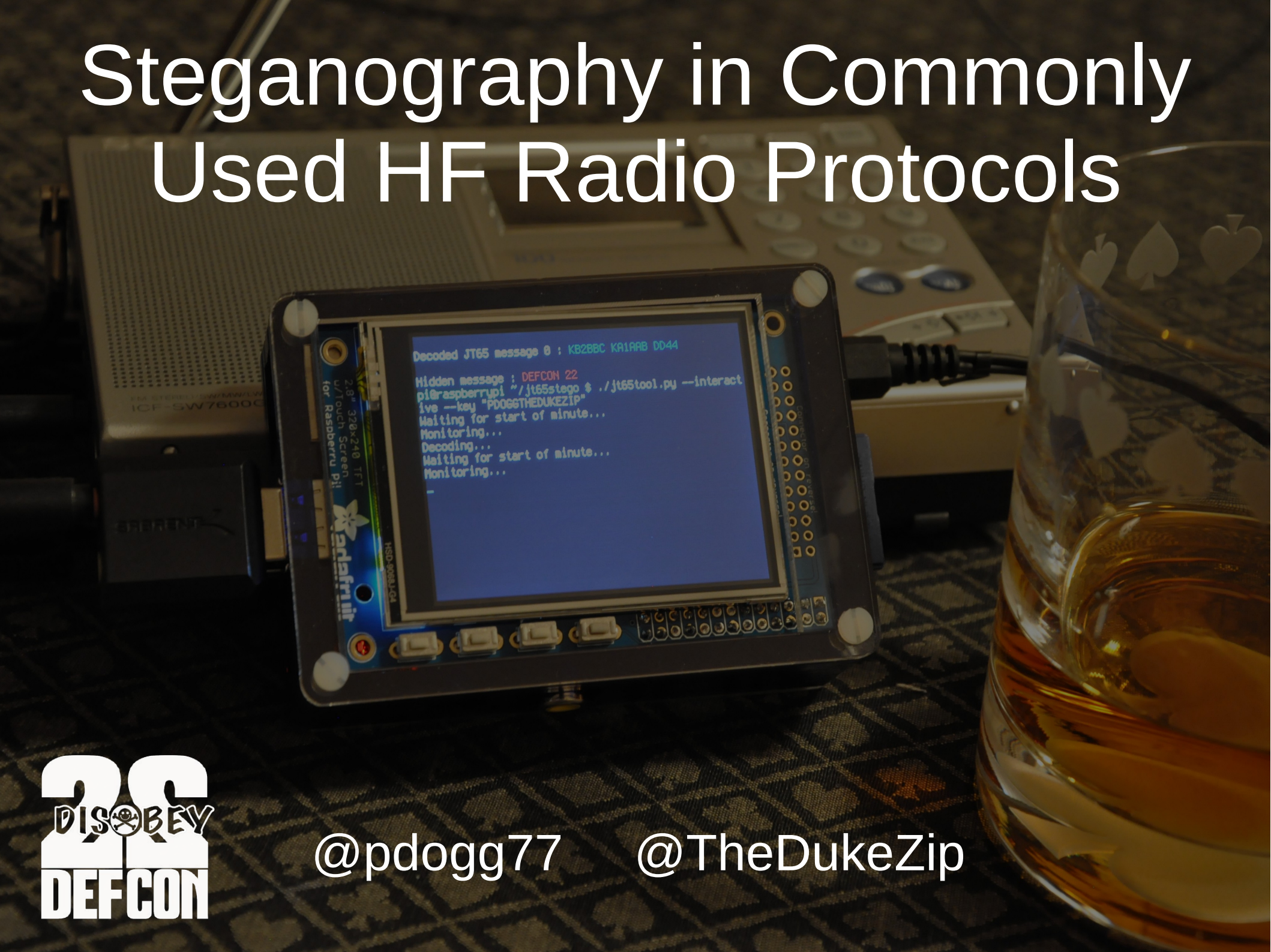
Steganography in Commonly Used HF Radio Protocols

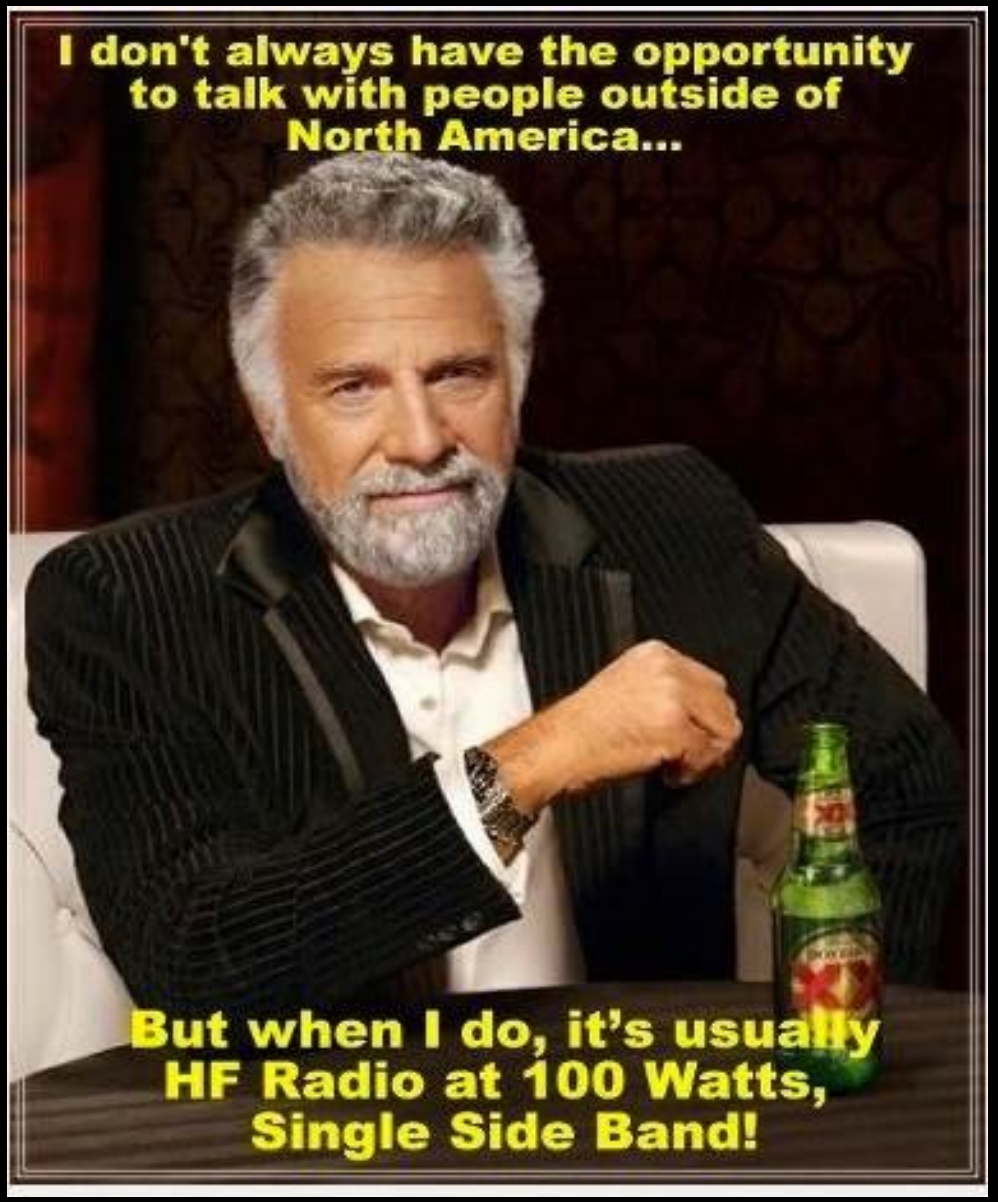


@pdogg77 @TheDukeZip

DISOBEY 22 DEFCON

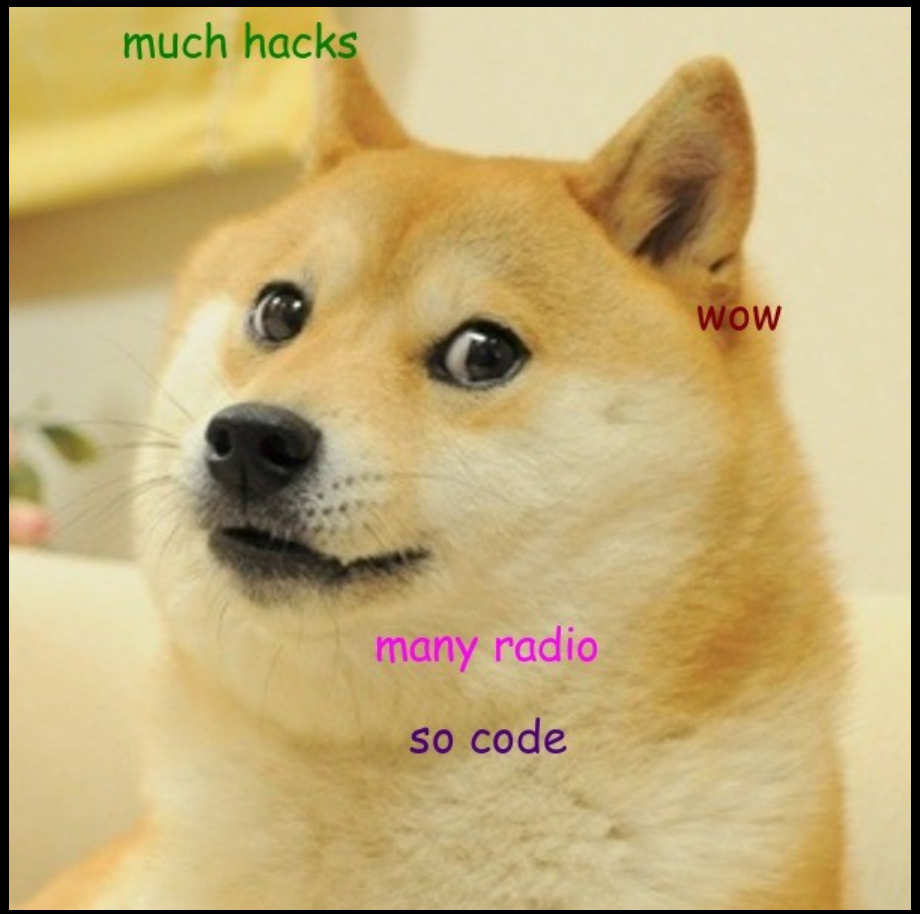
pdogg

- Paul / pdogg / @pdogg77
- Day Job: Security Researcher at Confer Technologies Inc.
- Hobby: Licensed as an amateur radio operator in 1986, ARRL VE
- This is my second trip to DEF CON



thedukezip

- Brent / thedukezip / @thedukezip
- Software &
  Systems Engineer (RF)
- Licensed ham radio op
  since 2006, ARRL VE



Why You Shouldn't Do This And Why We Didn't Do It On The Air

FCC Regulations (Title 47 – Part 97)

§ 97.113 Prohibited transmissions.

(a) No amateur station shall transmit:

…

(4) Music using a phone emission except as specifically provided elsewhere in this section; communications intended to facilitate a criminal act; messages encoded for the purpose of obscuring their meaning, except as otherwise provided herein; obscene or indecent words or language; or false or deceptive messages, signals or identification.

