Behind the Music:
MP3 Steganography

Mikhail Zaturenskiy

Why MP3 Steganography?

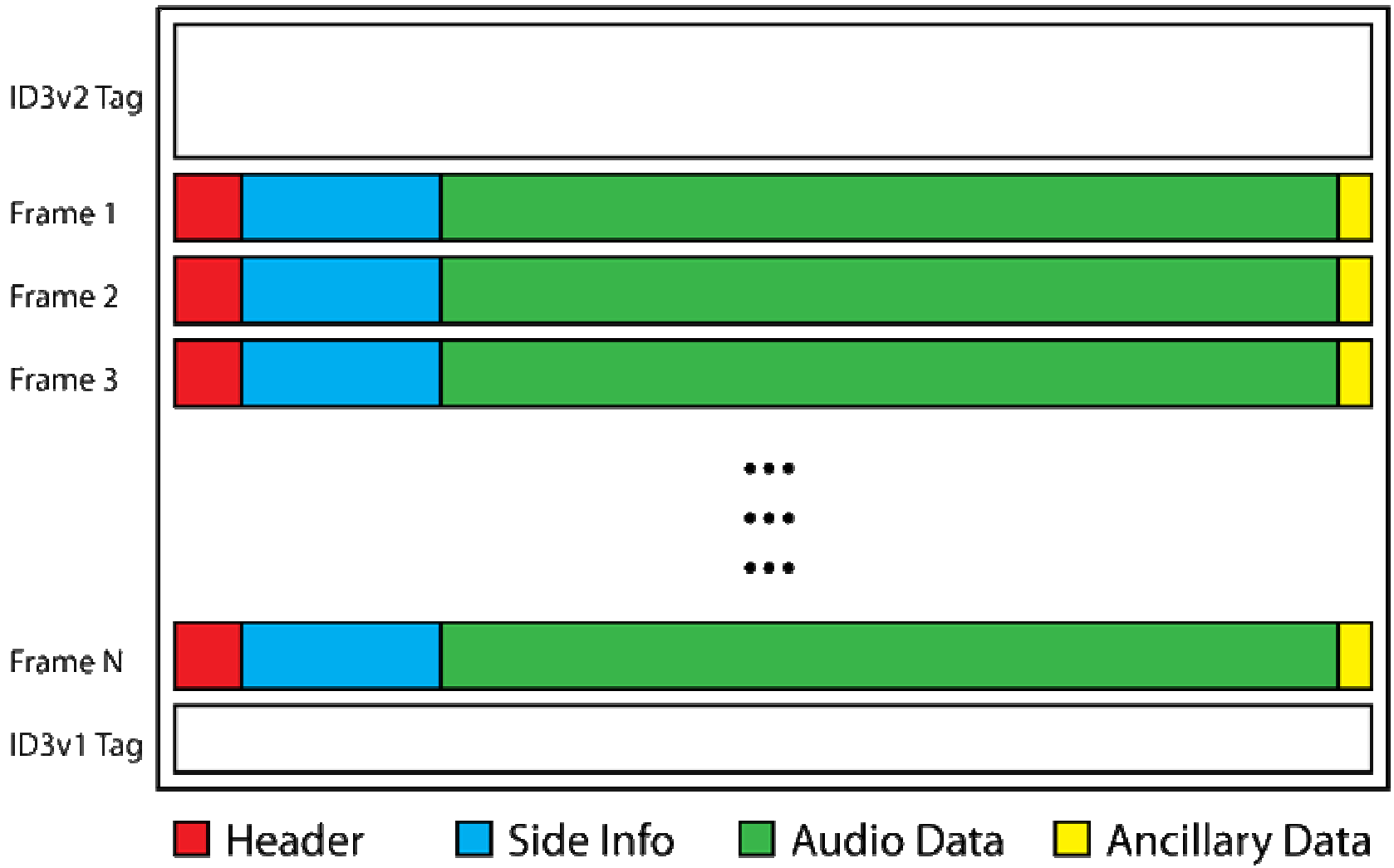
- Topic suggested to Prof. William Lidinsky by the FBI

- Offered as a semester project in ITM448-"System and Network Security"
- I accepted the project for Fall 2008
- Continued as Independent Study for Spring 2009

Objective

- Find a way to hide data inside an MP3 audio file
  - Fit as much data as possible
  - Make it not detectable
    - If nobody knows it's there, they probably won't look for it

MP3 Anatomy


ID3v2 Tag
Frame 1
Frame 2
Frame 3
Frame N
ID3v1 Tag
Header
Side Info
Audio Data
Ancillary Data

Where to hide covert data?

