Gordon L. Gibby MS MD KX4Z
15216 NW 41st Avenue
Newberry, Florida 32669

Aug 8 2019

Via ECFS
Marlene H. Dortch
Federal Communications Commission
445 12th Street, SW
Washington, DC 20554

Re: **Notice of Ex Parte Submission,**
Petition for Rulemaking filed by Amateur Radio Station Licensee
Ron Kolarik (K0IDT), RM-11831

Dear Ms. Dortch:

I submit this ex parte letter with new information about the history of amateur radio, and its impact on the present issue of properly decoding WINLINK transmissions captured over the air.

THE MISSING GRASP OF AMATEUR RADIO TECHNOLOGICAL HISTORY REVEALED IN THE RECENT CRITICISM OF LONG-STANDING AMATEUR PRACTICE

I. Assertions Made About WINLINK

Beginning with a presentation to the FCC by Theodore S. Rappaport, N9NB, memorialized in an FCC Filing[1] an onslaught of criticism of legal amateur radio practice has been waged. Dr.

[1] Rappaport, https://www.fcc.gov/ecfs/filing/10925839109476

Rappaport presented a slide talk[2] to the FCC. Slide 2 noted “Current ambiguity and national security risk for ARQ signaling” and slide 21 stated, “The ARRL and FCC have ignored the encryption problem, now the hobby is unable to self-police today’s data stations or the spectrum use…..National security and criminal activity is a concern if data increases without first addressing the lack of CW ID, lack of regulatory compliance, and lack of eavesdropping that exists today.” Slide 3 boldly proclaimed,

> “Today’s data users (Pactor 2-4, Winlink) are effectively *encrypted* w/ARQ. Part 97.309(a)(4) is unclear about the practical intent. Today, data traffic cannot be monitored by Official Observers / FCC…..” [emphasis original]

These themes continued in the submissions from Dr. Rappaport, who in addition to his electrical engineering fame, is also a professor of computer science:

Date	Comment
11/06/16	“I pointed out national security concerns with the current problem of encrypted data, which arises from the non published compression algorithms used in Pactor II, Pactor III, and Pactor IV, and also discussed how the identification of many ACDS stations are often encrypted, as well, since that is an option on the SCS modems.” [3]
02/21/18	“Their advocacy of the usage of Pactor 4 and their urging for immediate approval of Docket WT16-239 NPRM (or adoption of RM-11708) ignore substantial and legitimate harmful interference concerns which they have admitted in public forums, ignore current and important violations of Part 97 rules associated with today’s wideband data usage on the HF amateur radio bands, and introduce serious national security risks associated with Docket WT 16-239 NPRM and RM-11708. “ “….where many commenters stated the precise technical arguments about interference, the numerous FCC rules violations by ACDS transmissions, the improper use of amateur radio conducted over email and internet, and the need for documented coding (e.g. avoidance of **encryption through the use of open and published compression algorithms**). “ [emphasis added] “The Commission should note that the ARRL represents only about 20% of the licensed US amateur operators, so it cannot claim to be the national political voice for the hobby. In recent years, the broad US amateur radio community has become increasingly weary of the leadership and governance at ARRL (see, for example, http://MyARRLVoice.org). In RM-11306 and RM-11708, ARRL and Mr. Waterman have continually taken a position of advocacy for machine-to-machine and automated (and encrypted) HF data communications for email and internet usage, even though this represents the interests of a very small minority (perhaps only a few thousand out of 750,000) US amateur radio operators. “[4]

[2] https://ecfsapi.fcc.gov/file/10925839109476/FCCNPRM%20Docket%2016-239%20Final.pptx

[3] Rappaport, https://ecfsapi.fcc.gov/file/1110241203910/Reply%20to%20Comments%20NPRM.docx

05/11/18	“We discussed **national security** concerns voiced in public comments that urge WT16-239 and RM-11708 not be enacted, since the proposals ignore interference and would inadvertently perpetuate radio traffic that is impossible to self-monitor or intercept, leading to a greater increase of nefarious or illegal messages using proprietary schemes that are extremely difficult or impossible to intercept using “a man in the middle” amateur radio station. We discussed interference that occurs when wideband data is allowed to operate in the same spectrum as narrowband signals, and discussed where Part 97 rules are ambiguous in prohibiting (perceived by some to be a loophole) **proprietary compression or encryption** for over-the- air signaling” [5] [emphases added]
11/15/18	“The public records in the above cited proceedings make clear how the evolution of undocumented, proprietary transmission technologies such as Pactor and Winlink, ARDOP, Winmor, STANAG, and other HF transmission schemes that use controlling software (e.g., Winlink, which was designed for secure commercial and government maritime mobile radio systems) have created a national security problem in the amateur radio service, such that 3rd parties (e.g. other ham radio operators, or the FCC listening stations, themselves) cannot intercept and decode over-the-air transmissions when used in the popular Automated Repeat Request (ARQ) mode. Thus, Winlink with Pactor, and Winlink with ARDOP, Winmor, STANAG, or other modulations, cannot be intercepted or deciphered, over-the-air, by other amateur radio operators or the general public, thereby enabling users of the amateur radio service to provide obscured, private communications.”[6]
03/20/19	“I believe that if left unchecked and ignored, this continued assault on the basic principles of the amateur radio service will degrade the fabric and purpose of the hobby, will create massive and uncontrollable interference on the HF bands, will hamper the cultivation of trained technical experts and future engineers in the US, and will threaten the national security of the United States through transmissions that cannot be intercepted by the public. “ “Intercepting the Winlink/Pactor and other ARQ transmissions simply cannot be done, yet there is no admission of these facts by FCC, ARSFI/Winlink, or ARRL. FCC management should wonder why this is, when thousands have publicly complained about this for over a decade “ [7]
04/03/19	“Also, in your last paragraph of the web posting, I note that ARRL has still not

[4] Rappaport, https://ecfsapi.fcc.gov/file/1022117362165/FCC%20PS%2017-344%20Reply%20to%20ARRL%20and%20Steve%20Waterman%20from%20N9NB.pdf In paragraph 23 of this Document, Dr. Rappaport discusses at great length facilities of the SCS modems, which are disabled by WINLINK software, but appears unaware of that fact.

[5] Rappaport, https://ecfsapi.fcc.gov/file/10511986308556/Rappaport%20Ex%20Parte%20May%2011%202018.docx

[6] Rappaport, https://ecfsapi.fcc.gov/file/1111110314487/FCC%20EX%20PARTE%2016-239%20Eric%20Burger%20Nov%2011%202018.pdf

[7] Rappaport, https://ecfsapi.fcc.gov/file/1032167020169/FCC%20Letter%20RM%2011828.pdf

	acknowledged the problems of the "obscured" data in pending NPRM 16-239, or in ARRL's proposed 2.8 KHz bandwidth limit through its RM-11708 proposal. I ask again, has anyone at ARRL attempted to eavesdrop another Winlink ARSFI ARQ data protocol transmission (such as Pactor 2, Pactor 3, Winmor or ARDOP) intended for another station by using the W1AW Winlink account in the ham radio bands? This would inform the board of ARRL whether or not this email traffic can or cannot be intercepted over the air, and would inform the ARRL board if these transmissions are obscured for meaning to others, in violation of part 97 rules and which harm public participation of the hobby and which jeopardizes our national security as no one else can eavesdrop the on-air transmissions “[8]

Articles were also published in multiple national trade journals discussing obscured amateur radio communications with a national security risk (below is just a sample):

11/21/18	“If allowed, NPRM 16-239 would perpetuate the current violations and would authorize obscured transmissions of unlimited bandwidth over the global airwaves, further increasing the danger to our national security, since these transmissions cannot be intercepted or eavesdropped by other amateur radio operators or the FCC,” Rappaport’s filing said. ... The filing said public records clearly show how the evolution of undocumented, proprietary transmission technologies such as PACTOR and Winlink, ARDOP, Winmor, STANAG and other HF transmission schemes that use controlling software have created a national security problem in the amateur radio service. Third parties, including other ham radio operators or the FCC listening stations, cannot intercept and decode over-the-air transmissions when used in the popular automated repeat request (ARQ) mode.[9]
04/15/19	“To rectify this ongoing problem of effective encryption in amateur radio, and to open up the airwaves so that computer enthusiasts may intercept and experiment and learn from all transmissions, the FCC recently published a rule making proposal RM-11831, that would reiterate the need to keep all data communications open for all to intercept, while keeping email relay stations in their own allocated many sub-bands. Many who are improperly using HF radio for free private email are spreading false information about the proposal and its impacts. The proposal would not

[8] Rappaport, https://ecfsapi.fcc.gov/file/1040322516387/FCC%20Letter%20RM%2011831%20final.pdf

[9] Wendelken, S. Rappaport Suggests National Security Risks with Amateur Radio Violations, Radio Resource International, accessed at: https://www.rrmediagroup.com/News/NewsDetails/NewsID/17667

	end emailing in amateur radio, it would just open up the messages so all can hear and intercept.“ [10]

The Petitioner (Ron Kolarik) has also weighed in:

10/09/18	“ensure all transmissions remain open for over-the-air eavesdropping of station identification, message content, and capable of being fully decoded with publicly available methods as required by Part 97.113(a)(4) “[11]
07/18/19	Monitoring by Third Parties is Not Only Essential... “It’s the Law!!!” • Obscured Traffic Has Been a “Core Issue!” [12]

2. The History of Relevant Amateur Radio Technology Development

These claims of encryption, damage to the amateur hobby, obscured transmissions, and risk to the national security of millions of Americans would come as quite a surprise to those who pioneered these techniques, including

- Terry L. Fox, WB4JFI, who published version 2.0 of the AX.25 protocol in **1984**.   (AX.25 includes both unconnected and “connected” modes--- the latter using ARQ.)