How This Project Started...
Final Warning Slide...

- Hackers + Drinks = ***Project***
- WANC - We are not cryptographers
- We are not giving cryptographic advice
- You should talk to a cryptographer
- If you are a cryptographer, we welcome your input

What?

We set out to demonstrate it was possible (or impossible) to create a:

- Low Infrastructure
- Long Range
- Covert
- Point to Point, Broadcast or Mesh
- Short Message Protocol

Using existing consumer radio and computer equipment, leveraging a commonly used digital mode

Why?

- Avoid censorship
- Avoid spying

- We believe you have the right to communicate without this interference
- You COULD use our method to communicate, OR use similar techniques to create your own method

… Or “The Terrorists”

No Internet?

Amateur radio operators have expertise in this!

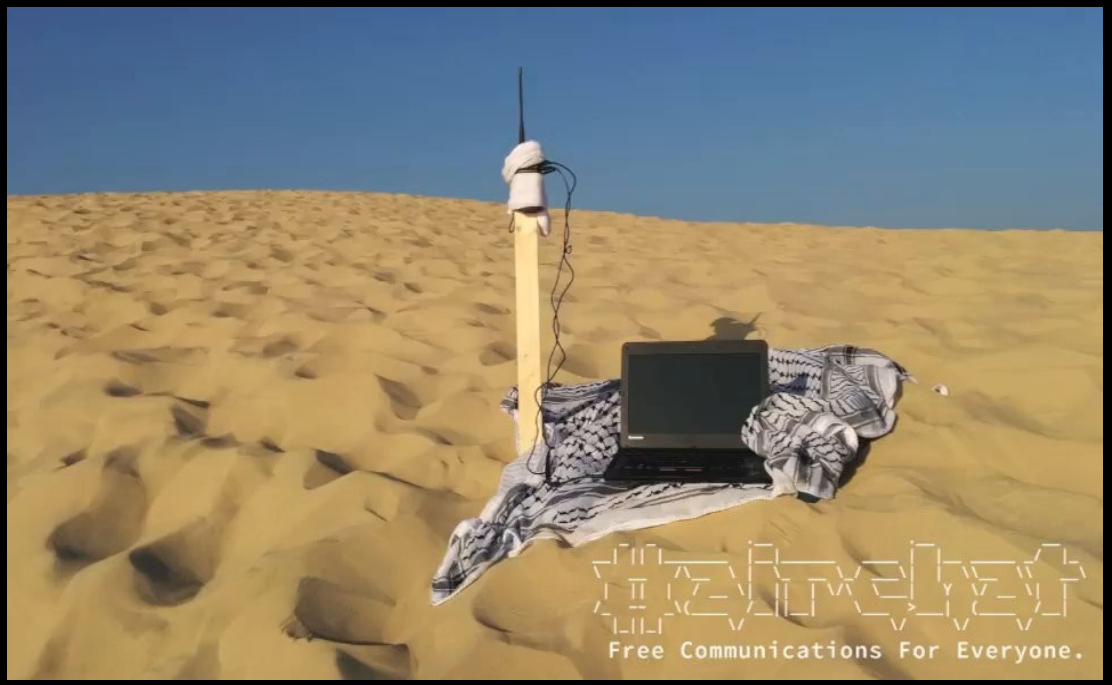
Amateur Radio

- Many frequency bands reserved for amateur radio operators to communicate
- Voice chat, digital modes...
- Take a multiple choice test to get licensed

- Reminder: The rules say you can't do what we're showing you...

AirChat

- Anonymous Lulzlabs
- Encrypted communication in plain sight
- Cool project with a different purpose
- Also breaks the rules



Good Steganography / Good OPSEC

- … means hiding well in plain sight.
- Invisible to normal users
- “Plausible deniability”
- Not this →

More Like This

NOT THIS!

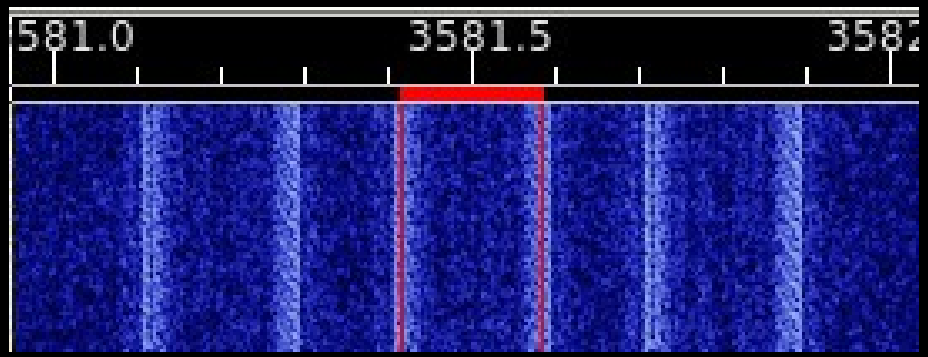
Guns == Good! Smartphones == BAD :)

Ways to Hide...

- Protocol features (headers, checksums etc)
- Timing or substitution
- Errors
- No “spurious emissions” etc... (against the rules, obvious, very “visible”)
- Candidate Protocol must:
  - … be in widespread common use
  - … have places to hide
  - … be relatively power efficient
  - Need no special hardware or closed software

Popular Sound Card Digital Modes

- RTTY
  - In use on radio since at least the 1920s
  - Baudot code – 5 bit symbols with a stop and a shift – “mark and space”
  - Amateurs almost always use a 45 baud version with 170hz carrier shift
  - Limited character set
- PSK31 etc.
  - Phase shift keying 31 baud...
  - Developed by Peter Martinez G3PLX in 1998
  - VERY tight protocol - “Varicode”



JT65

- Developed by Joe Taylor – K1JT – 2005
- Original paper: “The JT65 Communications Protocol”
- Designed for Earth-Moon-Earth communications. Also now widely used for skywave contacts
- Very power efficient
- Structured communication, very low data rate
- Open source implementation

JT65 Conversations

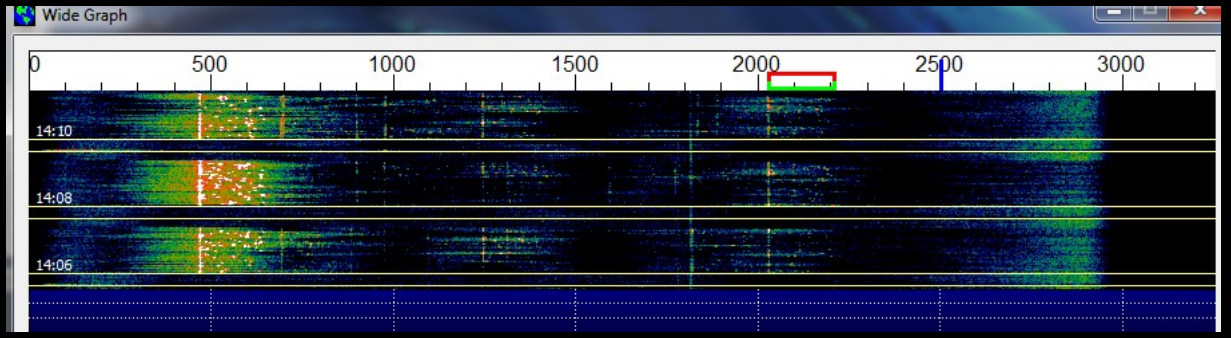
Some Common HF Ham Freqs:

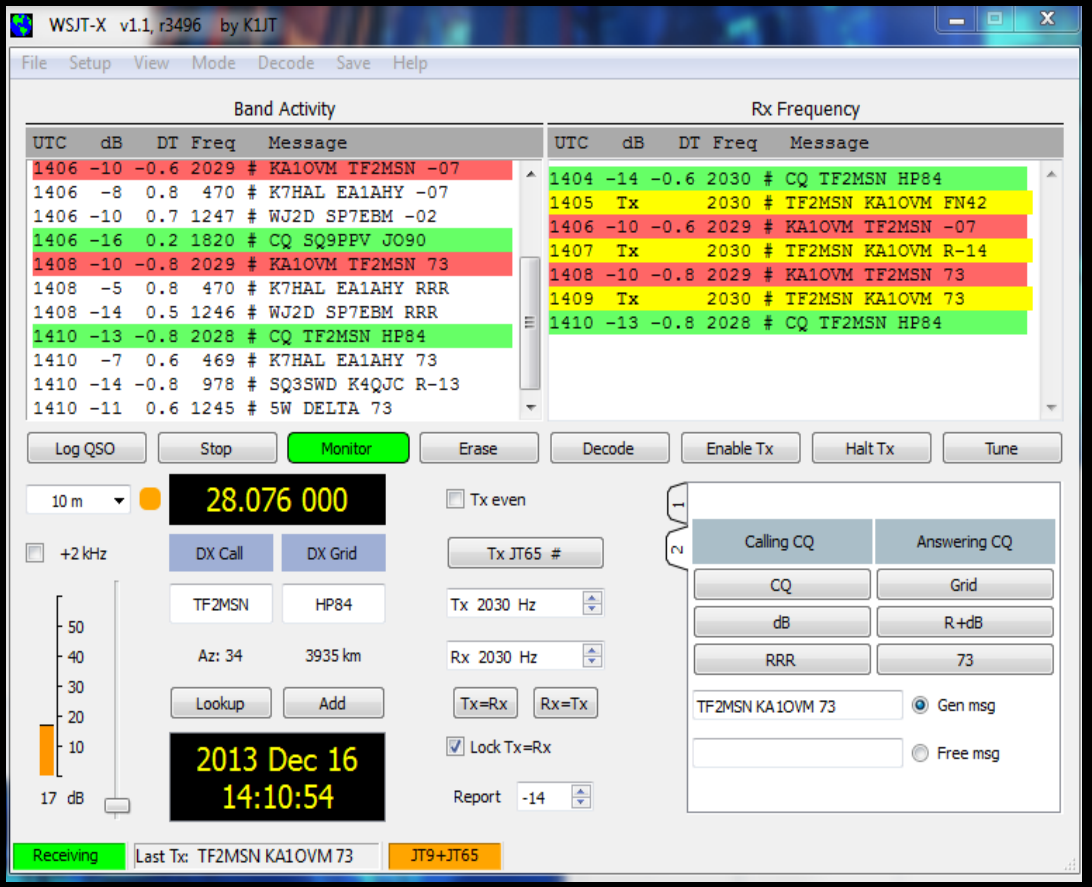
20m 14.076MHz

15m 21.076MHz

10m 28.076MHz

Upper Side Band





Some JT65 Technical Details

KB2BBC KA1AAB DD44

User Message

34 20 05 42 26 09 03 05 60 06 24 22

Source Encoding

58 11 22 20 52 15 02 29 24 61 19 32 51 08 31 55 47 13 10 49 59
55 30 30 30 38 57 19 22 24 13 18 11 29 15 09 17 24 25 19 31 57
44 21 58 13 08 61 27 29 52 34 20 05 42 26 09 03 05 60 06 24 22