1. Encode it in inaudible wave form before compression
   - MP3 is lossy – can be a problem
2. Insert it during compression process
   - Complex
3. Stuff it into empty spaces after compression
   - Unused header bits
   - Unused side information bits
   - Frames with no audio data
   - Overwriting Ancillary data
   - Padding bytes

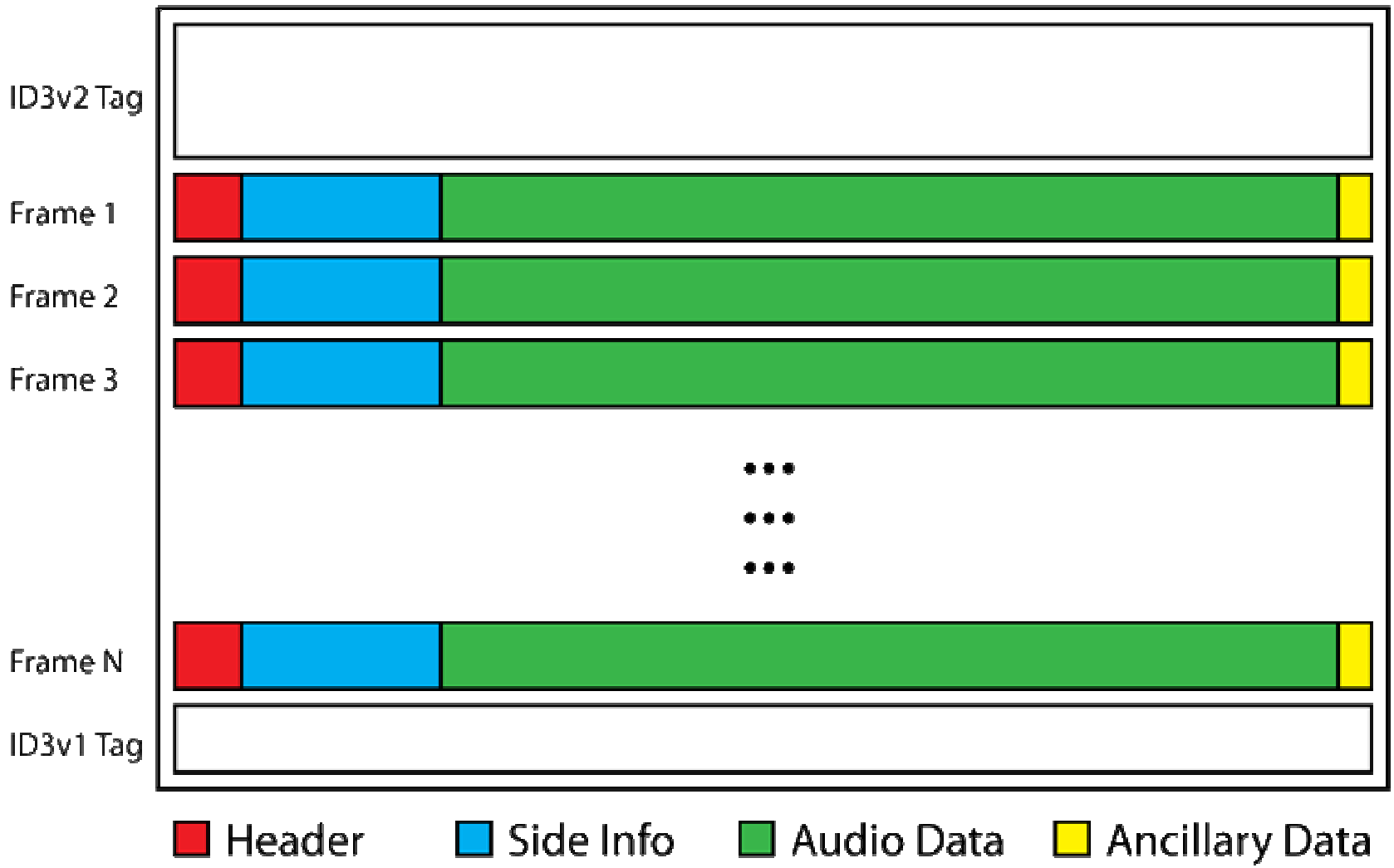
Some Existing Programs

- **Mp3stego**
  - Fabien A.P. Petitcolas
    - Cambridge (1998)
  - Inserts data during compression process
- **Mp3stego** (java)
  - Lukasz Maciak, Micheal Ponniah, Renu Sharma
    - Montclair State University (2005)
  - Padding byte stuffing attempt
- **Mp3stegz**
  - Achmad Zaenuri
    - Diponegoro University (2008)