- Eric Scace, K3NA, who published “a major effort towards updating version 2.0….in the 7th Computer networking Conference” in **1988**.

- William A. Beech NJ4P, Douglas E. Nielsen, N7LEM, and Jack Taylor, N7OO, who published Version 2.2 of the AX.25 Link Access Protocol for Amateur Packet Radio in July **1993**.[13]

But likely most surprised would be Jean-Paul Roubelat F6FBB, who built upon the previous file-transfer systems of the YAPP protocol and created the FBB bulletin board system, beginning in 1986[14]

[10] Rappaport, T. Professor Rappaport Urges Support of the Basic Tenets of Amateur Radio, Microwave Journal, accessed at;   https://www.microwavejournal.com/articles/32108-professor-rappaport-urges-support-of-the-basic-tenets-of-amateur-radio

[11] Kolarik, https://ecfsapi.fcc.gov/file/100918881206/PETITION%20FOR%20RULEMAKING.pdf

[12] Carson, Kolarik, McVey, White, https://ecfsapi.fcc.gov/file/1071958608259/July%2018%2C%202019%20Ex%20Parte%20Filing.pdf

[13] Beech W, et al., AX.25 Link Access Protocol for Amateur Packet Radio, Version 2.2, Revision July 1998 (c) 1997 by Tucson Amateur Packet Radio Corporation, portions (c) 1984, 1993 by the American Radio Relay League, Inc. Accessed at: https://www.tapr.org/pdf/AX25.2.2.pdf

[14] FBB (F6FBB) https://en.wikipedia.org/wiki/FBB_(F6FBB)

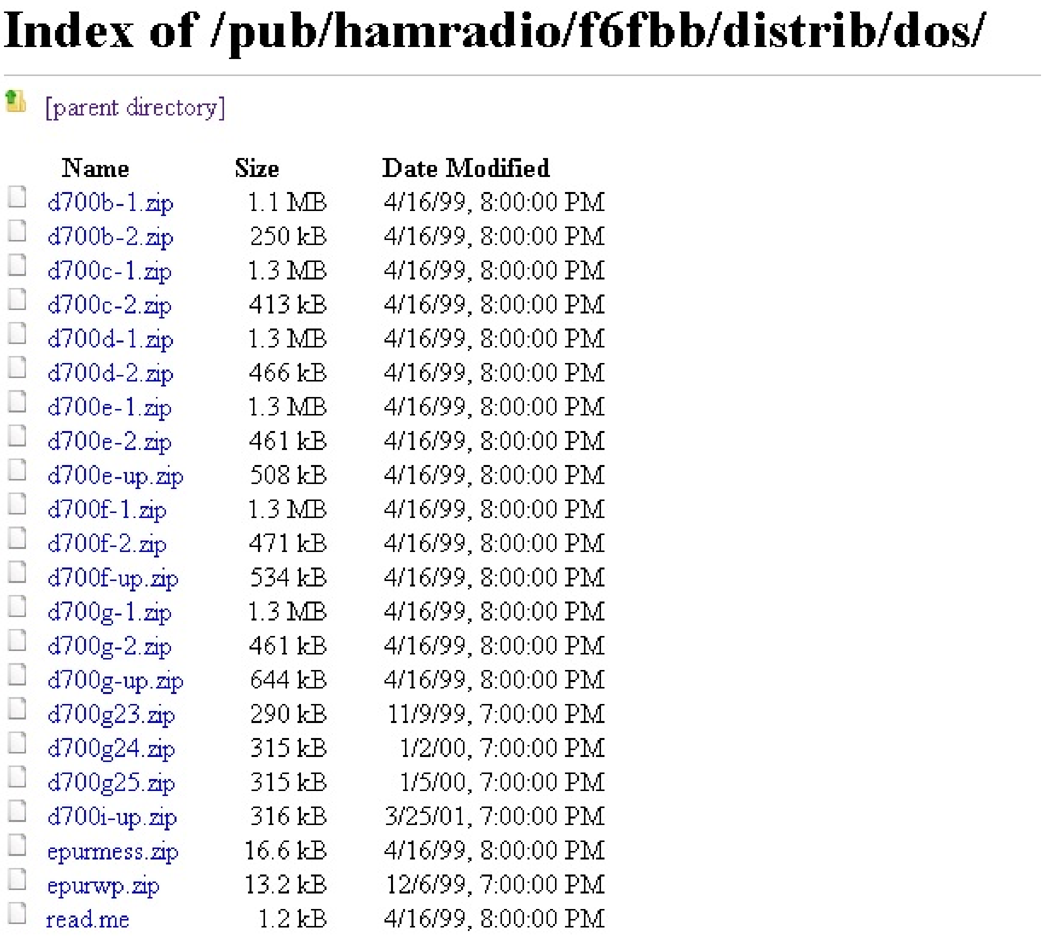
and continuing forward to develop what will be shown to be the basic file transfer systems so maligned by Dr. Rappaport….by **1999**.   Before the turn of the century.

This paper will attempt to demonstrate that the current file transfer systems accused of being so undecipherable and such a risk to national security…..are the *same basic systems* developed by Jean-Paul Roubelat by 1999, used for decades all over the world, and for which open source decoding software, and detailed explanations of operation have existed for *at least 20 years*, beginning with the MS-DOS operating system.   And that knowledge of precisely those systems allowed perfect decoding of over-the-air captures….just as it does tens of thousands of times for users of the current implementations of these technological advances.

There was apparently no significant national security concern about these techniques prior to 2016. Peter Helfert has indicated that in his experience, he was not aware of *anyone* complaining of the compression systems utilized in Roubelat's software.[15]   On the QRZ forum, I posted asking if anyone was aware of any complaints about LZHUF compression in packet bulletin board systems.[16]   No responses so far.

Roubelat's software can still be downloaded for inspection and usage from an extant ftp server, as for example, this listing of code for the DOS operating system of 1999:

[15] Helfert: https://forums.qrz.com/index.php?threads/huggins-did-it.667817/page-40#post-5154550
[16] Gibby: https://forums.qrz.com/index.php?threads/huggins-did-it.667817/page-41#post-5154595



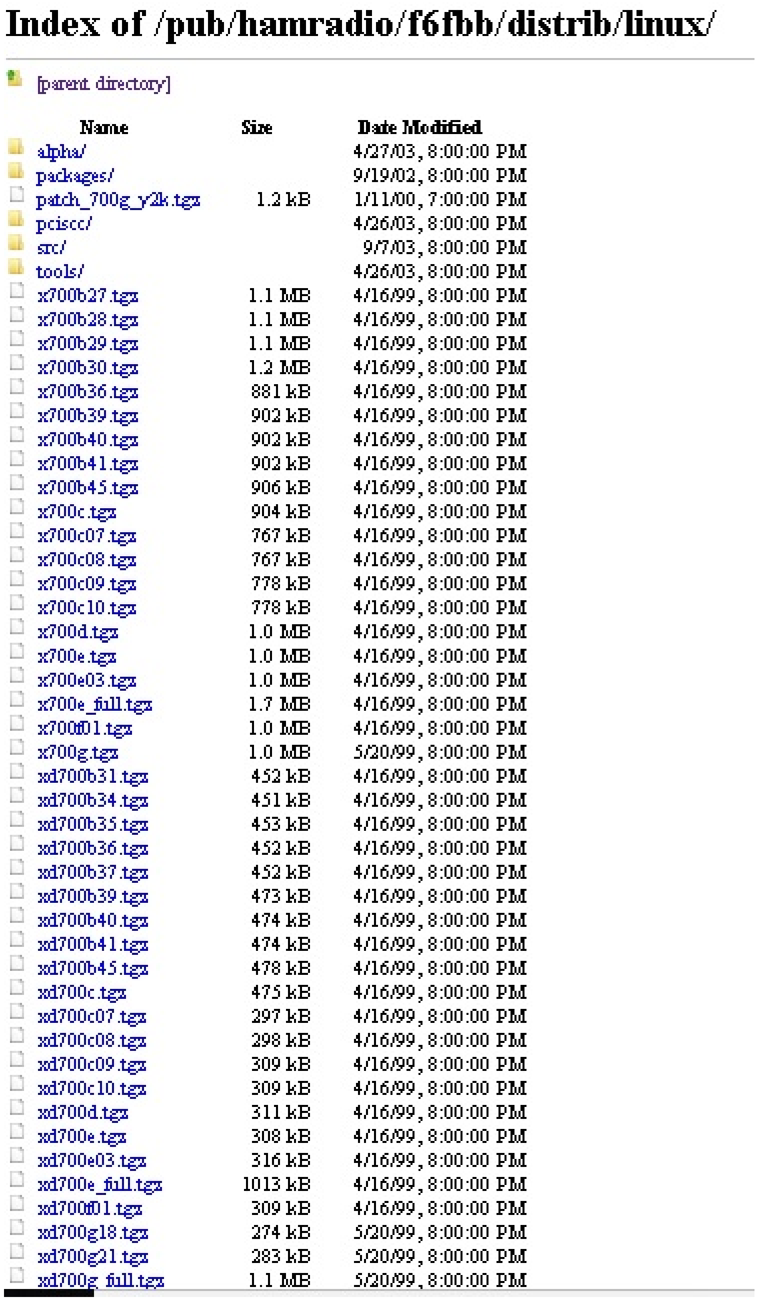
Index of /pub/hamradio/f6fbb/distrib/dos/