FEC

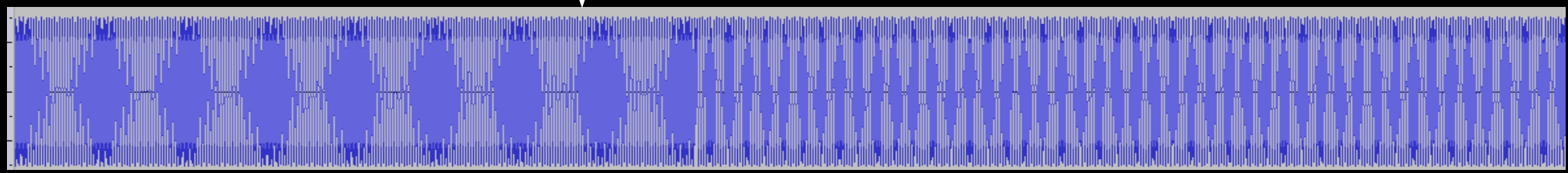
58 11 22 20 52 15 02
29 24 61 19 32 51 08
31 55 47 13 10 49 59
55 30 30 30 38 57 19
22 24 13 18 11 29 15
09 17 24 25 19 31 57
44 21 58 13 08 61 27
29 52 34 20 05 42 26
09 03 05 60 06 24 22

Matrix Interleaving

58 29 31 55 22 09 44 29 09 11 24 55 30 24 17 21 52 03 22 61 47
30 13 24 58 34 05 20 19 13 30 18 25 13 20 60 52 32 10 38 11 19
08 05 06 15 51 49 57 29 31 61 42 24 02 08 59 19 15 57 27 26 22

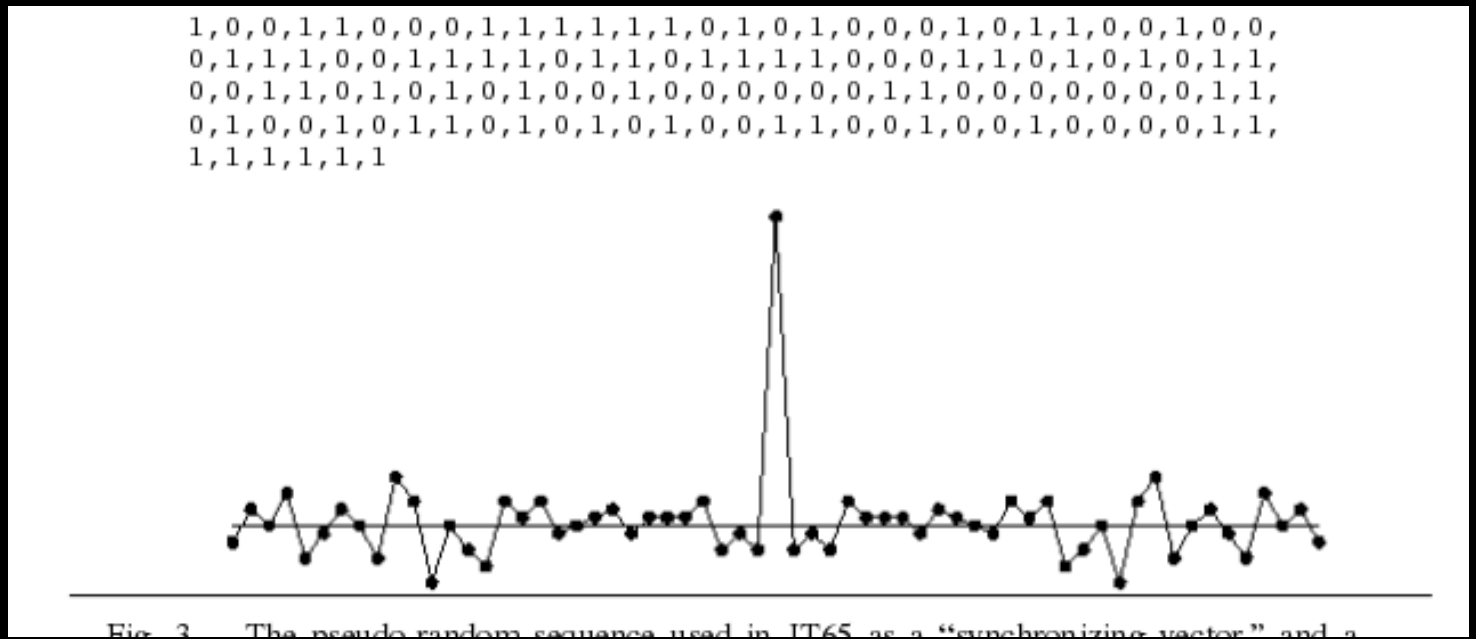
39 19 16 44 29 13 58 19 13 14 20 44 17 20 25 31 46 02 29 35 56
17 11 20 39 51 07 30 26 11 17 27 21 11 30 34 46 48 15 53 14 26
12 07 05 08 42 41 37 19 16 35 63 20 03 12 38 26 08 37 22 23 29

Gray Coding



Audio

- JT65 “packet” sliced into 126 .372s intervals – 47.8s
- 1270.5 Hz sync tone - “pseudo-random synchronization vector”
- Symbols - 1270.5 + 2.6917(N+2)m Hz
  - N is the integral symbol value, 0 ≤ N ≤ 63
  - m assumes the values 1, 2, and 4 for JT65 sub-modes A, B, and C



Hiding in Reed Solomon Codes

- Exploit error correction!
- Easy/PoC Mode: Shove in some errors... :) (static “key”)
- Medium mode: Shove in errors, add some random cover
- Hard Mode: Encrypt and pack message, add FEC
- Prior Work: Hanzlik, Peter “Steganography in Reed-Solomon Codes”, 2011

Encoding Steganography (Basic)

Steg: DEF CON 22

Source Encoding:

```
19 51 00 26 06 17 52 04 31 15 56 28
```

FEC:

```
24 42 21 21 43 42 56 22 19 51 00 26 06 17 52 04 31 15 56 28
```

Can tolerate 4 errors

Hiding Steganography

Key: pdogg thedukezip

Generate 20 'locations' based on SHA512

09 29 41 35 20 32 27 15 23 18 53 12 45 03 13 40 49 22 25 37

Injecting Errors

JT65: KB2BBC KA1AAB DD44

```
39 19 16 44 29 13 58 19 13 14 20 44 17 20 25 31 46 02 29 35 56
17 11 20 39 51 07 30 26 11 17 27 21 11 30 34 46 48 15 53 14 26
12 07 05 08 42 41 37 19 16 35 63 20 03 12 38 26 08 37 22 23 29
```

Steg: DEF CON 22

```
24 42 21 21 43 42 56 22 19 51 00 26 06 17 52 04 31 15 56 28
```

Key: pdogg thedukezip

```
09 29 41 35 20 32 27 15 23 18 53 12 45 03 13 40 49 22 25 37
```

Injecting Errors

JT65: KB2BBC KA1AAB DD44

39	19	16	44	29	13	58	19	13	14	20	44	17	20	25	31	46	02	29	35	56
17	11	20	39	51	07	30	26	11	17	27	21	11	30	34	46	48	15	53	14	26
12	07	05	08	42	41	37	19	16	35	63	20	03	12	38	26	08	37	22	23	29

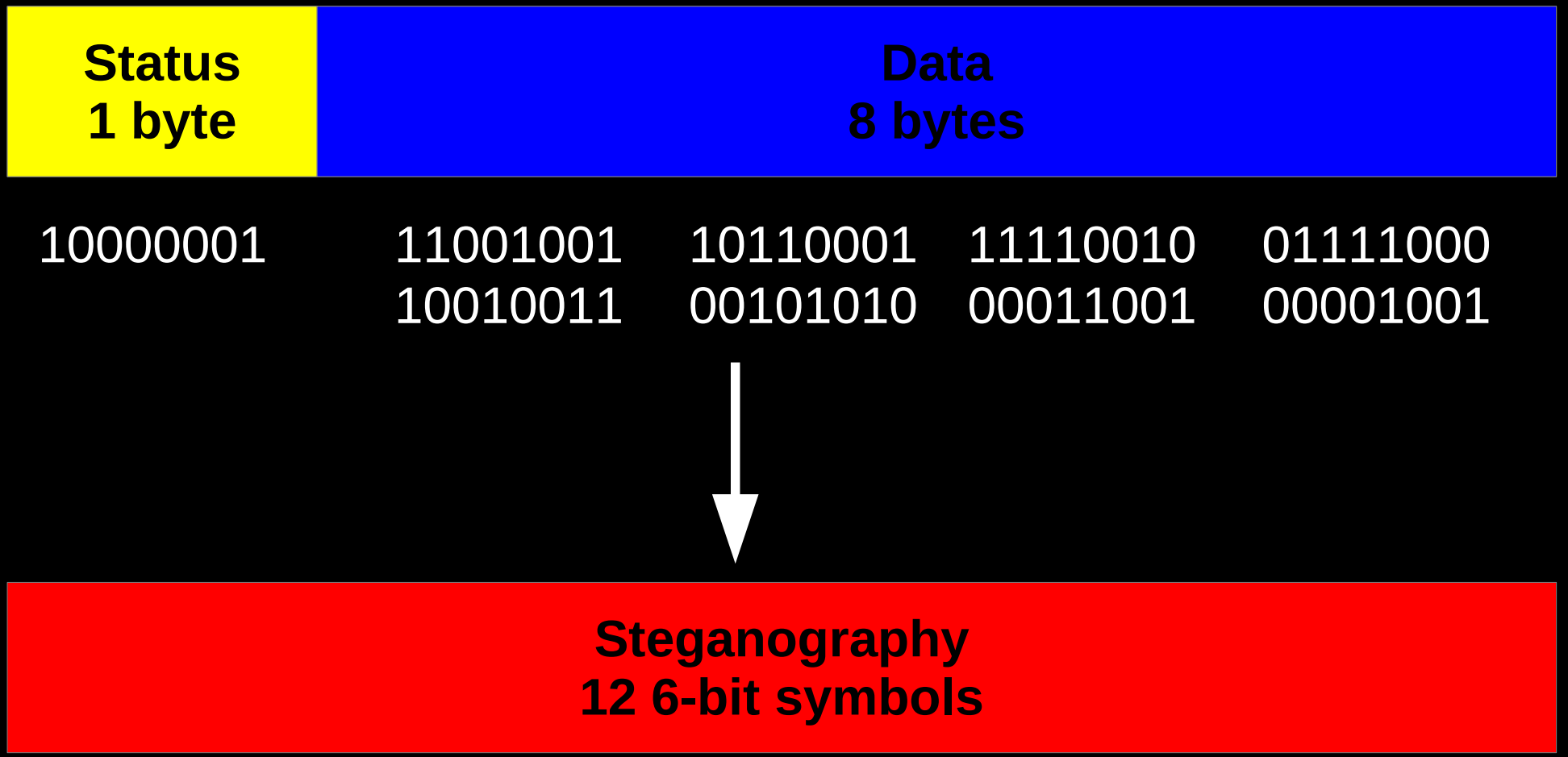
JT65: KB2BBC KA1AAB DD44
Steg: DEF CON 22
Key: pdogg thedukezip

39	19	16	17	29	13	58	19	13	24	20	44	26	52	25	22	46	02	51	35	43
17	15	19	39	56	07	56	26	42	17	27	42	11	30	21	46	28	15	53	04	21
12	07	05	06	42	41	37	31	16	35	63	00	03	12	38	26	08	37	22	23	29

What About Encryption?