  - Uses empty frames

Where to hide covert data?

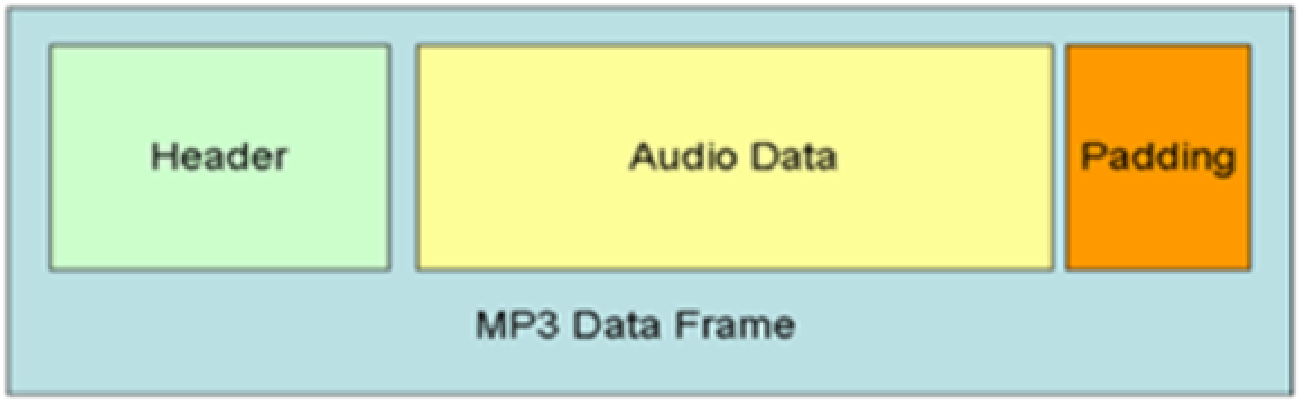
1. Encode it in inaudible wave form before compression
   - MP3 is lossy – can be a problem
2. Insert it during compression process
   - Complex
3. Stuff it into empty spaces after compression
   - Unused header bits
   - Unused side information bits
   - Frames with no audio data
   - Overwriting Ancillary data
   - Padding bytes

MP3 Anatomy


ID3v2 Tag
Frame 1
Frame 2
Frame 3
Frame N
ID3v1 Tag
Header
Side Info
Audio Data
Ancillary Data

Post Encoding Steganography

- Unused Header Bit Stuffing
- Padding Byte Stuffing



(Slide by Maciak, Ponniah and Sharma)

What's 1 header bit worth?