[parent directory]

Name	Size	Date Modified
d700b-1.zip	1.1 MB	4/16/99, 8:00:00 PM
d700b-2.zip	250 kB	4/16/99, 8:00:00 PM
d700c-1.zip	1.3 MB	4/16/99, 8:00:00 PM
d700c-2.zip	413 kB	4/16/99, 8:00:00 PM
d700d-1.zip	1.3 MB	4/16/99, 8:00:00 PM
d700d-2.zip	466 kB	4/16/99, 8:00:00 PM
d700e-1.zip	1.3 MB	4/16/99, 8:00:00 PM
d700e-2.zip	461 kB	4/16/99, 8:00:00 PM
d700e-up.zip	508 kB	4/16/99, 8:00:00 PM
d700f-1.zip	1.3 MB	4/16/99, 8:00:00 PM
d700f-2.zip	471 kB	4/16/99, 8:00:00 PM
d700f-up.zip	534 kB	4/16/99, 8:00:00 PM
d700g-1.zip	1.3 MB	4/16/99, 8:00:00 PM
d700g-2.zip	461 kB	4/16/99, 8:00:00 PM
d700g-up.zip	644 kB	4/16/99, 8:00:00 PM
d700g23.zip	290 kB	11/9/99, 7:00:00 PM
d700g24.zip	315 kB	1/2/00, 7:00:00 PM
d700g25.zip	315 kB	1/5/00, 7:00:00 PM
d700i-up.zip	316 kB	3/25/01, 7:00:00 PM
epurmess.zip	16.6 kB	4/16/99, 8:00:00 PM
epurwp.zip	13.2 kB	12/6/99, 7:00:00 PM
read.me	1.2 kB	4/16/99, 8:00:00 PM

Figure 1. Software still available from the 1999 FBB development, which uses the same LZHUF compression as WINLINK.

There is also source code for the linux operating system:



Index of /pub/hamradio/f6fbb/distrib/linux/

[parent directory]

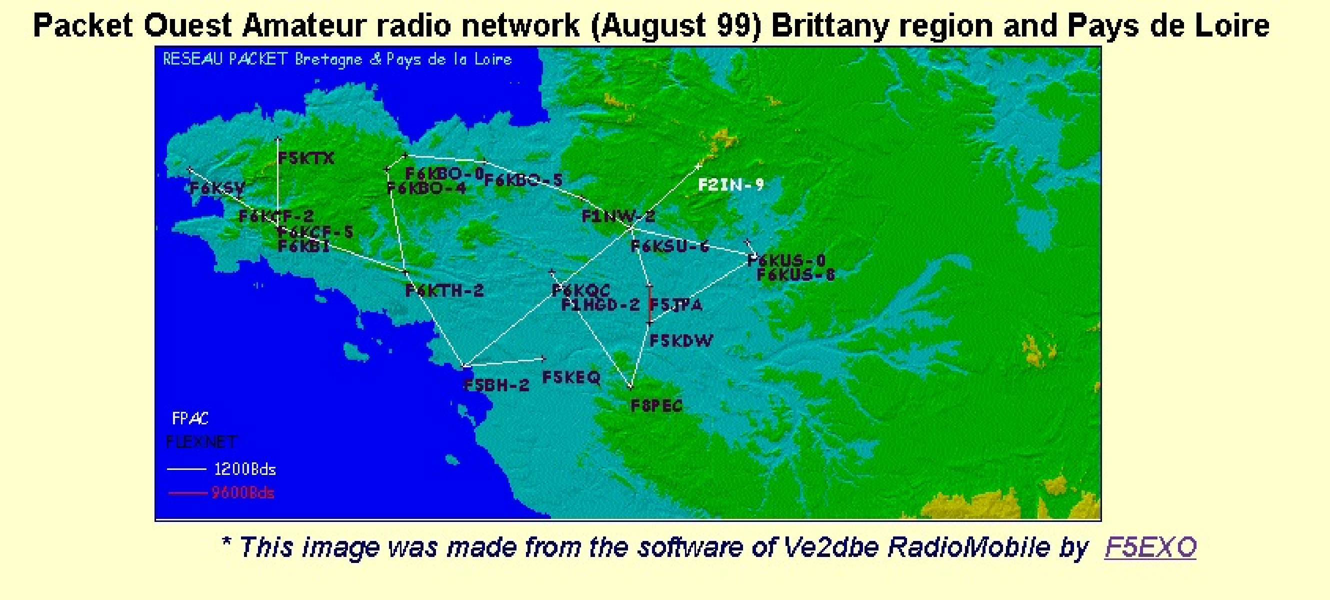
Name	Size	Date Modified
alpha/		4/27/03, 8:00:00 PM
packages/		9/19/02, 8:00:00 PM
patch_700g_y2k.tgz	1.2 kB	1/11/00, 7:00:00 PM
pciscc/		4/26/03, 8:00:00 PM
src/		9/7/03, 8:00:00 PM
tools/		4/26/03, 8:00:00 PM
x700b27.tgz	1.1 MB	4/16/99, 8:00:00 PM
x700b28.tgz	1.1 MB	4/16/99, 8:00:00 PM
x700b29.tgz	1.1 MB	4/16/99, 8:00:00 PM
x700b30.tgz	1.2 MB	4/16/99, 8:00:00 PM
x700b36.tgz	881 kB	4/16/99, 8:00:00 PM
x700b39.tgz	902 kB	4/16/99, 8:00:00 PM
x700b40.tgz	902 kB	4/16/99, 8:00:00 PM
x700b41.tgz	902 kB	4/16/99, 8:00:00 PM
x700b45.tgz	906 kB	4/16/99, 8:00:00 PM
x700c.tgz	904 kB	4/16/99, 8:00:00 PM
x700c07.tgz	767 kB	4/16/99, 8:00:00 PM
x700c08.tgz	767 kB	4/16/99, 8:00:00 PM
x700c09.tgz	778 kB	4/16/99, 8:00:00 PM
x700c10.tgz	778 kB	4/16/99, 8:00:00 PM
x700d.tgz	1.0 MB	4/16/99, 8:00:00 PM
x700e.tgz	1.0 MB	4/16/99, 8:00:00 PM
x700e03.tgz	1.0 MB	4/16/99, 8:00:00 PM
x700e_full.tgz	1.7 MB	4/16/99, 8:00:00 PM
x700f01.tgz	1.0 MB	4/16/99, 8:00:00 PM
x700g.tgz	1.0 MB	5/20/99, 8:00:00 PM
xd700b31.tgz	452 kB	4/16/99, 8:00:00 PM
xd700b34.tgz	451 kB	4/16/99, 8:00:00 PM
xd700b35.tgz	453 kB	4/16/99, 8:00:00 PM
xd700b36.tgz	452 kB	4/16/99, 8:00:00 PM
xd700b37.tgz	452 kB	4/16/99, 8:00:00 PM
xd700b39.tgz	473 kB	4/16/99, 8:00:00 PM
xd700b40.tgz	474 kB	4/16/99, 8:00:00 PM
xd700b41.tgz	474 kB	4/16/99, 8:00:00 PM
xd700b45.tgz	478 kB	4/16/99, 8:00:00 PM
xd700c.tgz	475 kB	4/16/99, 8:00:00 PM
xd700c07.tgz	297 kB	4/16/99, 8:00:00 PM
xd700c08.tgz	298 kB	4/16/99, 8:00:00 PM
xd700c09.tgz	309 kB	4/16/99, 8:00:00 PM
xd700c10.tgz	309 kB	4/16/99, 8:00:00 PM
xd700d.tgz	311 kB	4/16/99, 8:00:00 PM
xd700e.tgz	308 kB	4/16/99, 8:00:00 PM
xd700e03.tgz	316 kB	4/16/99, 8:00:00 PM
xd700e_full.tgz	1013 kB	4/16/99, 8:00:00 PM
xd700f01.tgz	309 kB	4/16/99, 8:00:00 PM
xd700g18.tgz	274 kB	5/20/99, 8:00:00 PM
xd700g21.tgz	283 kB	5/20/99, 8:00:00 PM
xd700g_full.tgz	1.1 MB	5/20/99, 8:00:00 PM

Figure 2;   FBB software for Linux distributions.   Same LZHUF compression as WINLINK system.