- We have 12 * 6 = 72 bits to play with
- We need 8 bit bytes...
- Well that gives us exactly 9 bytes

“Packing” Function



“Status” Byte

Status
1 byte

- Track how many total packets in message
- Flags for first / last packet
- Track size for stream ciphers

```
Waiting for start of minute...

Decoded JT65 message 0 : KB2BBC KA1AAB DD44

Steg detected! (1/3) total packets received.
Monitoring...
Decoding...
Waiting for start of minute...

Decoded JT65 message 0 : KA1AAB KB2BBC DD44

Steg detected! (2/3) total packets received.
Monitoring...
Decoding...
Waiting for start of minute...

Decoded JT65 message 0 : KB2BBC KA1AAB DD44

Hidden message : SEE YOU AT DEF CON 22
```

“Status” Byte – Stream Cipher

First packet: `(0x80) | (# of total packets)`

Middle packets: `Packet Number`

Last packet: `(0x40) | (# of bytes in packet)`

Max size: 64 packets (512 bytes)

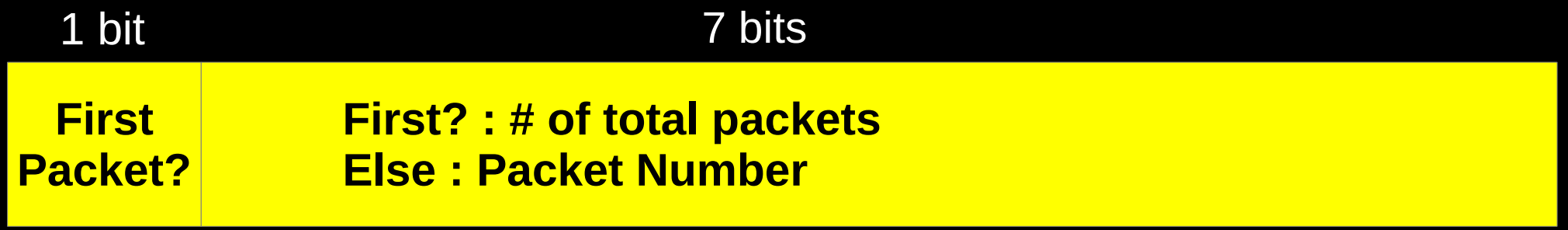
1 bit	1 bit	6 bits
First Packet?	**Last Packet?**	**First? : # of total packets** **Last? : # of bytes in packet** **Else : Packet Number**

“Status” Byte – Block Cipher

First packet: `(0x80) | (# of total packets)`

Other packets: `Packet Number`

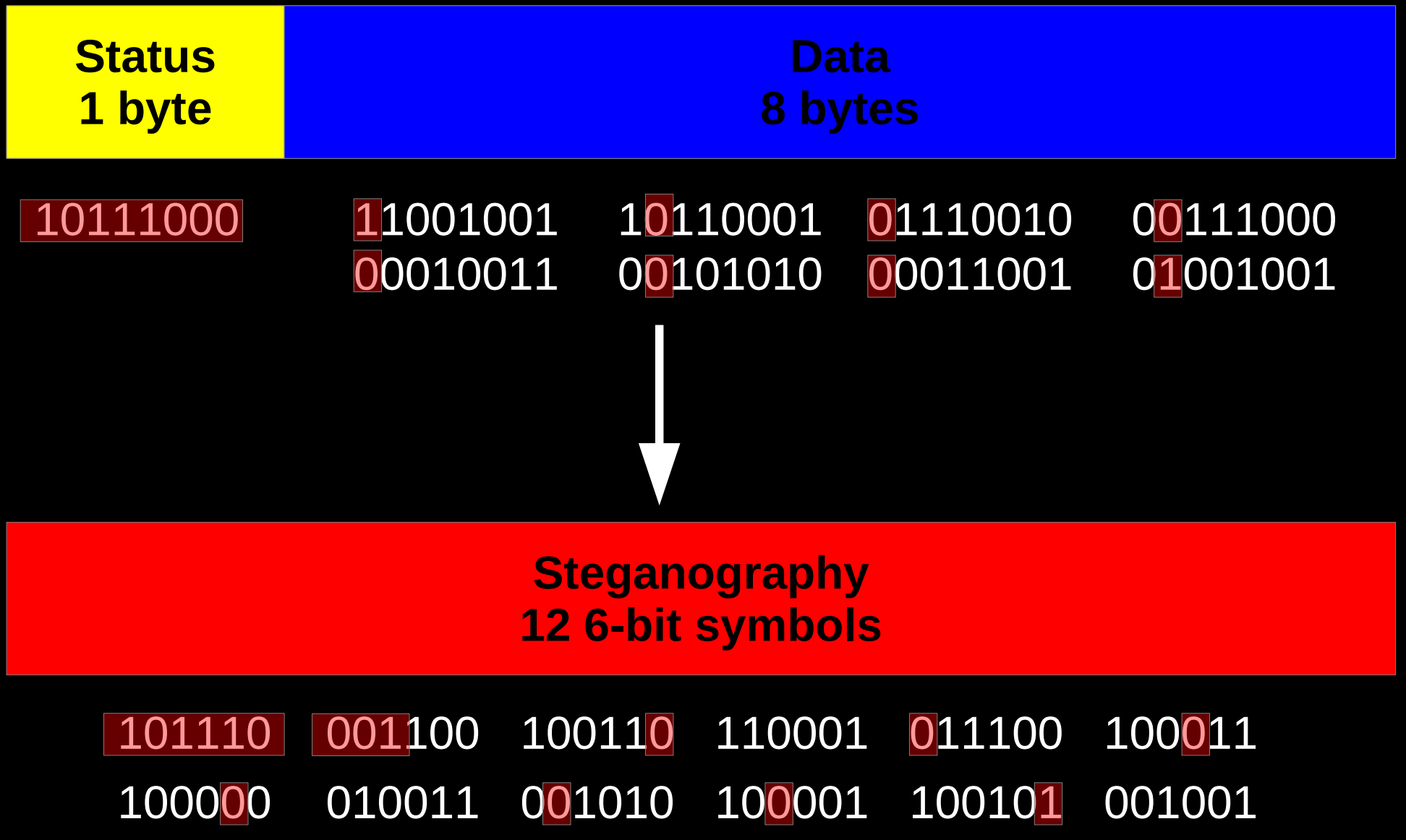
Max size: 128 packets (1024 bytes)

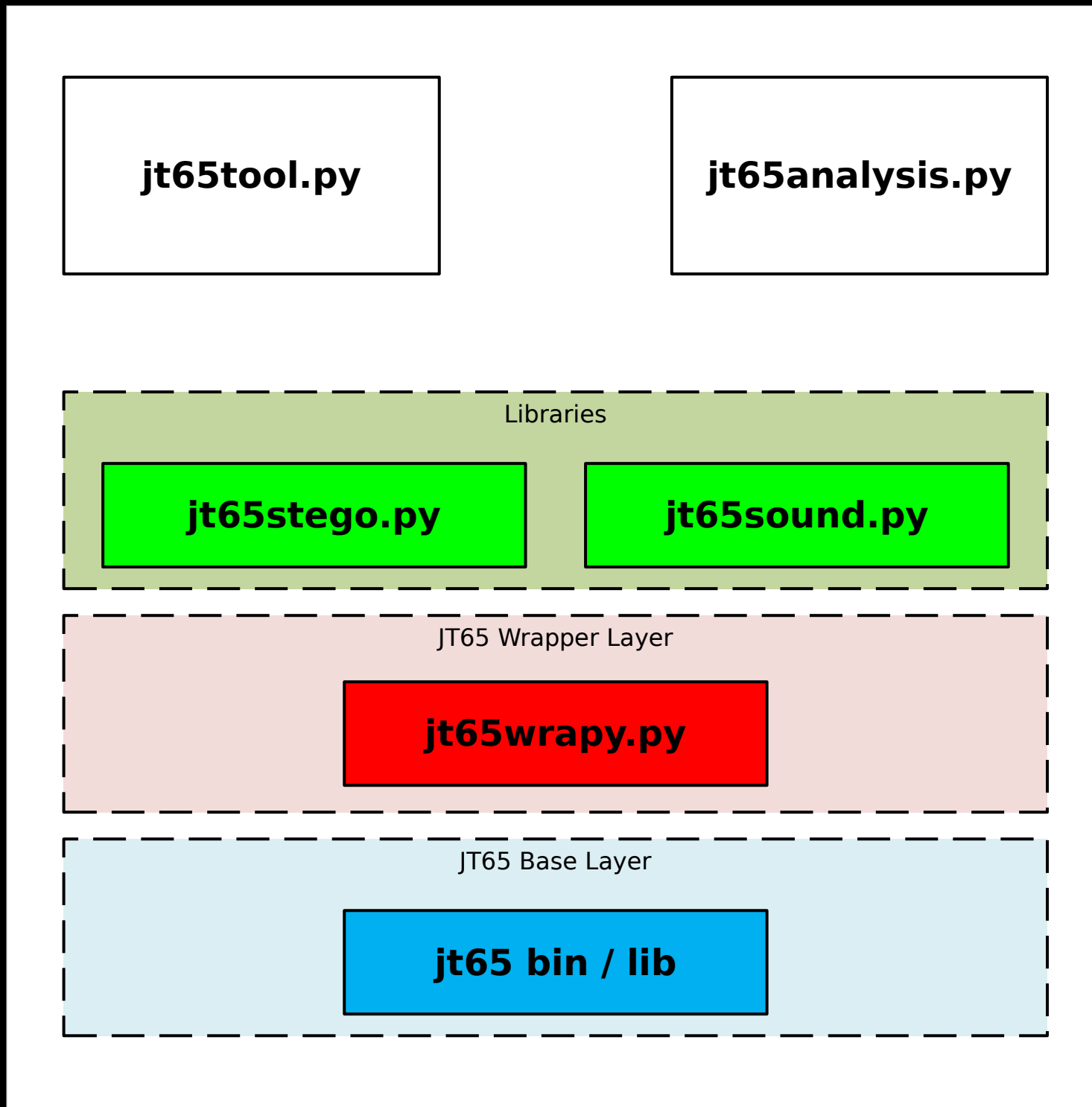


Hiding the Status Byte

- We'll talk about analysis in a bit...
- Steganography traffic was trivial to pick out of normal traffic because of this byte :(

Perform Bit Swap




jt65tool.py
jt65analysis.py
Libraries
jt65stego.py
jt65sound.py
JT65 Wrapper Layer
jt65wrapy.py
JT65 Base Layer
jt65 bin / lib

Tool Demo...

“Feed Reader” RasPi Demo...