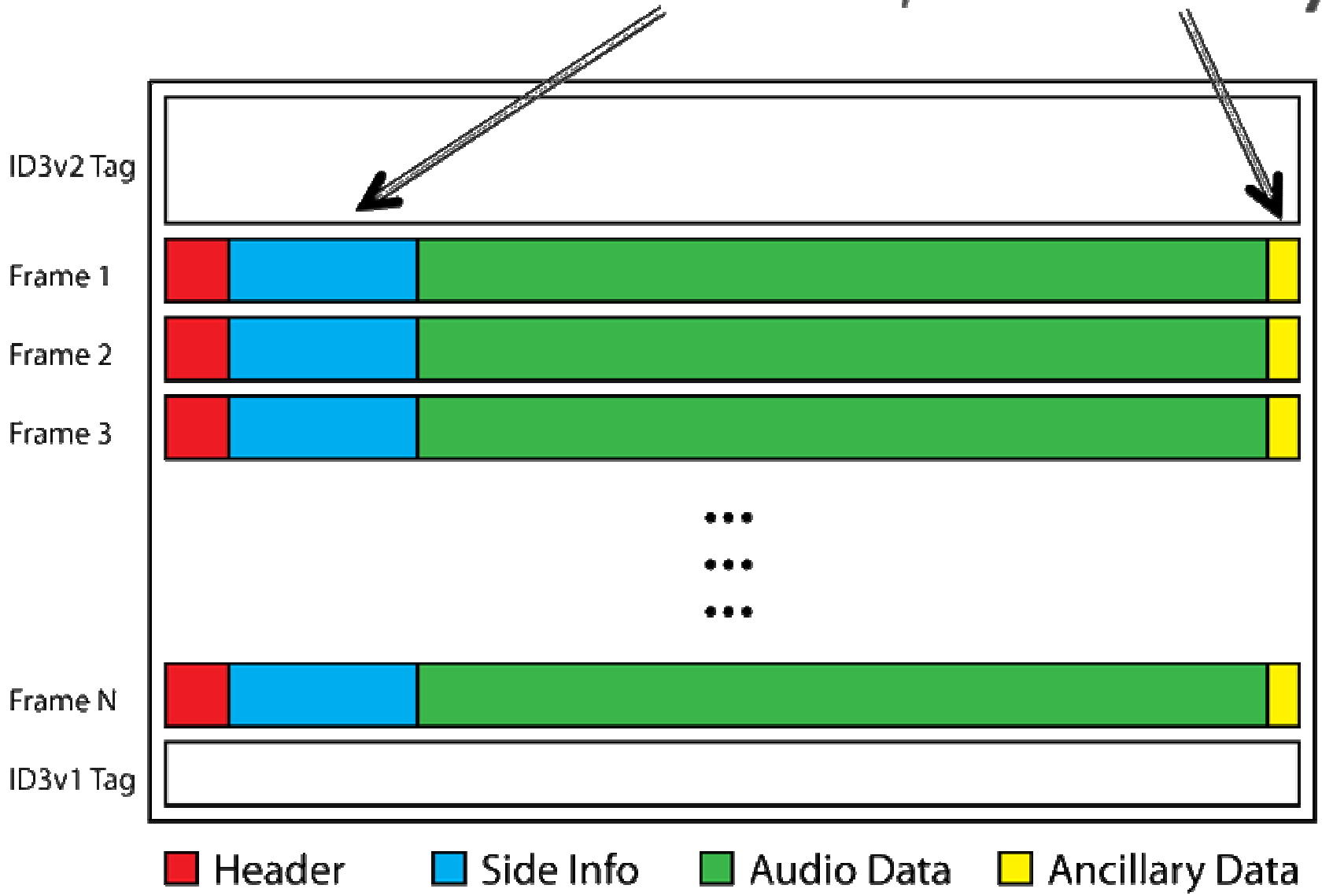
1 header bit ≈
280 bytes of data
per minute of audio

(MP3's use ~38.46 frames per second)

# of Header Bits Used	“Injectable” Bytes of data per minute of audio
1	280
2	570
3	860
4	1150
5	1440

(approximations)

What about Side Info/Ancillary?



My Ultimate Goals

- Secure!
- Platform-independent (Java)
- Mix of techniques
  - Unused Header Bit Stuffing
  - Unused Side Info Bit Stuffing
  - Empty Frame Stuffing
  - Ancillary Data Stuffing
- Let user decide which techniques to use
- Flexible
  - Will work with ANY MP3 audio file
- Fast
- User-friendly