Note that the majority of the dates on this freely available information are **April 16, 1999**.

The FBB bulletin board system was a quite popular amateur radio BBS system, with world wide usage.   This was a portion of the explosion of amateur radio packet communications growth, the amateur radio version of e-mail and texting, long before the creation of cell- and smartphones.   Far from damaging amateur radio, these systems were party of a dynamic, enthusiastic and learning amateur radio community.

Node maps for portions of France (where FBB was developed) can be studied here: http://www.f6fbb.org/net/net_.htm and one such map is reproduced here[17]:



Packet Ouest Amateur radio network (August 99) Brittany region and Pays de Loire

* This image was made from the software of Ve2dbe RadioMobile by F5EXO

Figure 3: FBB systems in a portion of France in 1999

By 1999[18], Jean-Paul had developed a very sophisticated system for transferring files of both ASCII and binary from his FBB amateur radio bulletin board system. The basic details remain on their web page today, and are also contained in documents that can be downloaded from the directories shown in figures above and read in English.

FBB FILE TRANSFER PROTOCOLS - 1999

Jean-Paul's creation of the System Identifier ("SID") strings explains some of the otherwise "cryptic" character strings that show up, even today, as WINLINK client-server or peer-to-peer systems are

[17] http://www.f6fbb.org/net/fbre.htm , translation by Google from the French.

[18] This may have happened much before 1999; I was not involved in those years, and cannot find readily available web servers covering periods before that.

preparing for a transfer.[19] The **SID string** allows each side of a transfer to know what capabilities exist in the counterparty, and are always sent as the first line after connection is established.

Either two or three fields are separated by hyphens ("-"). For the FBB system, an example is [FBB-7.00-AB1FHMX$] and the breakdown is::

- Author String (FBB for Jean-Paul's system, WL2K for Winlink server, RMS Express or similar for client WINLINK system)

- Version Number or Author Data ("7.00" for the 1999 FBB system, 5.0 for current WINLINK server)

- Feature List, comprised of letters, with an optional appended version number. (I received B2FWIHJM$ on a recent WINLINK connection)

and obviously this same system is in use in 2019 with the WINLINK system, for which I received WL2K-5.0-B2FWIHJM$ on a recent connections

The letters of the Feature list are explained as follows in the 1999 edition:

A: acknowledge for personal messages
B: FBB compressed protocol V0 is supported
B1: FBB compressed protocol V1 is supported
F: FBB basic protocol supported
H: Hierarchical location designators supported
M: Message identifier supported
X: Compressed batch forwarding supported
$: BID supported (must be last character of the list)

To which the WINLINK group has furthered the B1 protocol with the B2 protocol which follows B1 rules

[19] An example of a full WINLINK transfer of a message (over an amateur radio microwave frequency):

```
WL2K-5.0-B2FWIHJM$]
;PQ: 48339189
CMS>
  ;FW: KX4Z FLBDR-OPER|42164657 KXFOURZ-EM|50455978
  [RMS Express-1.5.22.0-B2FHM$]
  ;PR: 35559043
  ; WL2K DE KX4Z (EL89RQ)
  FC EM 9E8VIDHGRANY 1076 624 0
  F> 3C
FS Y
*** Sending 9E8VIDHGRANY.
FF
*** Completed send of message 9E8VIDHGRANY
*** Sent 1 message.  Bytes: 677,  Time: 00:01,  bytes/minute: 36027
  FQ
```

The communication of this SID allows the two parties in communication to choose protocols for transfers that both possess.

The 1999 FBB system and current web site explain three forwarding protocols, each of which is backwards compatible with previous versions:

1. ASCII basic protocol [SID character: F]
2. Binary compressed protocol version 0 [SID character: B]
3. Binary compressed protocol version 1.

The FBB system indicates that "source code is available" – the link[20] leading toward the lzhuf compression algorithm, precisely what is still used today, 20 years later, by the WINLINK system throughout.

1. ASCII Basic Protocol (SID character "F")

This protocol allows for the automated passage of up to five ASCII messages. The http://www.f6fbb.org/ web site does not discuss the "first" of two forward protocol, other than to state "it is the standard MBL/RLI protocol." The site does discuss the second ASCII basic protocol: A proposed transfer is set up by a FB proposal, such as

FB P F6FBB FC1GHV FC1mvP 24657_F6FBB 1345

which is explained on the FBB web site as follows:

FB	Identifies type of proposal
P	Type of message: P= private, B= bulletin (ie. for one amateur or for the entire bulletin board)
F6FBB	Sender field
FC1GHV	BBS of the recipient
FC1MVP	Recipient
24657_F6FBB	"BID ou MID"
1345	Size of message in bytes

The F6FBB web site includes the back-and-forth characters used between the two computerized stations to begin the transfer if agreed to, and how to recognize the end of all remaining messages ("FF") and the response of the counterparty if they have not other messages to send either, "FQ" followed by a disconnect.

[20] Link: ftp://ftp.f6fbb.org/pub/f6fbb/utils/lzhuf_1.zip This link however does not work; I was able to obtain it from one of the still extant mirror sites of the f6fbb site. The code is as one would expect for the lzhuf compression/decompression algorithm, and was a portion of what tipped Mr. Huggins and myself to the necessary steps to obtain decompression.

2. Binary Compressed Forwarding Version 0

The FBB web site describes this as an extension to the ASCII system, indicated by a “B” in the SID, and as an extension, still requiring the letter “F” as part of the SID. This system allows two possible commands:

FA proposes a compressed ASCII message (already implemented)
FB proposes a binary compressed file (not yet implemented according to the web site)

Despite that confusing explanation, the web site continues on to explain that the compressed ASCII message will be transferred in BINARY FORM, using a format derived by the YAPP protocol, in which each message includes a header, blocks of data, and an end-of-message marker and checksum.

Proposal Example: FB P FC1CDC F6ABJ F6aXV 24754_F6FBB 345

As one can see, this is very similar to the proposal for the previously described uncompressed ASCII transfer.

Format of the ASCII compressed message: (type FA)
<SOH> (start of header) 1 byte = 01 Hex
Length of header 1 byte = Length of title and offset, including the two NUL characters
<NUL> 1 byte = 00 hexadecimal
Offset 1 to 6 ASCII bytes[21]
<NUL> 1 byte = 00 hexadecimal

Although the FBB page indicates the binary format had not yet been implemented [likely because the B1 version obviated the need] , the format of the header for an FB transfer was already specified:

<SOH> (start of header) 1 byte = 01 Hex
Length of the header 1 byte = length of filename and offset including the 2 NUL characters
Name of the file (1 to 80 ASCII bytes)
<NUL> 1 byte = 00 hexadecimal
Offset 1 to 6 ASCII bytes
<NUL> 1 byte = 00 hexadecimal

The FBB website includes the information:

> “French regulations require that the title of the message or the file name are transmitted in readable ASCII and are not compressed”

and to this day, the WINLINK system does precisely that as has been observed multiple times.

[21] The “offset” appears throughout to refer to how to position a fragment of a file, should the file be transmitted as multiple, separate fragments.

The web site further gives the specification for the Data Block format, of 1 to 256 bytes:

<STX> (start of data block)   1 byte = 02 hexadecimal
Size of data 1 byte = 00 to ff hex; 00 indicating 256 bytes
Data bytes, 1 to 256 bytes

The final data block is followed by the end of file specifier and the checksum:

<EOT> 1 byte = 04 hexadecimal
Checksum: 1 byte = 00 to ff hexadecimal

The FBB page indicates that the majority of the protocol comes from the YAPP protocol and thanks WA7MBL

3. Binary Compressed Forwarding Version 1

The availability of this protocol is indicated by the **B1** symbols in the SID. It is an extension of the Version 0 protocol and allows for the transmission of compressed ASCII or binary files.   The 7 fields of the previous version's proposal are required, but additional fields may be added.