Analysis/Steganalysis

- Defined set of legitimate JT65 packets
- “Known Cover Attack”
- Receive packet → Decode → Encode
- Demodulator provides “probability” or confidence
- Theory:
  - Packets suspected to contain steganography can be easily distinguished by some quantitative measure

Known Cover

JT65: KB2BBC KA1AAB DD44

39	19	16	44	29	13	58	19	13	14	20	44	17	20	25	31	46	02	29	35	56
17	11	20	39	51	07	30	26	11	17	27	21	11	30	34	46	48	15	53	14	26
12	07	05	08	42	41	37	19	16	35	63	20	03	12	38	26	08	37	22	23	29

JT65: KB2BBC KA1AAB DD44
Steg: DEF CON 22
Key: pdogg thedukezip

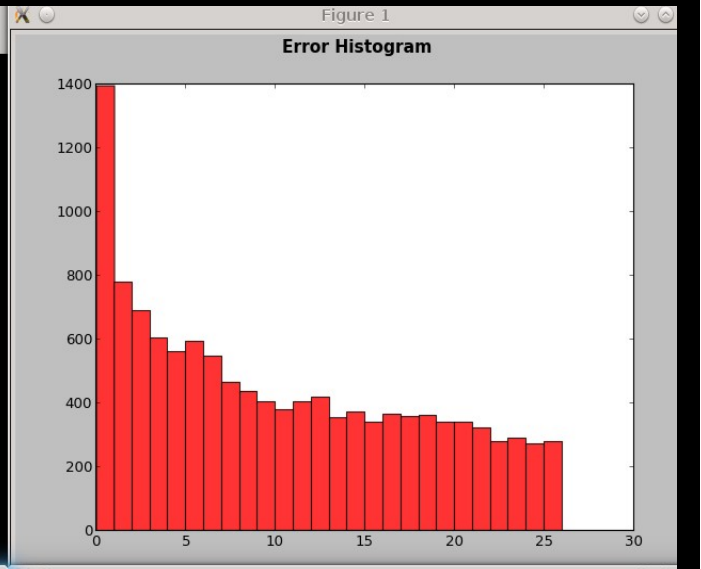
39	19	16	17	29	13	58	19	13	24	20	44	26	52	25	22	46	02	51	35	43
17	15	19	39	56	07	56	26	42	17	27	42	11	30	21	46	28	15	53	04	21
12	07	05	06	42	41	37	31	16	35	63	00	03	12	38	26	08	37	22	23	29

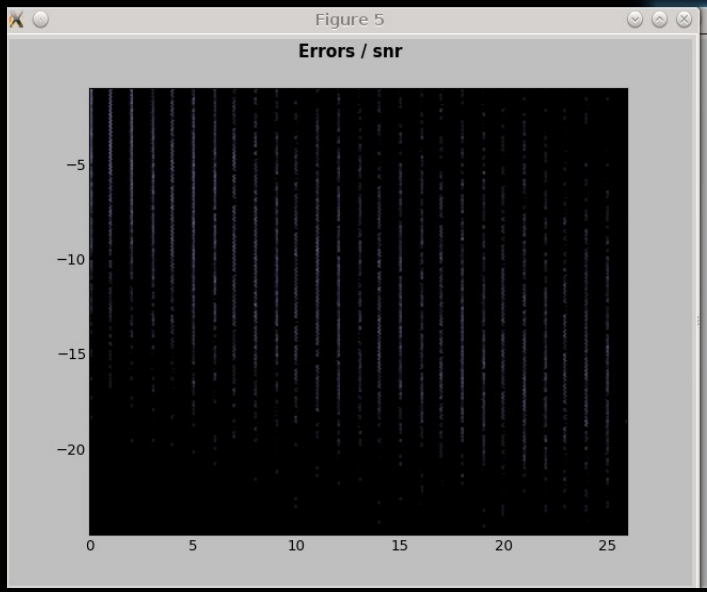
Analysis Module

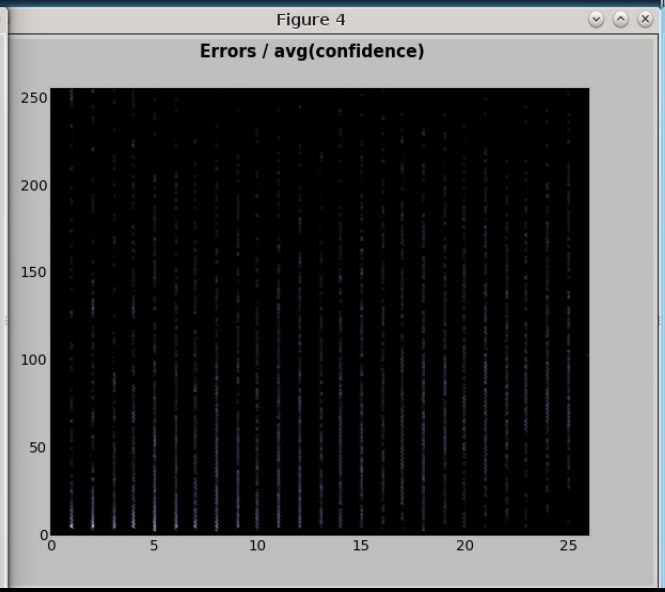
~/code/gits/jt65stego : python

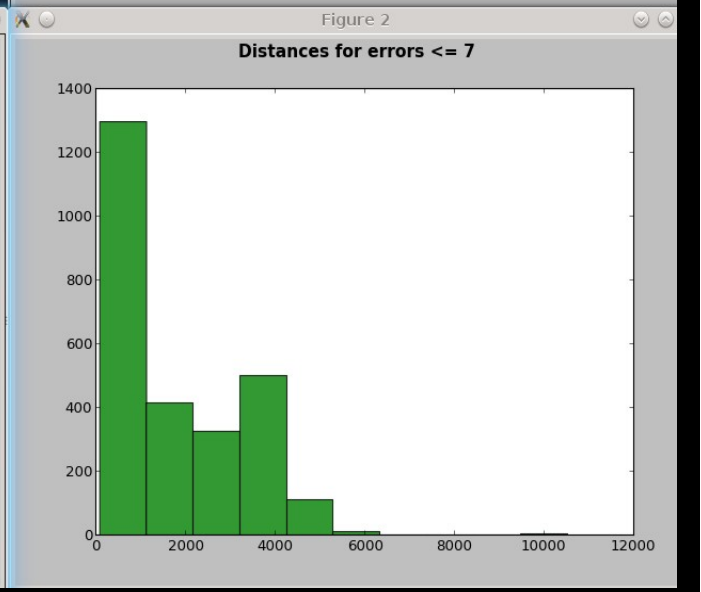
File Edit View Bookmarks Settings Help

```
:~/code/gits/jt65stego$ python jt65analysis.py --distance FN42 --text 20mpacketsoutput.txt
Number of packets in file:                    11934


Median Number of Errors:                      8.0
Average Number of Errors:                     9.68694486342
Standard Deviation of Errors:                 7.68047167928
Error Bins:
[1396  778  690  603  560  592  548  464  434  402  379  403  417  355  370
  339  366  357  362  341  341  322  277  288  270  276    4    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0]


Median SNR:                                   -8.58761152965
Average SNR:                                  -8.8181056207
Standard Deviation SNR:                       5.57110168923


Number of packets in set with 7 or less errors: 5631
       2659 have distance data
       Max Distance of Set:                   10518.49107
       Median Distance of Set:                1179.20450965
       Average Distance of Set:               1858.87448763
       90% Distance of Set:                   3687.56180076
Press Enter to continue...
```