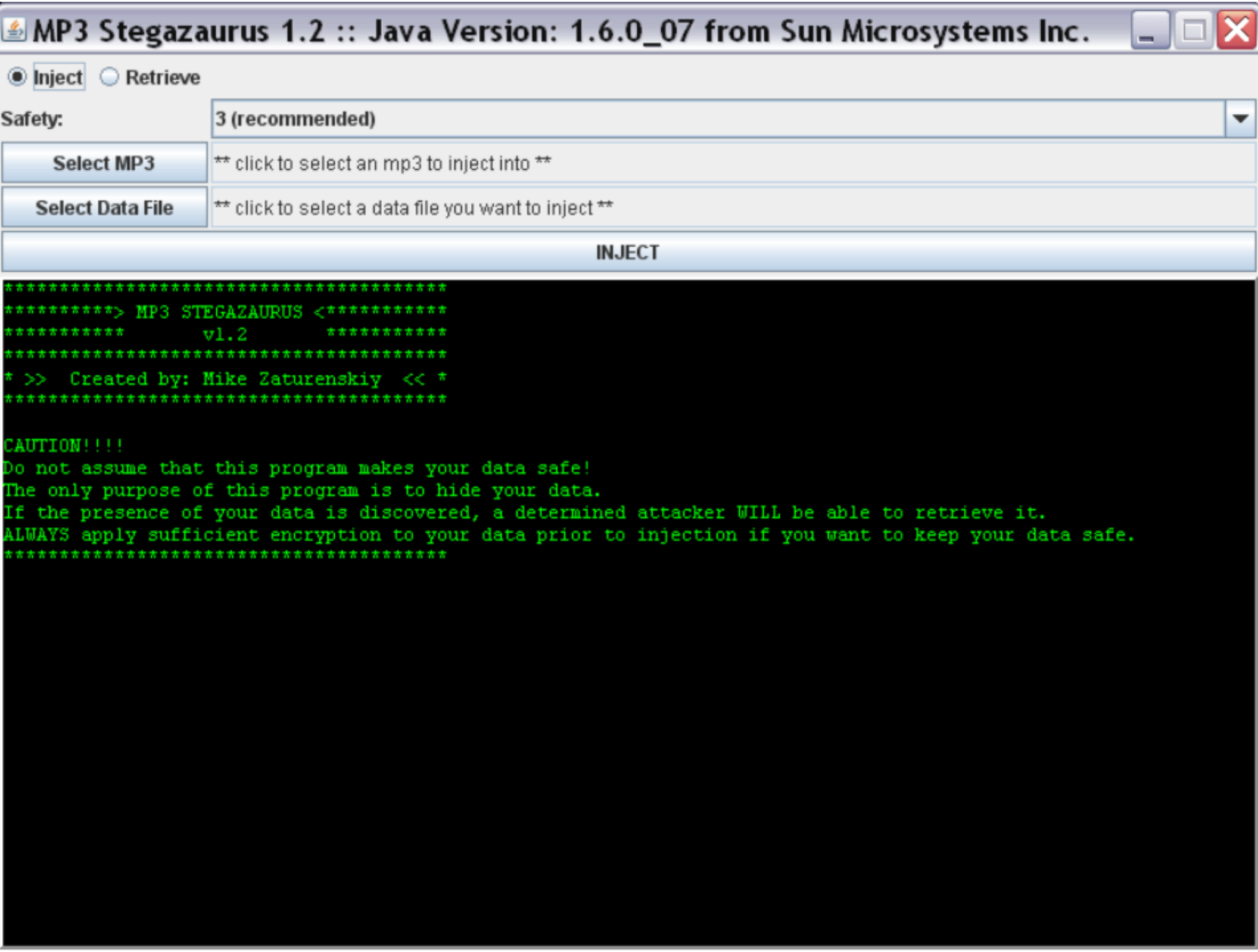
Fall 08 – MP3 STEGAZAURUS 1.1

- Working Java program
- Command line interface
- No effect on file size or audio quality
- Unused Header Bit Stuffing
  - User choice to use between 1 and 5 bits
- Works with:
    - CBR files, some VBR files
    - ID3 tagged files
    - Different bit-rate files
- Inject any file, not just Text
  - Works with binary data, tested with a small WAV file
- Fast!
  - Injects/Retrieves in seconds