The FS line specifying the response of the system being asked to accept a file includes:

+ or Y: Yes, message accepted
- or N: No, message has already been received
= or L: Later, already receiving this message
H: Message is accepted but will be held
R: Message is rejected
E: There is an error in the line
Ioffset or Aoffset: Yes, message accepted from (offset)

An example of submission of an ASCII message is:

FA P FC1CDC F6ABJ F6AXV 24754_F6FBB 345

while an example of a submission of a binary file might be

FB P FC1CDC F6ABJ 24754_F6FBB 345

Messages are transferred with headers

- FA (ASCII): Same as for Version 0
- FB (Binary)   Same as for Version 0

The data blocks are sent somewhat differently from the Version 0:

<STX>   1 byte = 02 Hex
Size of data, 1 byte = 00 to ff hex (00 indicates 256 bytes, the maximum)
Date bytes 1 to 256 bytes

FIRST transmitted block of data must contain a header with:
CRC16 of the full binary file) 2 bytes
Size of the full uncompressed file (4 bytes)
And this data must be in little-endian Intel format (less significant end first)
The LAST data block is followed by the end of file specifier (04 Hex) and the checksum of the data sentence

Checksum: equal to the sum of all the data bytes of the transmitted data, modulo 256 (8 bits) and then two's complemented
Checking then involves the sum of the data bytes fro the fie and the received checksum modulo 256 shall be equal to 0

<u>Compression From 1999:</u>   The FBB page notes that the LZHUF_1 program, used with option e1, generates a binary compressed file with:

2 bytes CRC16
Length 4 bytes
Data – filling the remainder of the file

For the Version 0 forwarding, on the portion from offset 2 will be sent
For the Version 1 forwarding, the 6 "top bytes" will always be sent "then if resume seek to asked offset, then send data"

Having studied the above information, the "B2F" protocol of the WINLINK system now is much easier to understand and this explains the SID string used by current WINLINK stations: B2 indicating the B2 protocol, and the F indicating that this is an extension from the original FBB ASCII protocol.

The WINLINK B2 protocol includes a new FC proposal (likely because the transfer no longer depends on linked bulletin board systems to deliver messages):

FC <type> <control message> EM   <Unique ID> UncompressedSize CompressedSize

An example:

FC EM 9E8VIDHGRANY 1076 624 0

The message handling for the FC compressed binary is the same as for the FB

The WINLINK web page describing the B2F protocol includes this explanation that there is no other compression other than LZHUF:

> As explained in the introduction, B2F and it's payload compression mechanism is always implemented on top of a transport (radio or wireline) protocol, such as Pactor 1-4, ARDOP, VARA, AX.25, Telnet over TCP/IP, etc. In the specific case of using SCS Pactor modems to handle the radio protocol, Winlink programs connect to the hardware using W8DED extended host mode and initialize the hardware using the hardware's "MODE 0" command, which puts the hardware into strict ASCII mode without any native compression. This is the case for other radio modes as well, if their modems implement native compression or manipulation that would obscure the meaning of content payload, or hamper on-air monitoring with the correct tools. [22]

Multiple Groups Have Successfully Utilized This Information

The complaint that this system is somehow impossible to decipher is ludicrous. **We are not at all the first group to recreate the ability to read WINLINK compressed messages**. Obviously [1] John Wiseman has had *open source* connected WINLINK transfers working for many, many years.[23] [2] Paclink-Unix created a Unix version of a WINLINK client[24] but development may have stalled after 2013.

[3] PAT DEVELOPMENT – New Protocol!

Pat is an open-source GO-based multi-platform *open source* connected WINLINK client software.[25] [26] It operates with WINMOR, ARDOP, PACKET[27] and has a beta-version for PACTOR.[28]

The GO WINLINK development *wrote their own lzhuf routine*[29], using paclink-unix source as a reference, and even added a new B2 protocol, an FD protocol. (Availability of that protocol announced by a G in the FBB SID). This group also chose to compress---but just as the WINLINK developers, they used an end-to-end compression, employing gzip instead of lzhuf in this new protocol. Those questioning the choices of WINLINK developers might note that this group also chose full-file compression. Gzip dates from 1992 and has its basis in LZ compression.[30] *This is an example of amateur radio experimental advances due to the WINLINK system*.[31]

One reason for re-stating all this information in a formal FCC Filing is to guarantee that it remains available to anyone needing to re-code as we had to.

[22] https://winlink.org/B2F
[23] Wiseman: http://www.cantab.net/users/john.wiseman/Documents/LinBPQ_RMSGateway.html
[24] http://paclink-unix.sourceforge.net/
[25] http://getpat.io/
[26] PAT source code available at: https://github.com/la5nta/pat
[27] https://github.com/la5nta/wl2k-go
[28] https://github.com/harenber/ptc-go/blob/master/README.md
[29] https://godoc.org/github.com/la5nta/wl2k-go/lzhuf
[30] Wikipedia: https://en.wikipedia.org/wiki/Gzip
[31] https://github.com/la5nta/wl2k-go#gzip-experiment

Despite the public existence of all these systems clearly able to decode these messages, the proponents of RM-11831 argued that a non-connected (3rd party) monitor would be unable to decode these transmissions of binary compressed messages. What was quite surprising was that despite many frequent statements (documented above) over a period *years* that this was 'a matter of national security," I was unaware of any publications of even exploratory experiments to begin to create such a monitoring system. .

3. Two Amateur Radio Operators Begin Experiments

- On April 9, 2019 I published my very simple experiment demonstrating a special case in which a 3rd party station could decode a WINLINK message, with a full explanation of the limitation of the method used, which was primarily to demonstrate there is no encryption.[32]

- John Huggins KX4O independently began his efforts to capture WINLINK packets and decode them, approximately on April 10, 2019.[33]

- **On July 30, 2019, John Huggins presented a partially successful decode of one winlink message received over the air**, with an impaired decode of the remainder, suggesting decompression difficulties. The method utilized was a brute-force experimental manual handling of the received packets and the lzhuf.c source software compiled into a decompression/compression executable.[34] This success was accomplished in approximately 16 weeks, and was a very impressive achievement!

But why the failure to completely decompress? And how to find the correct start-point for decompression (trial and error having been utilized by Huggins)? As it turns out, the missing information is *all in the historical record of the decades-old technological advances* of amateur radio by pioneers.

I thus subsequently discovered missing clues in the material presented above from the FBB web site and the WINLINK web site, to understand the packet structure. Very early on Monday August 5, I published a file containing the image of the marked packets in the data Huggins had downloaded (see Figure 4 below). [35] I then subsequently published a fuller explanation of my findings and suggestions of how to further the decoding on the QRZ web site. [36]

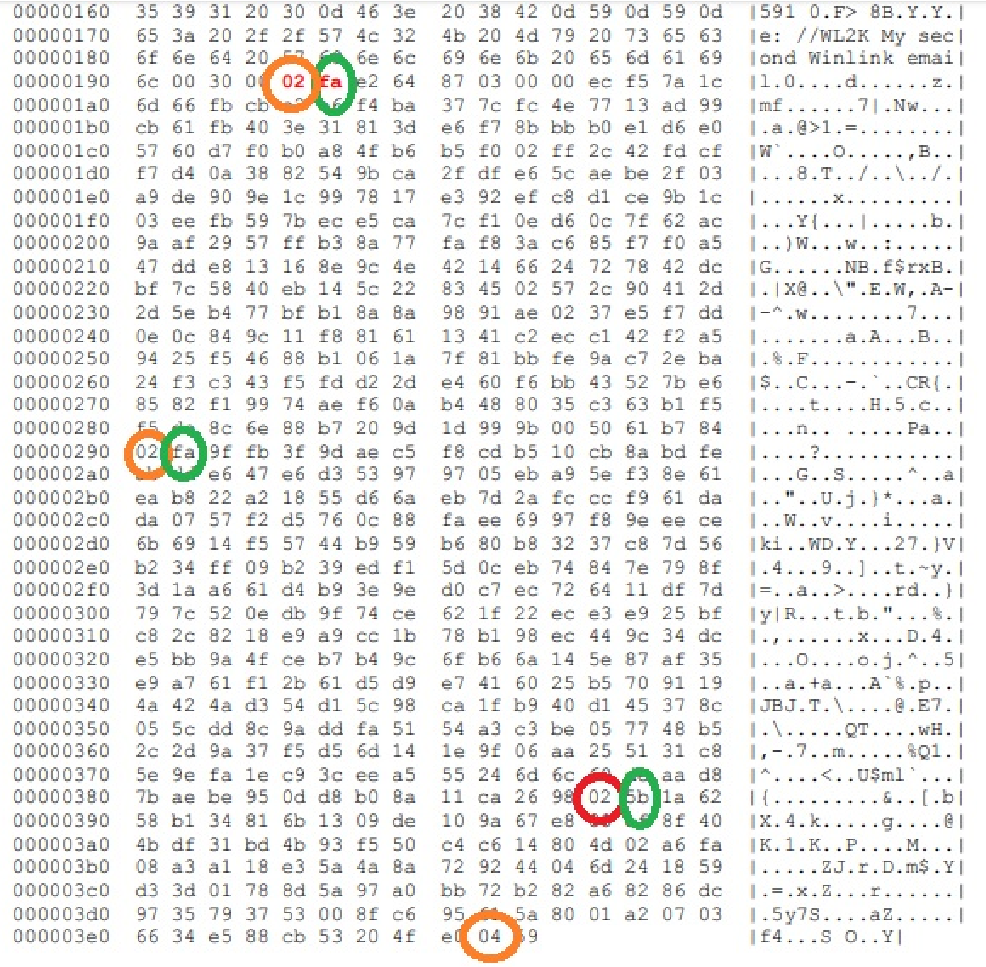
[32] Gibby: https://ecfsapi.fcc.gov/file/10410170249078/FCCRM11831-4.pdf