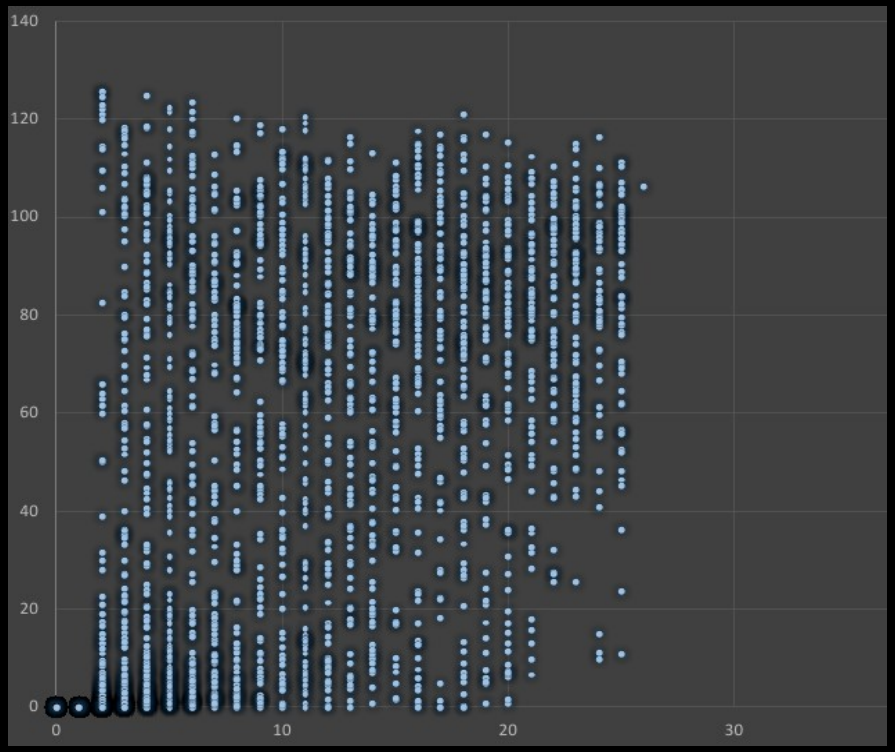


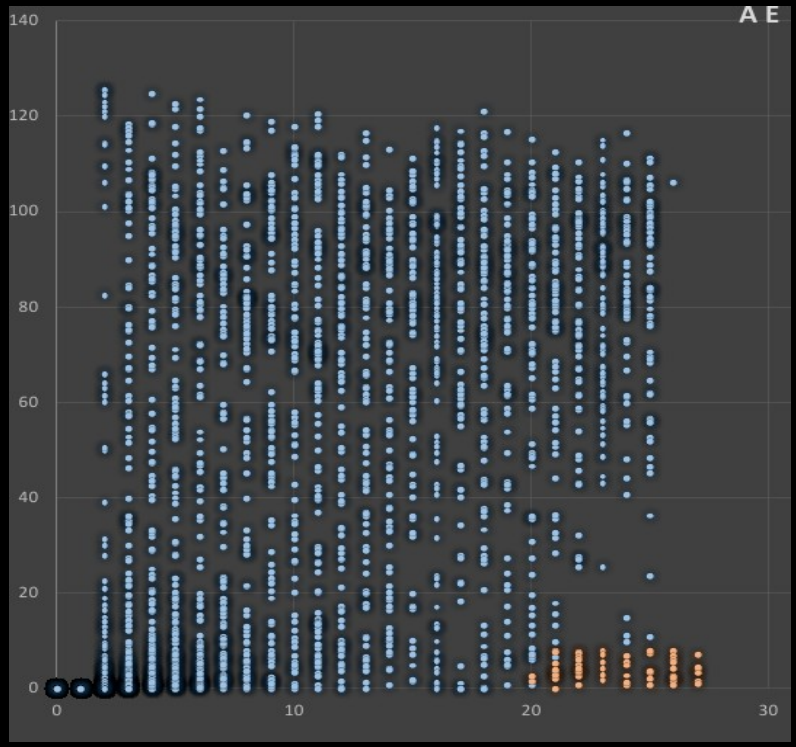




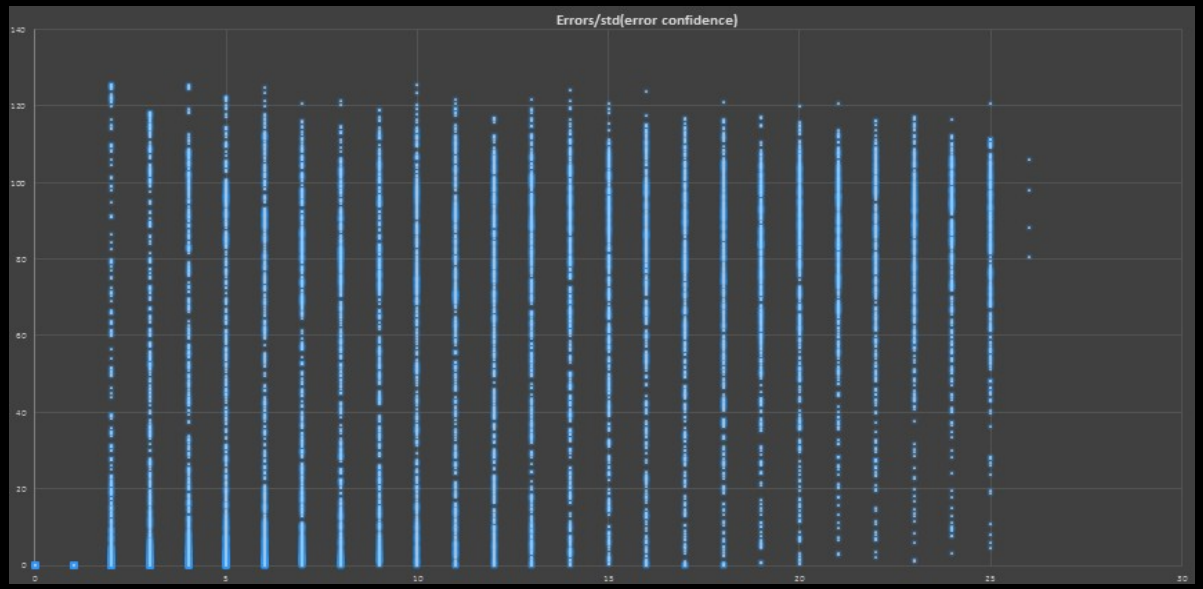


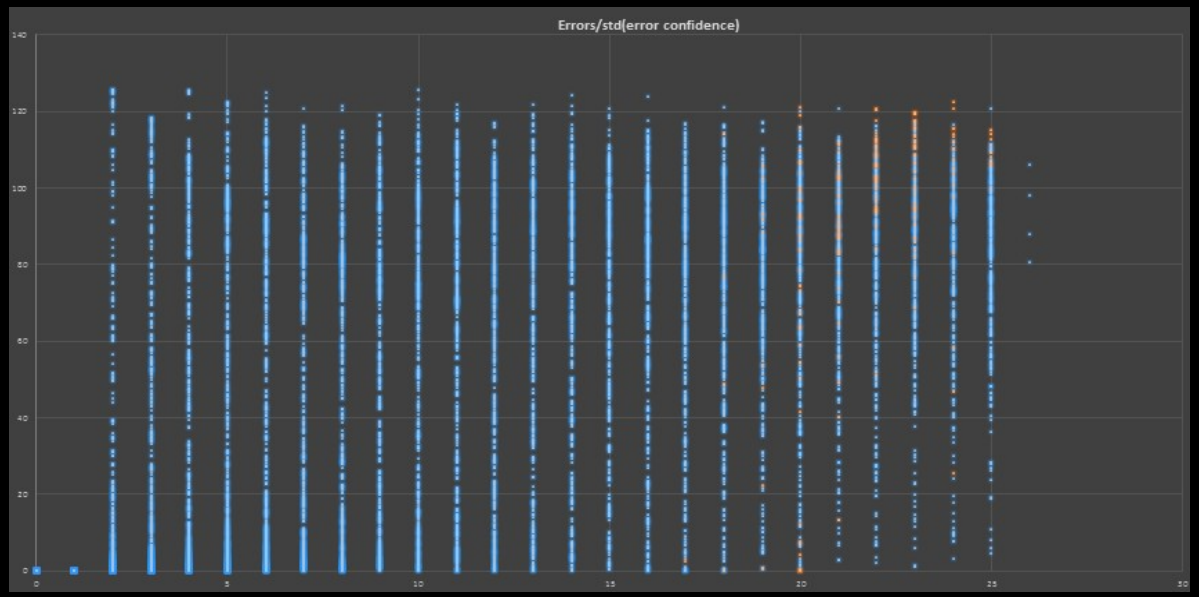
Finding Steganography is Easy



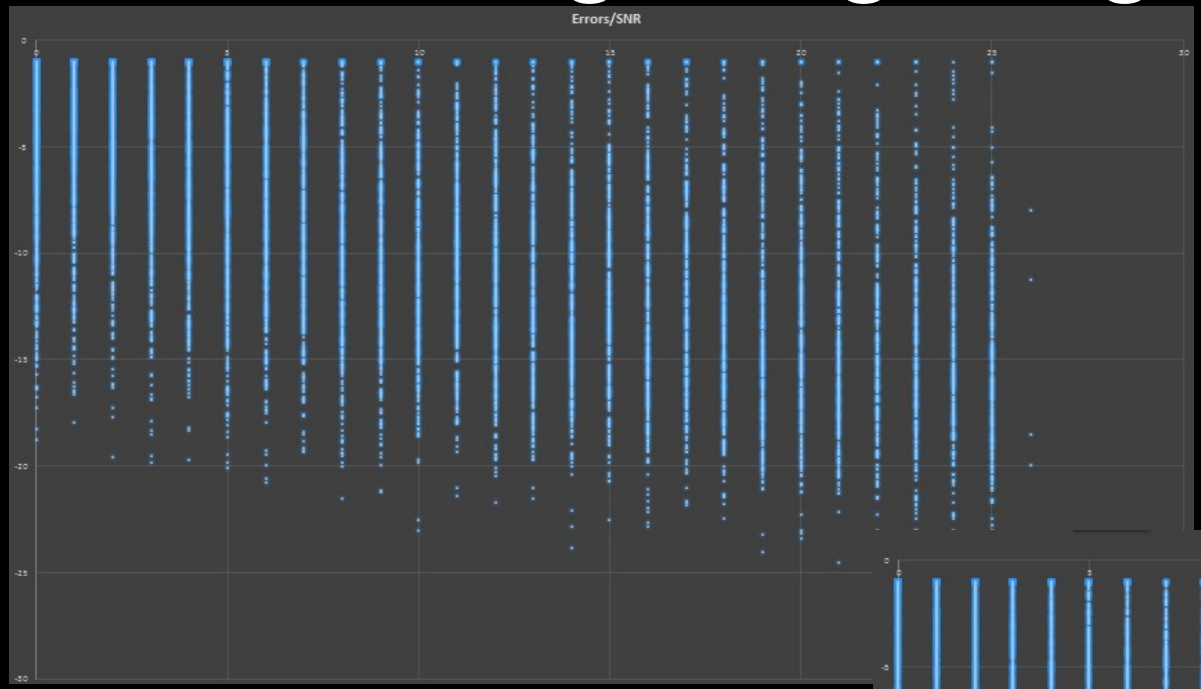


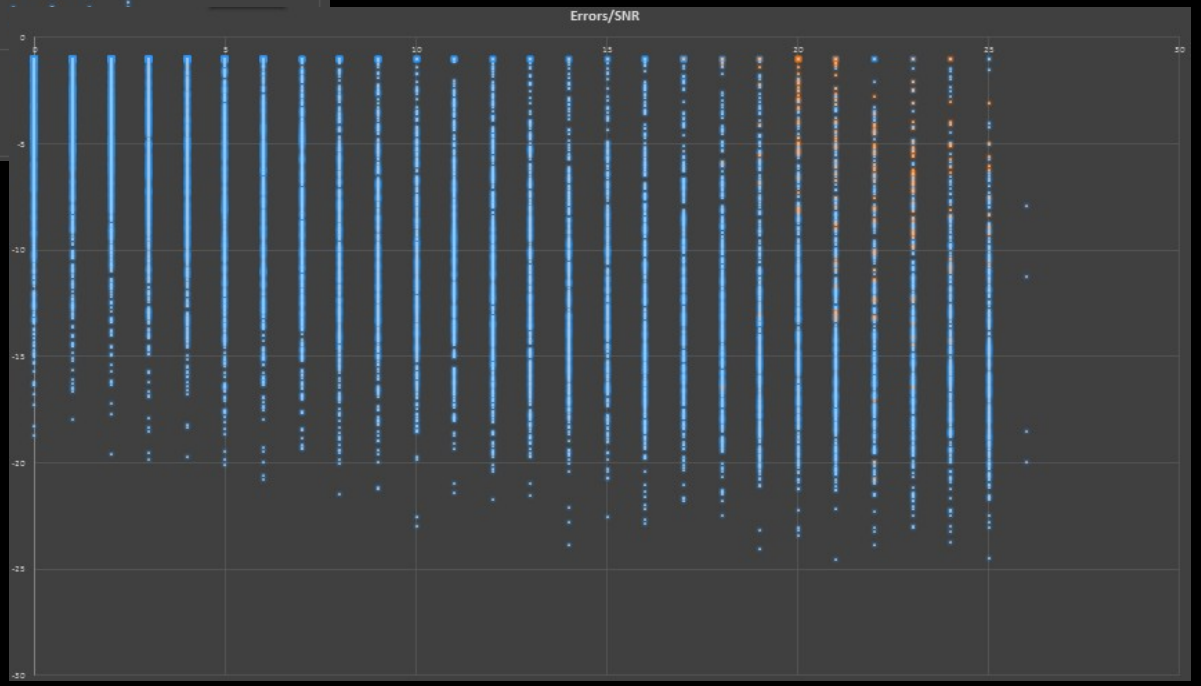
Finding Steganography is Hard