CBR: Constant Bit Rate
VBR: Variable Bit Rate

Spring 09 – Progress so far

- Working Java program
- **GUI**
- No effect on file size or audio quality
- Unused Header Bit Stuffing
  - User choice to use between 1 and 5 bits
- **Side Info Bit Stuffing**
  - **3 extra bits per frame (about 860 bytes per minute)**
- Works with:
    - **VBR and CBR files (better compatibility)**
    - ID3 tagged files
    - Different bit-rate files
- Inject any file, not just Text
  - Works with binary data, tested with a small WAV file and JPG images
- Fast!
  - Injects/Retrieves in seconds


MP3 Stegazaurus 1.2 :: Java Version: 1.6.0_07 from Sun Microsystems Inc.
Inject
Retrieve
Safety:
3 (recommended)
Select MP3
** click to select an mp3 to inject into **
Select Data File
** click to select a data file you want to inject **
INJECT
**********> MP3 STEGAZAURUS <**********
********** v1.2 **********
* >> Created by: Mike Zaturenskiy << *
CAUTION!!!!
Do not assume that this program makes your data safe!
The only purpose of this program is to hide your data.
If the presence of your data is discovered, a determined attacker WILL be able to retrieve it.
ALWAYS apply sufficient encryption to your data prior to injection if you want to keep your data safe.

Note

1. Filename of your hidden file is NOT stored!
2. When you retrieve your file, you MUST know the "safety" number used to embed

- What does this mean?
  - An attacker not knowing the "safety" number nor the file type will have a more difficult time determining what is stored in the MP3 file

- YOU SHOULD STILL USE ENCRYPTION

Findings

- Using **3 bits** per frame header is optimal
  - Use of bit **5** causing issues on portable MP3 Players
  - Use of bit **4** causing issue with one PC MP3 Player
- Skip 2 first frames of file to avoid detection by Winamp and Win. Media Player
- Padding Byte Stuffing **DOES NOT WORK**

Why Padding?

$$FrameSize = 144 * \frac{BitRate}{SampleRate + Padding}$$

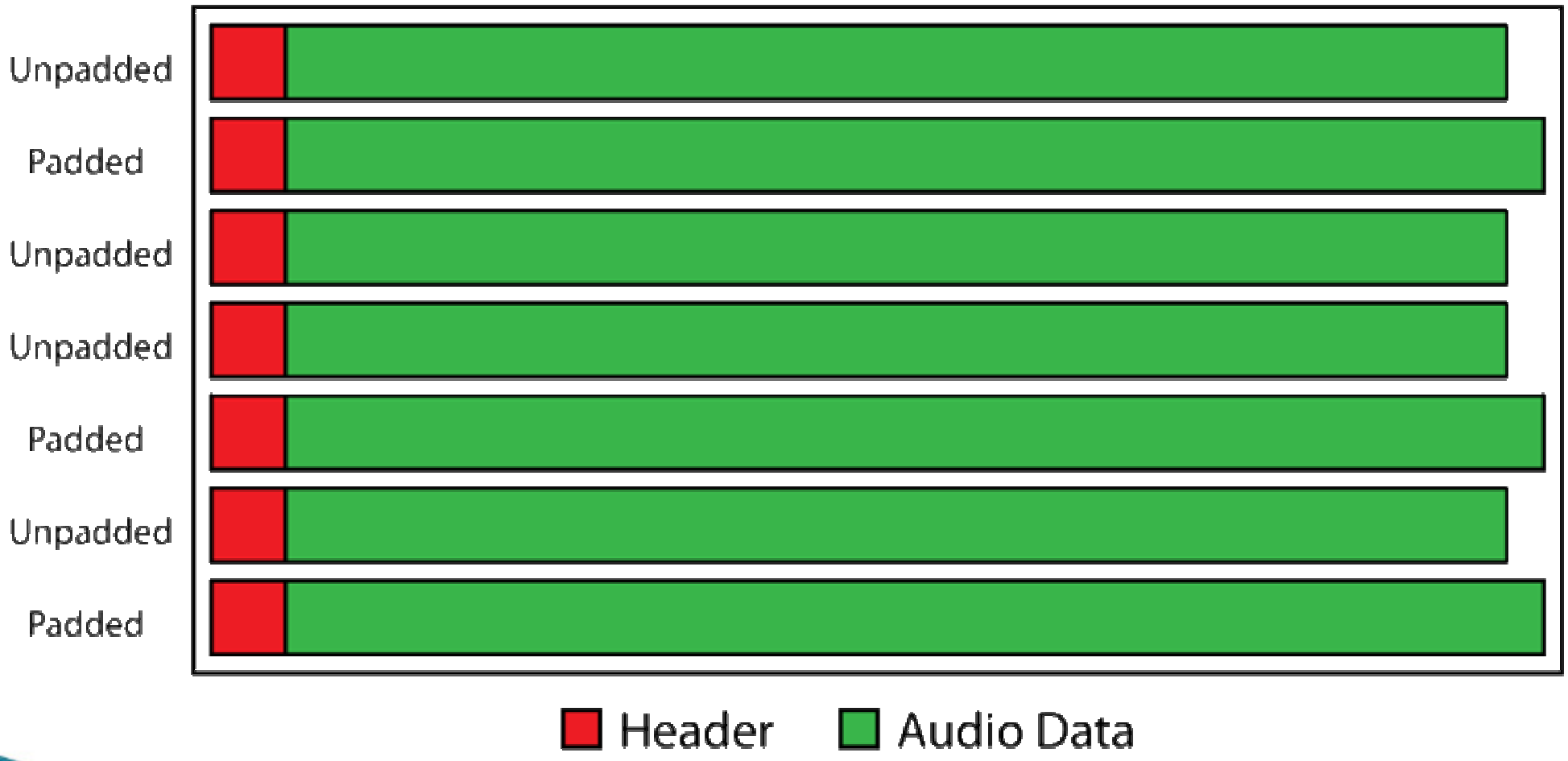
(in bytes)