[33] Huggins: https://forums.qrz.com/index.php?threads/new-digital-petition-at-the-fcc-rm-11831.652589/page-86#post-5026997

[34] Huggins: https://ecfsapi.fcc.gov/file/1073182572879/KX4O_Demonstration_OTA_Winlink_Decoding.pdf

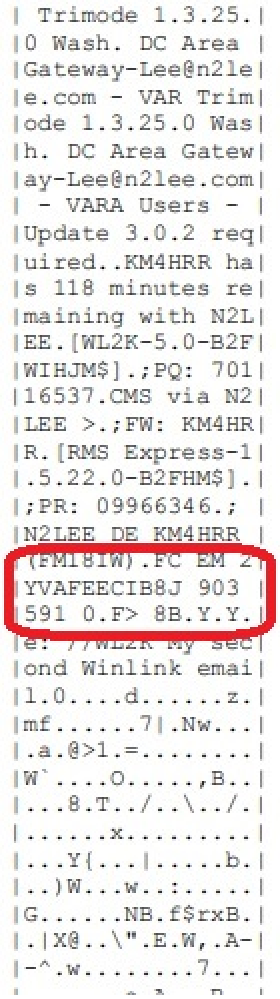
[35] Gibby, August 5, 2019: https://forums.qrz.com/index.php?threads/huggins-did-it.667817/page-26#post-5152104

[36] Gibby: https://forums.qrz.com/index.php?threads/huggins-did-it.667817/page-34#post-5153432



```
00000160  35 39 31 20 30 0d 46 3e  20 38 42 0d 59 0d 59 0d  |591 0.F> 8B.Y.Y.|
00000170  65 3a 20 2f 2f 57 4c 32  4b 20 4d 79 20 73 65 63  |e: //WL2K My sec|
00000180  6f 6e 64 20 57 69 6e 6c  69 6e 6b 20 65 6d 61 69  |ond Winlink emai|
00000190  6c 00 30 00 02 fa e2 64  87 03 00 00 ec f5 7a 1c  |l.0....d......z.|
000001a0  6d 66 fb cb [illegible] [illegible] f4 ba  37 7c fc 4e 77 13 ad 99  |mf......7|.Nw...|
000001b0  cb 61 fb 40 3e 31 81 3d  e6 f7 8b bb b0 e1 d6 e0  |.a.@>1.=........|
000001c0  57 60 d7 f0 b0 a8 4f b6  b5 f0 02 ff 2c 42 fd cf  |W`....O.....,B..|
000001d0  f7 d4 0a 38 82 54 9b ca  2f df e6 5c ae be 2f 03  |...8.T../..\../.|
000001e0  a9 de 90 9e 1c 99 78 17  e3 92 ef c8 d1 ce 9b 1c  |......x.........|
000001f0  03 ee fb 59 7b ec e5 ca  7c f1 0e d6 0c 7f 62 ac  |...Y{...|.....b.|
00000200  9a af 29 57 ff b3 8a 77  fa f8 3a c6 85 f7 f0 a5  |..)W...w..:.....|
00000210  47 dd e8 13 16 8e 9c 4e  42 14 66 24 72 78 42 dc  |G......NB.f$rxB.|
00000220  bf 7c 58 40 eb 14 5c 22  83 45 02 57 2c 90 41 2d  |.|X@..\".E.W,.A-|
00000230  2d 5e b4 77 bf b1 8a 8a  98 91 ae 02 37 e5 f7 dd  |-^.w........7...|
00000240  0e 0c 84 9c 11 f8 81 61  13 41 c2 ec c1 42 f2 a5  |.......a.A...B..|
00000250  94 25 f5 46 88 b1 06 1a  7f 81 bb fe 9a c7 2e ba  |.%.F............|
00000260  24 f3 c3 43 f5 fd d2 2d  e4 60 f6 bb 43 52 7b e6  |$..C...-.`..CR{.|
00000270  85 82 f1 99 74 ae f6 0a  b4 48 80 35 c3 63 b1 f5  |....t....H.5.c..|
00000280  f5 [illegible] 8c 6e 88 b7 20 9d  1d 99 9b 00 50 61 b7 84  |...n.. .....Pa..|
00000290  02 fa 9f fb 3f 9d ae c5  f8 cd b5 10 cb 8a bd fe  |....?...........|
000002a0  [illegible] [illegible] e6 47 e6 d3 53 97  97 05 eb a9 5e f3 8e 61  |...G..S.....^..a|
000002b0  ea b8 22 a2 18 55 d6 6a  eb 7d 2a fc cc f9 61 da  |.."..U.j.}*...a.|
000002c0  da 07 57 f2 d5 76 0c 88  fa ee 69 97 f8 9e ee ce  |..W..v....i.....|
000002d0  6b 69 14 f5 57 44 b9 59  b6 80 b8 32 37 c8 7d 56  |ki..WD.Y...27.}V|
000002e0  b2 34 ff 09 b2 39 ed f1  5d 0c eb 74 84 7e 79 8f  |.4...9..]..t.~y.|
000002f0  3d 1a a6 61 d4 b9 3e 9e  d0 c7 ec 72 64 11 df 7d  |=..a..>....rd..}|
00000300  79 7c 52 0e db 9f 74 ce  62 1f 22 ec e3 e9 25 bf  |y|R...t.b."...%.|
00000310  c8 2c 82 18 e9 a9 cc 1b  78 b1 98 ec 44 9c 34 dc  |.,......x...D.4.|
00000320  e5 bb 9a 4f ce b7 b4 9c  6f b6 6a 14 5e 87 af 35  |...O....o.j.^..5|
00000330  e9 a7 61 f1 2b 61 d5 d9  e7 41 60 25 b5 70 91 19  |..a.+a...A`%.p..|
00000340  4a 42 4a d3 54 d1 5c 98  ca 1f b9 40 d1 45 37 8c  |JBJ.T.\....@.E7.|
00000350  05 5c dd 8c 9a dd fa 51  54 a3 c3 be 05 77 48 b5  |.\.....QT....wH.|
00000360  2c 2d 9a 37 f5 d5 6d 14  1e 9f 06 aa 25 51 31 c8  |,-.7..m.....%Q1.|
00000370  5e 9e fa 1e c9 3c ee a5  55 24 6d 6c 60 [illegible] aa d8  |^....<..U$ml`...|
00000380  7b ae be 95 0d d8 b0 8a  11 ca 26 98 02 5b 1a 62  |{.........&..[.b|
00000390  58 b1 34 81 6b 13 09 de  10 9a 67 e8 [illegible] [illegible] 8f 40  |X.4.k.....g....@|
000003a0  4b df 31 bd 4b 93 f5 50  c4 c6 14 80 4d 02 a6 fa  |K.1.K..P....M...|
000003b0  08 a3 a1 18 e3 5a 4a 8a  72 92 44 04 6d 24 18 59  |.....ZJ.r.D.m$.Y|
000003c0  d3 3d 01 78 8d 5a 97 a0  bb 72 b2 82 a6 82 86 dc  |.=.x.Z...r......|
000003d0  97 35 79 37 53 00 8f c6  95 61 5a 80 01 a2 07 03  |.5y7S....aZ.....|
000003e0  66 34 e5 88 cb 53 20 4f  e0 04 59                 |f4...S O..Y|
```

Figure 4; Huggins's capture of PACTOR packets directly from a radio, with the start of sequential packets marked (02, followed by packet length) and the End of File marked (04)



```
| Trimode 1.3.25.|
|0 Wash. DC Area |
|Gateway-Lee@n2le|
|e.com - VAR Trim|
|ode 1.3.25.0 Was|
|h. DC Area Gatew|
|ay-Lee@n2lee.com|
| - VARA Users - |
|Update 3.0.2 req|
|uired..KM4HRR ha|
|s 118 minutes re|
|maining with N2L|
|EE.[WL2K-5.0-B2F|
|WIHJM$].;PQ: 701|
|16537.CMS via N2|
|LEE >.;FW: KM4HR|
|R.[RMS Express-1|
|.5.22.0-B2FHM$].|
|;PR: 09966346.; |
|N2LEE DE KM4HRR |
|(FM18IW).FC EM 2|
|YVAFEECIB8J 903 |
|591 0.F> 8B.Y.Y.|
|e: //WL2K My sec|
|ond Winlink emai|
|1.0....d......z.|
|mf......7|.Nw...|
|.a.@>1.=........|
|W`....O.....,B..|
|...8.T../..\../.|
|......x.........|
|...Y{...|.....b.|
|..)W...w..:.....|
|G......NB.f$rxB.|
|.|X@..\".E.W,.A-|
|-^.w........7...|
```

Figure 5: The FC proposal that preceded the packet capture by Huggins.