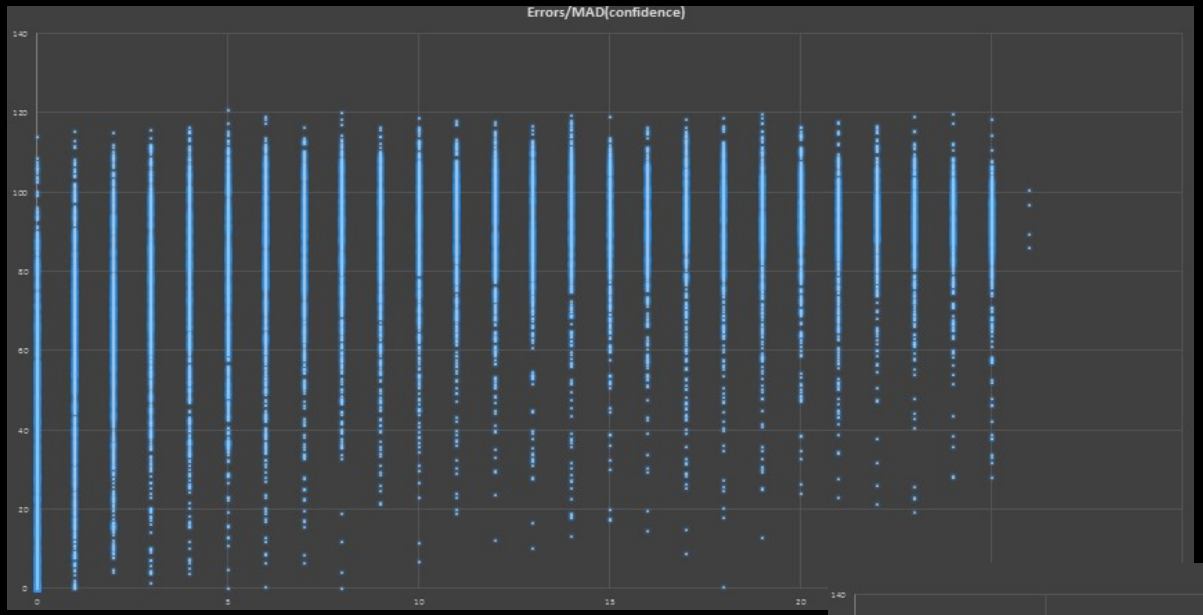


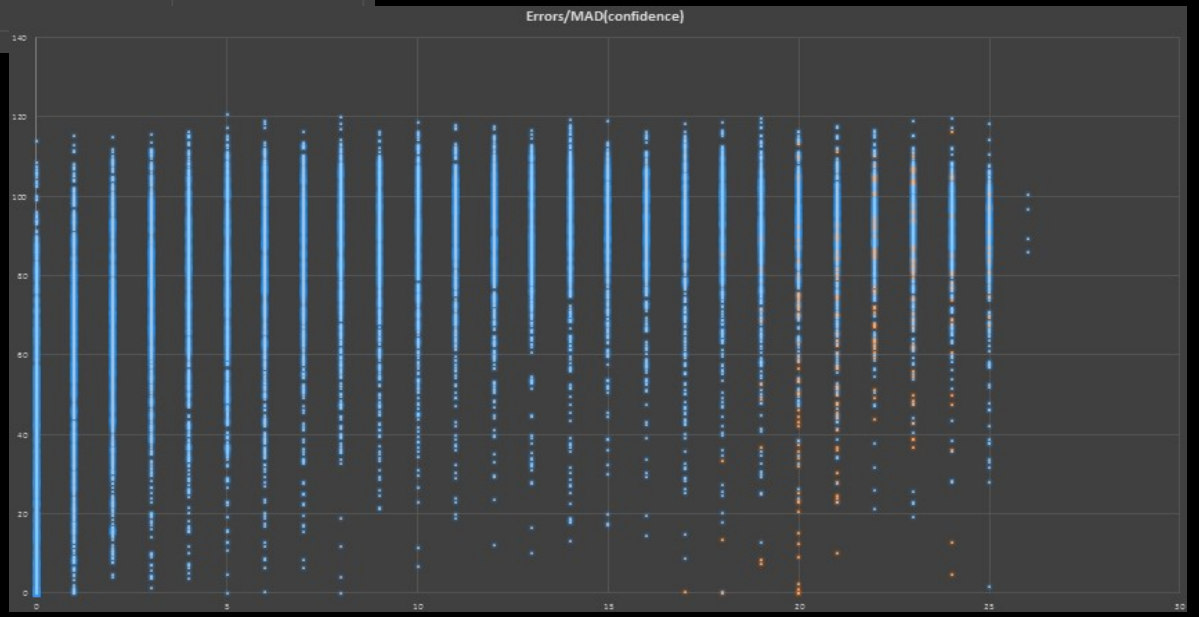
Finding Steganography is Hard



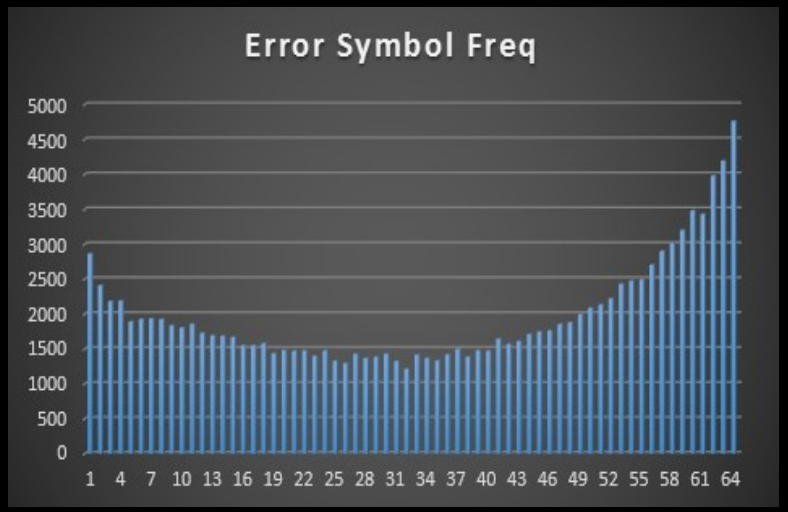


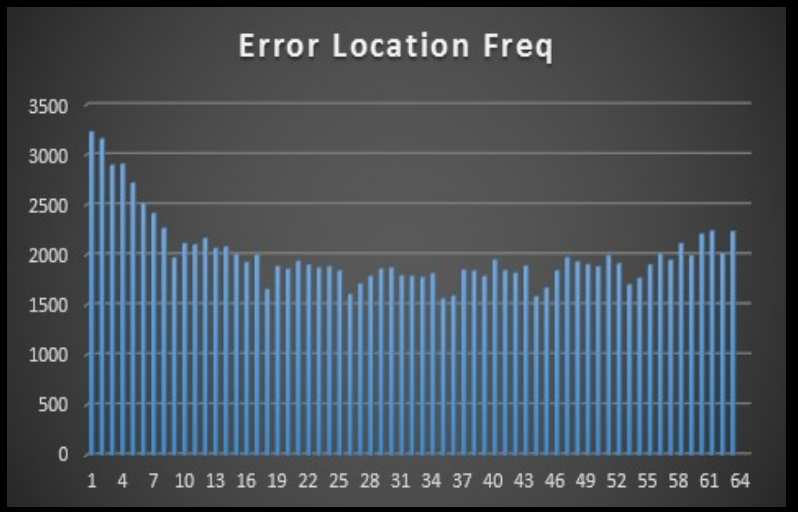
Finding Steganography is Hard

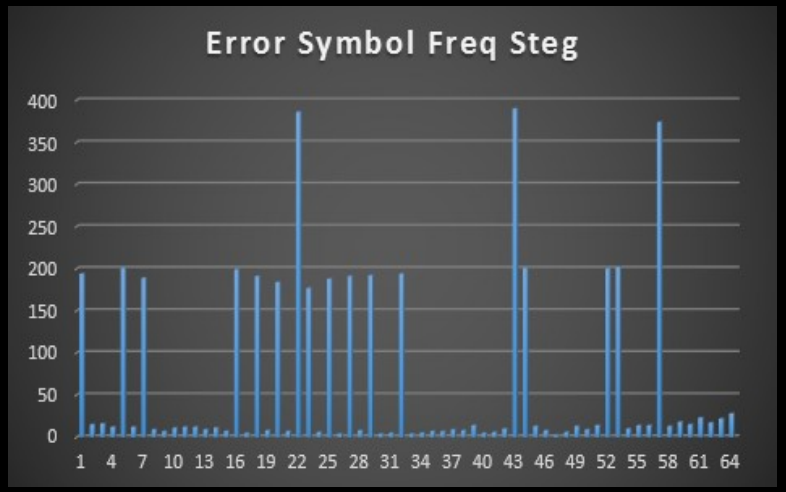


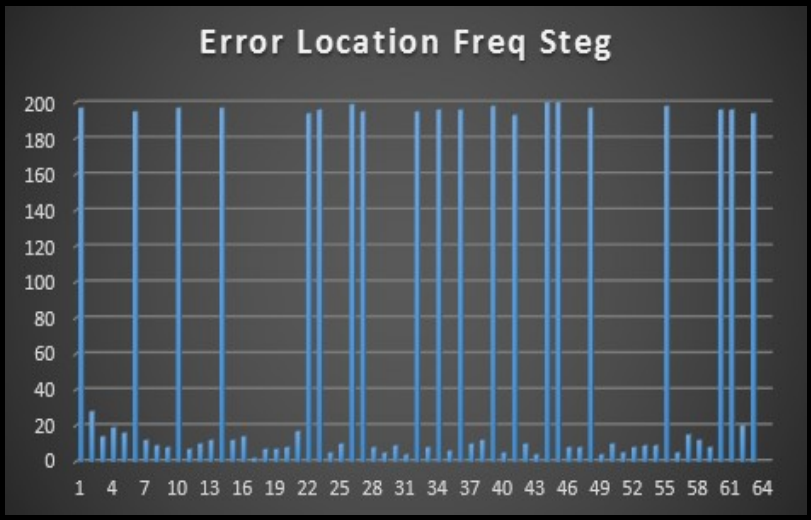


Interesting Patterns (and a warning)





