, the LSB method will typically increase the size of the image, since adding noise to the least significant bits increases the amount of colors and can affect lossless compression used in formats such as PNG.

Bibliography