Huggins and I were unsure how to deal with the <<STX> (02 hex), total number of bytes > first two characters of each packet, and initially neither of us understood the required bytes at the beginning of the file, giving a file size.

Examination of the 1999 FBB lzhuf_1 source code, and John Wiseman's source code provided the necessary clue, and these two source codes are discussed in the Appendix. They revealed that we needed to provide the length of the (unencoded) file, in unsigned long format, little-endian (Intel microprocessor) arrangement.

Employing his expertise with this clue, on Tuesday August 6, 2019, John Huggins then correctly deduced that the first bytes had to be the unencoded length in little-endian (Intel) format: **903 (0x0387) the unencoded file size** and inserted them at the beginning of the file as shown below:[37] [38]

```
==> ztemptest.hex <==
00000000 87 03 00 00 00 00 00 00 ec f5 7a 1c 6d 66 fb cb |..........z.mf..|
00000010 e2 e6 f4 ba 37 7c fc 4e 77 13 ad 99 cb 61 fb 40 |....7|.Nw....a.@|
00000020 3e 31 81 3d e6 f7 8b bb b0 e1 d6 e0 57 60 d7 f0 |>1.=........W`..|
00000030 b0 a8 4f b6 b5 f0 02 ff 2c 42 fd cf f7 d4 0a 38 |..O.....,B.....8|
```

Huggins then additionally stripped out the unnecessary <02> <size> packet headers, using his binary file editor (hexer) and ended up with the following (presented as a hex dump of the binary data file):[39]

```
00000000  87 03 00 00 00 00 00 00  ec f5 7a 1c 6d 66 fb cb  |..........z.mf..|
00000010  e2 e6 f4 ba 37 7c fc 4e  77 13 ad 99 cb 61 fb 40  |....7|.Nw....a.@|
00000020  3e 31 81 3d e6 f7 8b bb  b0 e1 d6 e0 57 60 d7 f0  |>1.=........W`..|
00000030  b0 a8 4f b6 b5 f0 02 ff  2c 42 fd cf f7 d4 0a 38  |..O.....,B.....8|
00000040  82 54 9b ca 2f df e6 5c  ae be 2f 03 a9 de 90 9e  |.T../..\../.....|
00000050  1c 99 78 17 e3 92 ef c8  d1 ce 9b 1c 03 ee fb 59  |..x............Y|
00000060  7b ec e5 ca 7c f1 0e d6  0c 7f 62 ac 9a af 29 57  |{...|.....b...)W|
00000070  ff b3 8a 77 fa f8 3a c6  85 f7 f0 a5 47 dd e8 13  |...w..:.....G...|
00000080  16 8e 9c 4e 42 14 66 24  72 78 42 dc bf 7c 58 40  |...NB.f$rxB..|X@|
00000090  eb 14 5c 22 83 45 02 57  2c 90 41 2d 2d 5e b4 77  |..\".E.W,.A--^.w|
000000a0  bf b1 8a 8a 98 91 ae 02  37 e5 f7 dd 0e 0c 84 9c  |........7.......|
000000b0  11 f8 81 61 13 41 c2 ec  c1 42 f2 a5 94 25 f5 46  |...a.A...B...%.F|
000000c0  88 b1 06 1a 7f 81 bb fe  9a c7 2e ba 24 f3 c3 43  |............$..C|
000000d0  f5 fd d2 2d e4 60 f6 bb  43 52 7b e6 85 82 f1 99  |...-.`..CR{.....|
000000e0  74 ae f6 0a b4 48 80 35  c3 63 b1 f5 f5 de 8c 6e  |t....H.5.c.....n|
000000f0  88 b7 20 9d 1d 99 9b 00  50 61 b7 84 9f fb 3f 9d  |.. .....Pa....?.|
00000100  ae c5 f8 cd b5 10 cb 8a  bd fe ab dd e6 47 e6 d3  |.............G..|
00000110  53 97 97 05 eb a9 5e f3  8e 61 ea b8 22 a2 18 55  |S.....^..a.."..U|
00000120  d6 6a eb 7d 2a fc cc f9  61 da da 07 57 f2 d5 76  |.j.}*...a...W..v|
00000130  0c 88 fa ee 69 97 f8 9e  ee ce 6b 69 14 f5 57 44  |....i.....ki..WD|
00000140  b9 59 b6 80 b8 32 37 c8  7d 56 b2 34 ff 09 b2 39  |.Y...27.}V.4...9|
00000150  ed f1 5d 0c eb 74 84 7e  79 8f 3d 1a a6 61 d4 b9  |..]..t.~y.=..a..|
00000160  3e 9e d0 c7 ec 72 64 11  df 7d 79 7c 52 0e db 9f  |>....rd..}y|R...|
00000170  74 ce 62 1f 22 ec e3 e9  25 bf c8 2c 82 18 e9 a9  |t.b."...%..,....|
00000180  cc 1b 78 b1 98 ec 44 9c  34 dc e5 bb 9a 4f ce b7  |..x...D.4....O..|
00000190  b4 9c 6f b6 6a 14 5e 87  af 35 e9 a7 61 f1 2b 61  |..o.j.^..5..a.+a|
000001a0  d5 d9 e7 41 60 25 b5 70  91 19 4a 42 4a d3 54 d1  |...A`%.p..JBJ.T.|
000001b0  5c 98 ca 1f b9 40 d1 45  37 8c 05 5c dd 8c 9a dd  |\....@.E7..\....|
000001c0  fa 51 54 a3 c3 be 05 77  48 b5 2c 2d 9a 37 f5 d5  |.QT....wH.,-.7..|
000001d0  6d 14 1e 9f 06 aa 25 51  31 c8 5e 9e fa 1e c9 3c  |m.....%Q1.^....<|
000001e0  ee a5 55 24 6d 6c 60 de  aa d8 7b ae be 95 0d d8  |..U$ml`...{.....|
000001f0  b0 8a 11 ca 26 98 1a 62  58 b1 34 81 6b 13 09 de  |....&..bX.4.k...|
00000200  10 9a 67 e8 88 ef 8f 40  4b df 31 bd 4b 93 f5 50  |..g....@K.1.K..P|
```

[37] Huggins; Explanation of all steps he took to decode the file: https://forums.qrz.com/index.php?threads/decode-off-the-air-winlink-message-request-for-programming-help.668470/page-3#post-5154052

[38] Note that the GO lzhuf package indicates that the length is required, but the CRC here may be optional; there is an 8-bit CRC at the end of the data, immediately after the EOF (04 hex) in the FC protocol. https://godoc.org/github.com/la5nta/wl2k-go/lzhuf

[39] Huggins, https://www.hamradio.me/graphs/WinlinkTests/ztemptest.hex

```
00000210  c4 c6 14 80 4d 02 a6 fa  08 a3 a1 18 e3 5a 4a 8a  |....M........ZJ.|
00000220  72 92 44 04 6d 24 18 59  d3 3d 01 78 8d 5a 97 a0  |r.D.m$.Y.=.x.Z..|
00000230  bb 72 b2 82 a6 82 86 dc  97 35 79 37 53 00 8f c6  |.r.......5y7S...|
00000240  95 61 5a 80 01 a2 07 03  66 34 e5 88 cb 53 20 4f  |.aZ.....f4...S O|
00000250  e0 59                                             |.Y|
00000252
```

John Huggins had already compiled the lzhuf.c file (and I also, following his lead). For decoding it requires the following command:

```
lzhuf d <file to be decoded>  <output file to be filled>
```

and using this corrected data file with the re-inserted file size, Huggins **immediately brought back the entire WINLINK email**[40], which included the message being replied to, including all the MIME headers specified in the WINLINK documentation file:[41] A completely perfect decode of an over-the-air captured WINLINK message on August 6, 2019.

This was approximately 17 weeks after Huggins began working on this problem.

MID: 2YVAFEECIB8J
Date: 2019/07/29 16:29
Type: Private
From: KM4HRR
To: KW4SHP
Subject: Re: //WL2K My second Winlink email
Mbo: KM4HRR
Body: 748

Fanatstic! Awesome stuff. Look slike it's working justr fine. Congrats!!!

73,
Brendan KM4HR

----- Message from KW4SHP sent 2019/07/28 23:41 -----

Message ID: FO40YS492PHY
Date: 2019/07/28 23:41
From: KW4SHP
To: KM4HRR
Source: KW4SHP

[40] Huggins: https://www.hamradio.me/graphs/WinlinkTests/ztemptest.txt

[41] WINLINK: https://winlink.org/B2F

Subject: //WL2K My second Winlink email

Brendan:

Just completed my Winlink HT setup.

I purchased a mobilinkd TNC3 and attached it to my Baofeng BF-F8HP with a SlimJim in the attic.
I'm using Bluetooth from the TNC3 to my desktop PC running Winlink software.

Thanks for stoking my interest in this at Field Day.

My goal is to replace the PC with a Raspberry Pi using my Android Cell as a mouse/keyboard/display via VNC over Wifi to the Raspberry.