Why Padding?

Bit Rate = 160kbps
SampleRate = 44.1khz

$$FrameSize = 144 * \frac{160000}{441000 + 0} = \boxed{522.448 \text{bytes}}$$

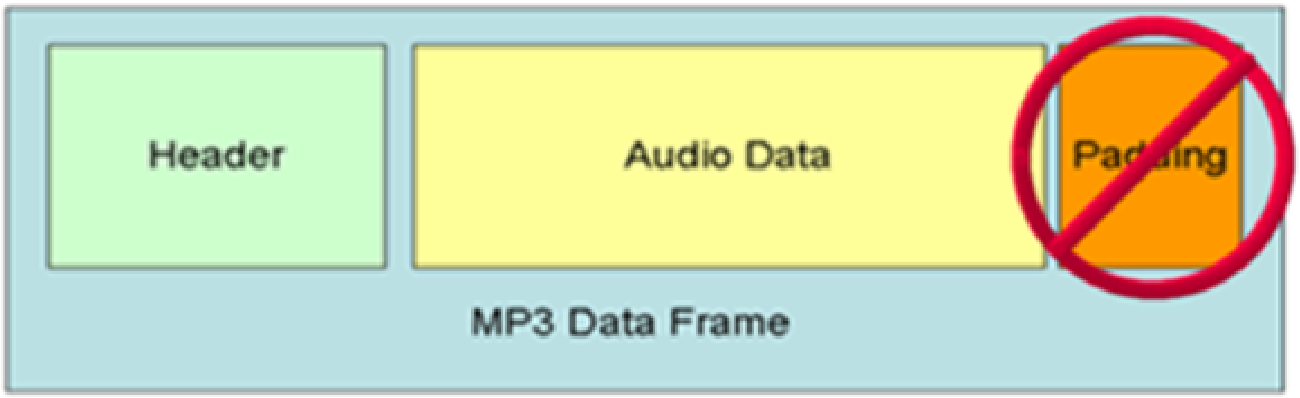
How Padding Works



Post Encoding Steganography
Issues

- Unused Header Bit Stuffing

- Padding Byte Stuffing



What's next?

- Add **Empty Audio Frame Stuffing**
  - Potentially store more data
- Research **Ancillary bit Stuffing**
- Research more Side Info bit Stuffing
- Research more Header bit Stuffing
- Spread data over **multiple MP3 files**
  - Store larger files
  - More secure, need all pieces to get original data
- Add option to **"clean" audio file** to get rid of hidden data
- No plan to implement compression or encryption
  - This can be done by existing tools more efficiently