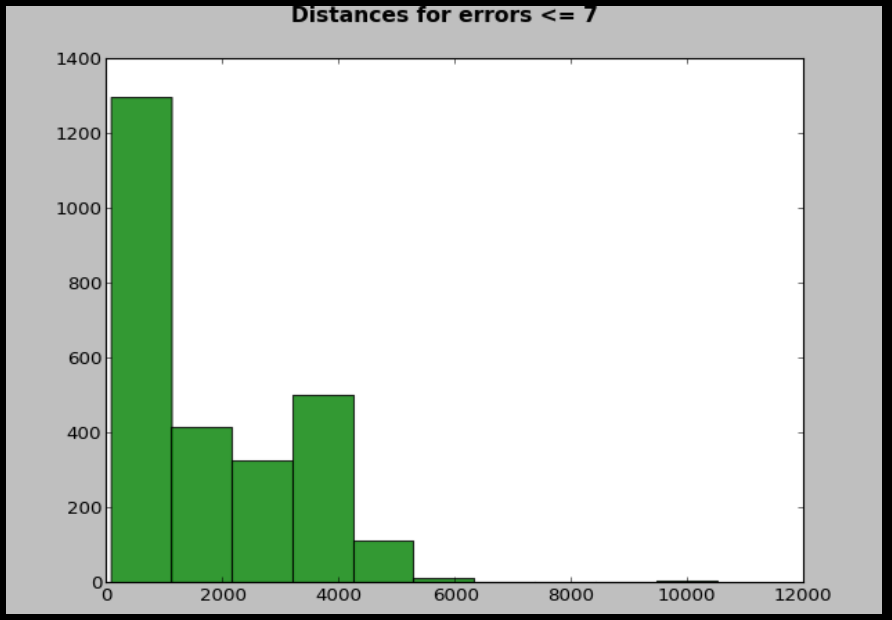




Distance

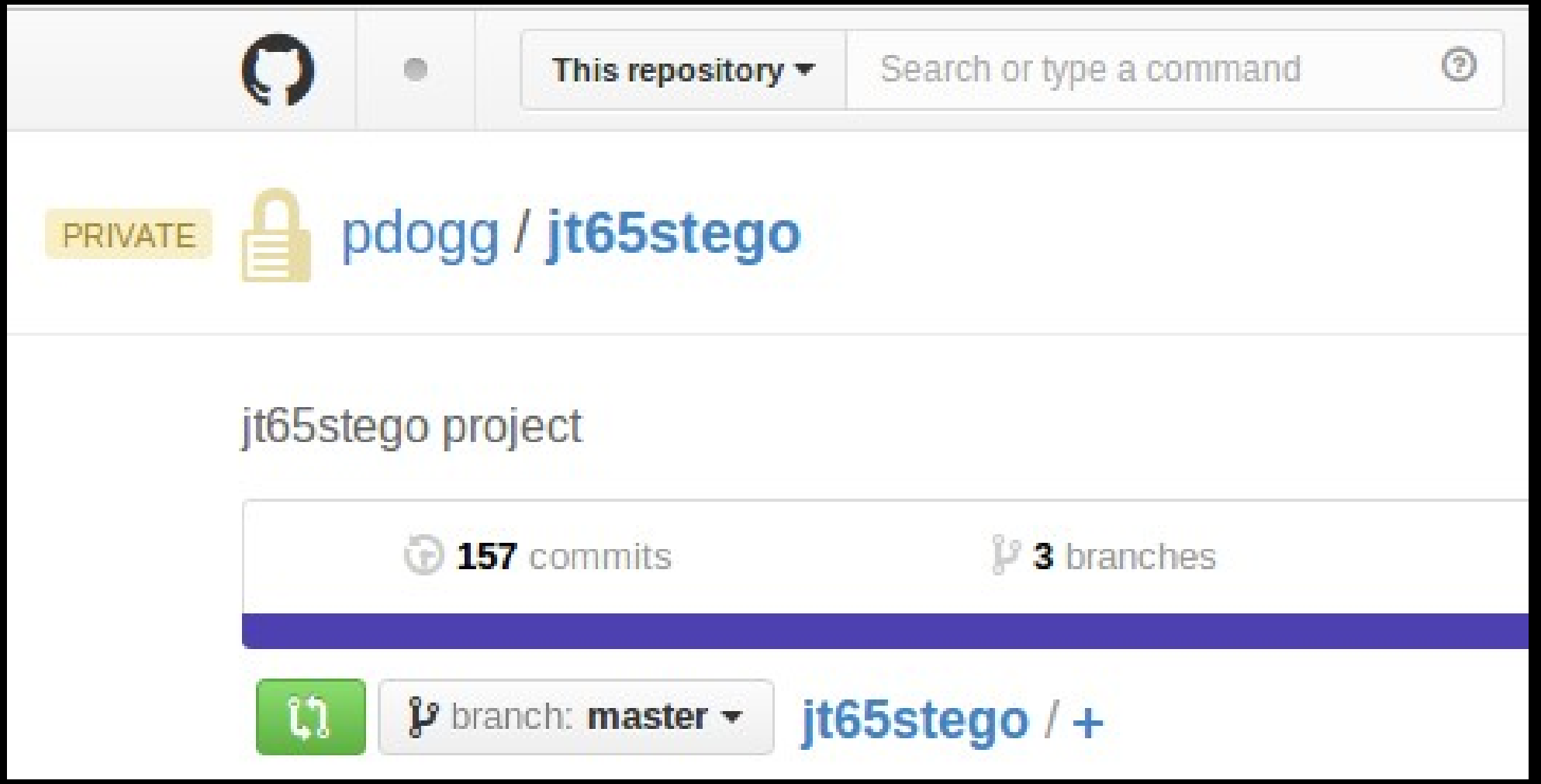
- Considering we cannot SEND these packets
- Let's pretend we received them (<= 7 errors)
- How far away were the senders?

```
Number of packets in set with 7 or less errors: 5631
        2659 have distance data
        Max Distance of Set:                    10518.49107
        Median Distance of Set:                 1179.20450965
        Average Distance of Set:                1858.87448763
        90% Distance of Set:                    3687.56180076
```

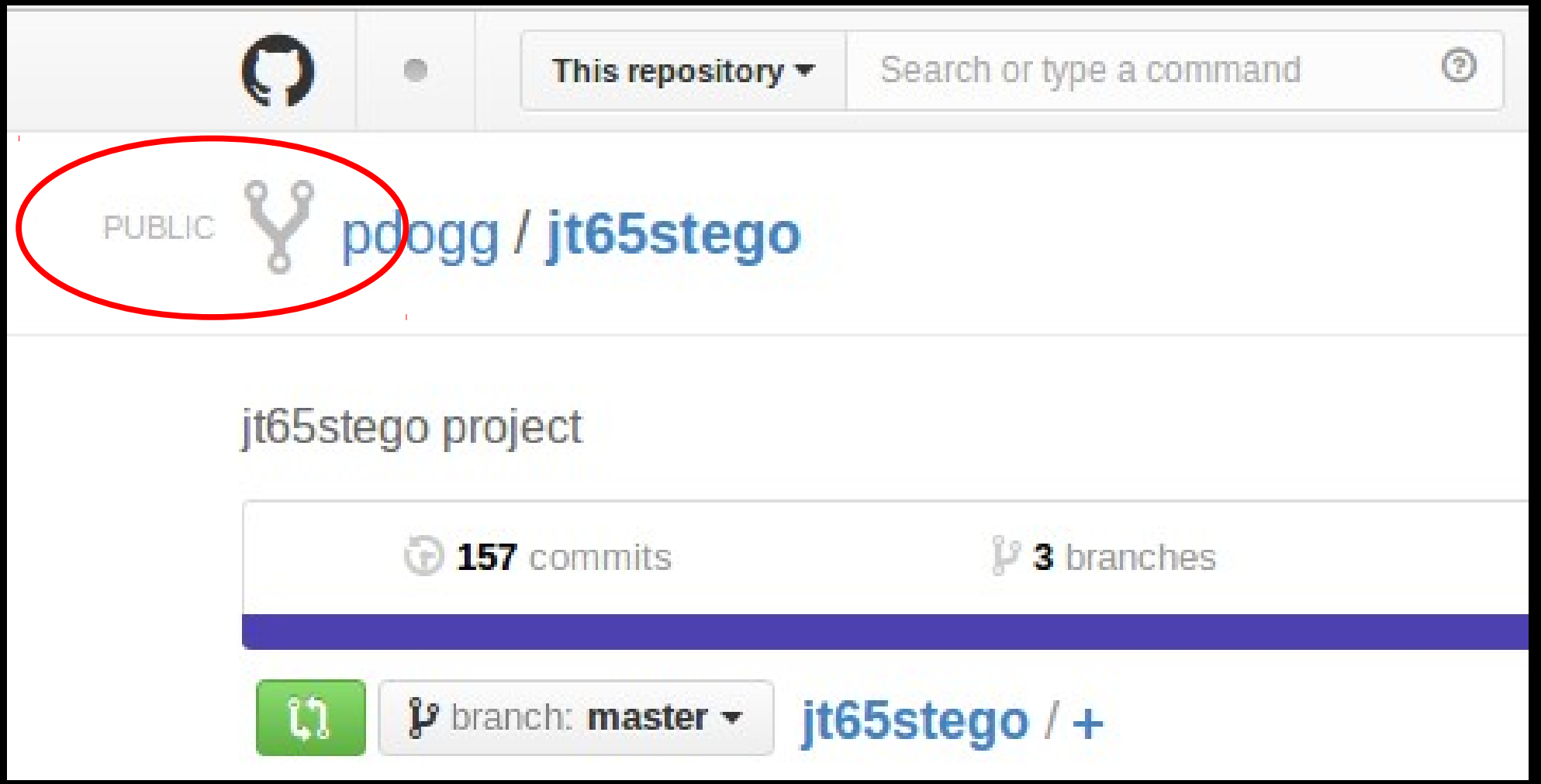


Effectiveness as a World Wide Short Message Protocol

How to get it?


This repository
Search or type a command
PRIVATE
pdogg / jt65stego
jt65stego project
157 commits
3 branches
branch: master
jt65stego / +

Available today!

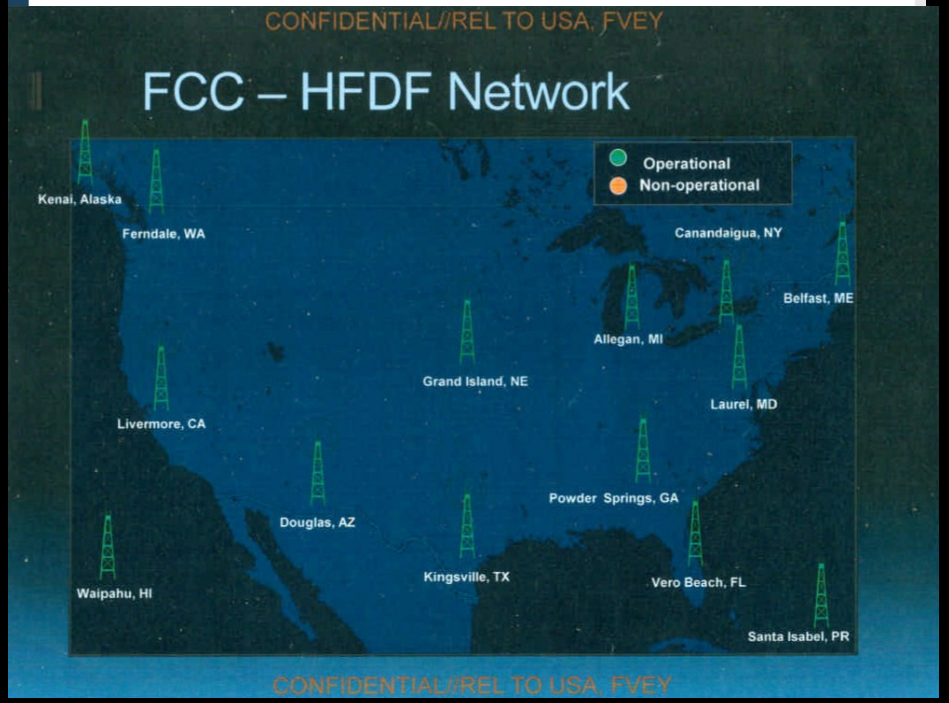


Oh yeah, it's on your conference DVD too...

“Vulnerabilities” / Known Limitations

- Analysis and Detection
  - As discussed / other methods
- Transmitter location (foxhunting)
  - Well studied problem/game by amateurs and TLAs
  - FCC/DEA/NSA - SANDKEY(1)
- Message Forgery
- Storage / long term cryptographic analysis

(1)
http://cryptomeorg.siteprotect.net/dea-nsa-sandkey.pdf



Conclusions

- Protocols and methods such as those presented can, in theory, provide a platform for short message communications with desirable properties:
  - Low infrastructure
  - Long distance
  - Covert
  - Plausibly deniable
- Potential for analysis and detection
  - Especially for well equipped adversaries

Next Steps / Further Areas of Study

- Continued Detection / Counter Detection Work
- Cryptographic Improvements
- Enhanced amateur applications
- Useful protocols and networks

Ham Exam

Crypto & Privacy Village

Sunday 12 PM – 3 PM

Get an FCC FRN!

Exam Cram Session

Wireless Village

Sunday morning - TBA

THANKS!

@pdogg77
@TheDukeZip

https://www.github.com/pdogg/jt65stego/

Special Thanks @masshackers