Baby steps....

73
Steve Palmer

View of message from off-the-air reconstruction by Huggins

This exact message can be viewed on the WINLINK distributed receiver viewer (minus the MIME header) as follows:

Fanatstic! Awesome stuff. Look slike it's working justr fine. Congrats!!!

73,
Brendan KM4HR

----- Message from KW4SHP sent 2019/07/28 23:41 -----

Message ID: FO40YS492PHY
Date: 2019/07/28 23:41
From: KW4SHP
To: KM4HRR
Source: KW4SHP
Subject: //WL2K My second Winlink email

Brendan:

Just completed my Winlink HT setup.

I purchased a mobilinkd TNC3 and attached it to my Baofeng BF-F8HP with a SlimJim in the attic.
I'm using Bluetooth from the TNC3 to my desktop PC running Winlink software.

Thanks for stoking my interest in this at Field Day.

My goal is to replace the PC with a Raspberry Pi using my Android Cell as a mouse/keyboard/display via VNC over Wifi to the Raspberry.

Baby steps....

73
Steve Palmer

View of message from the WINLINK VIEWER

Notes on the number of bytes in the size variable:
C code is not always machine-independent (witness the big-endian issue) and the size of an unsigned long is reportedly usually 8 bytes[42] This may explain why John Huggins had to fill 8 bytes of the array with the little-endian version of the unencode file size.

Additionally, the packet headers are dependent on the particular TNC / mode utiized, and are added during transmission AFTER the file is compressed, so on the receiving side, they must be removed BEFORE the file is uncompressed.

4. Where To Go After This Success?

Thus, it is now quite obviously only a software development task to build a system which can decode WINLINK messages directly off the air, *presuming the actual ability to HEAR the signal with sufficient signal to noise ratio*. And as I have proposed, the development of a diversity receiving system would make this even more successful.[43] [44] [45] The SCS corporation is literally creating new firmware to assist in ordering the packets from multiple streams, recognizing repeats and solving the other technical issues, with repeated releases of new firmware almost immediately after requests.[46] [47]

[42] https://www.tutorialspoint.com/cprogramming/c_data_types

[43] See p 6ff here: https://www.qsl.net/nf4rc/2019/SpyingOnWINLINKV2.pdf or in the published Amazon text: https://www.amazon.com/dp/1080563199

[44] See p11 in https://ecfsapi.fcc.gov/file/10730701023399/ResponseToRappaportJuly24Filing.pdf

[45] See p4 in https://ecfsapi.fcc.gov/file/10722131064325/REPLYtoCarsonExParteFilingProposal.pdf

[46] Helfert: https://forums.qrz.com/index.php?threads/huggins-did-it.667817/page-44#post-5155634 This is at least the second such firmware addition almost immediately created after an amateur request.

[47] SCS updated firmware: https://1drv.ms/u/s!AtLEi-HvlIdLjXzwtL9tmLmSzmCl?e=KBnNEZ

With the level of packet data now made easily available to the software developer, a ground-breaking diversity-receiver monitoring system is now within reach, and would make for a fascinating thesis for a graduate student at a prestigious American school of engineering.

To me, it is curious that brilliant persons would claim that this was a matter of national security, while possessing all the skills and resources necessary to accomplish it….but leave it to a semi-retired physician and an interested amateur radio operator to accomplish the task.

However, what John Huggins accomplished is even greater than the solution of the decompression technique – for the first time in the ongoing discussion, we have evidence that even if you significantly foul up the process (as was done by starting with an incorrect set of data in the first bytes of the file for at least 8 bytes and improperly leave various characters in the text as well) – **there is still some chance of recognizing the contents**. I am not an expert on the inner workings of the **`lzhuf`** compression system, and certainly further research would be useful in this area to determine the sensitivity of the system to various levels of error, and if desired, to develop mitigating strategies based on a knowledge of how the compression tables are constructed and utilized.

In conclusion:

- The claims made that WINLINK transmissions cannot be decoded, or cannot be decoded from the air, are conclusively proven false.
- The compression systems, and the bases of protocols that the WINLINK system utilizes **have been in use in amateur radio for at least 20 years**, throughout most of which Dr. Rappaport and others never apparently brought up any of these concerns, until 2016.
- The Nation remained secure, despite the use of this compression for 20 years.
- Two amateurs managed to acquire the necessary understand to capture and decompress WINLINK messages, in a matter of 17 weeks.
- Had the Petitioner and proponents, with their combined vast expertise and resources, truely wished to have had an on-the-air monitoring system, they had at least 3 years until now, since their first publicly expressed concern, in which to even begin experiments, yet none have been reported. They likely could have done what two amateurs did, but in days, not weeks.
- Possible amateur research and development in to such areas as distributed receiving, and automatic power control has never occurred, and *should*, if these harassing complaints can be swept aside. While some good outcomes have definitely come from the Petitioner's efforts, it is time to *stop this and move forward*.

Sincerely,

/s/ Gordon L. Gibby MD KX4Z
15216 NW 41st Avenue
Newberry, FL 32669

EMAIL DISTRIBUTION LIST

Eric Burger	Eric.Burger@fcc.gov
Lisa Fowlkes	Lisa.Fowlkes@fcc.gov
Ajit Pai	Ajit.Pai@fcc.gov
Geoffrey Starks	geoffrey.starks@fcc.gov
Michael O’Rielly	mike.o'rielly@fcc.gov
Jessica Rosenworcel	Jessica.Rosenworcel@fcc.gov
Rachael Bender	Rachael.Bender@fcc.gov
Zenji Nakazawa	Zenji.Nakazawa@fcc.gov
Michael Wilhelm	Michael.Wilhelm@fcc.gov
Curt Bartholomew	Curt.Bartholomew@fcc.gov
Erin McGrath	Erin.McGrath@fcc.gov
Brendan Carr	Brendan.Carr@fcc.gov
Julius Knapp	Julius.Knapp@fcc.gov
Michael Ha	michael.ha@fcc.gov
Ronald Repasi	Ronald.Repasi@fcc.gov
Bruce Jacobs	Bruce.Jacobs@fcc.gov
Donald Stockdale	Donald.Stockdale@fcc.gov
Roger Noel	Roger.Noel@fcc.gov
Scot Stone	Scot.Stone@fcc.gov
Rosemary Harold	Rosemary.Harold@fcc.gov
Charles Cooper	charles.cooper@fcc.gov
Laura Smith	Laura.Smith@fcc.gov

APPENDIX: The Clue from the LZHUF Algorithm

Note that the first accomplishment of the 1999 FBB decode routine reads an unsigned int from the file:

```
void Decode(void)  /* recover */
{
            char *ptr;
            int  i, j, k, r, c;
            unsigned long int  count;
            unsigned int  crc_read;

            if (version_1)
            {
                  if (fread(&crc_read, sizeof crc_read, 1, infile) < 1)
                        Error("Can't read");  /* read size of text */
                  printf("File CRC  = %04x\n", crc_read);
            }
```

----

and the first section of John Wiseman's Decode routine also reads in a size. The 02 fa at the start of the packet dump that Huggins had been using would clearly be incorrect for the size of the file, possibly explaining why the executable ran on interminably without stopping.

```
unsigned     long   textsize = 0, codesize = 0;


void Decode(CIRCUIT * conn)
{
    unsigned char *ptr;
    char * StartofMsg;
    short  i, j, k, r;
    short c;
    unsigned long count;
    unsigned short  crc_read;
    int Index = 0;
    struct FBBHeaderLine * FBBHeader= &conn->FBBHeaders[0];    // The Headers from
an FFB forward
    BOOL NTS = FALSE;

    getbuf = 0;
    getlen = 0;
    textsize = 0;
    codesize = 0;

    infile = &conn->MailBuffer[0];

    crc = 0;

    . . .   <inapplicable code skipped for brevity>

#else

    for (i = 0 ; i < sizeof(textsize) ; i++)
        ptr[i] = (unsigned char)crc_fgetc();

#endif

    // Temp fix for duff MACBPQ (Message Length sent big-endian)

    if (textsize > 500000)
    {
        char x[4];
        char y[4];

        memcpy(x, &textsize, 4);
        y[0] = x[3];
        y[1] = x[2];
        y[2] = x[1];
        y[3] = x[0];

        memcpy(&textsize, y, 4);

        if (textsize > 5000000)
        {
            nodeprintf(conn, "*** Message Size Invalid %d\r", textsize);
            Debugprintf("*** Message Size Invalid %d\r", textsize);
            free(conn->MailBuffer);
            conn->MailBufferSize=0;
            conn->MailBuffer=0;
            conn->CloseAfterFlush = 20;                   // 2 Secs
```

```
                        return;
                }
        }

        Logprintf(LOG_BBS, conn, '|', "Uncompressing Message Comp Len %d Msg Len %d CRC
%x",
                        conn->TempMsg->length, textsize, crc);
```

The answer is that we needed to provide the size of the unencoded file as an unsigned long, as the first item of the file provided to lzhuf.