3 Breaking Globalstar Satellite Communications

by Colby Moore

It might be an understatement to say that hackers have a fascination with satellites. Fortunately, with advancements in Software Defined Radio such as the Ettus Research USRP and Michael Ossmann's HackRF, satellite hacking is now not only feasible, but affordable. Here we'll discuss the reverse engineering of Globalstar's Simplex Data Service, allowing for interception of communications and injection of data back into the network.

Rumor has it, that after deployment, Globalstar's first generation of satellites began to fail, possibly due to poor radiation hardening. This affected the return path data link, where Globalstar would transmit to a user. To salvage the damaged satellite network, Globalstar introduced a line of simplex products that enable short, one-way communication from the user to Globalstar.

The nature of the service makes it ideal for asset tracking and remote sensor monitoring. While extremely popular with oil and gas, military, and shipping industries, this technology is also widely used by consumers. A company called SPOT produces consumer-grade asset trackers and personal locator beacons that utilize this same technology.

Globalstar touts their simplex service as "extremely difficult" to intercept, noting that the signal's "Low-Probability-of-Intercept (LPI) and Low- Probability-of-Detection(LPD) provide over-the-air security."[7]

In this article I'll outline the basics for reverse engineering the Globalstar Simplex Data Services modulation scheme and protocol, and will provide the technical information necessary to interface with the network.

3.1 Network Architecture

The network is comprised of many Low Earth Orbit, bent-pipe satellites. Data is transmitted from the user to the satellite on an uplink frequency and repeated back to Earth on a downlink frequency. Globalstar ground stations all over the world listen for this downlink data, interpret it, and expose it to the user via an Internet-facing back-end. Each ground station provides a several thousand mile window of data coverage.

Bent-pipe satellites are "dumb" in that they do not modify the transmitted data. This means that the data on the uplink is the same on the downlink. Thus, with the right knowledge, a skilled adversary can intercept data on either link.

3.2 Tools and Code

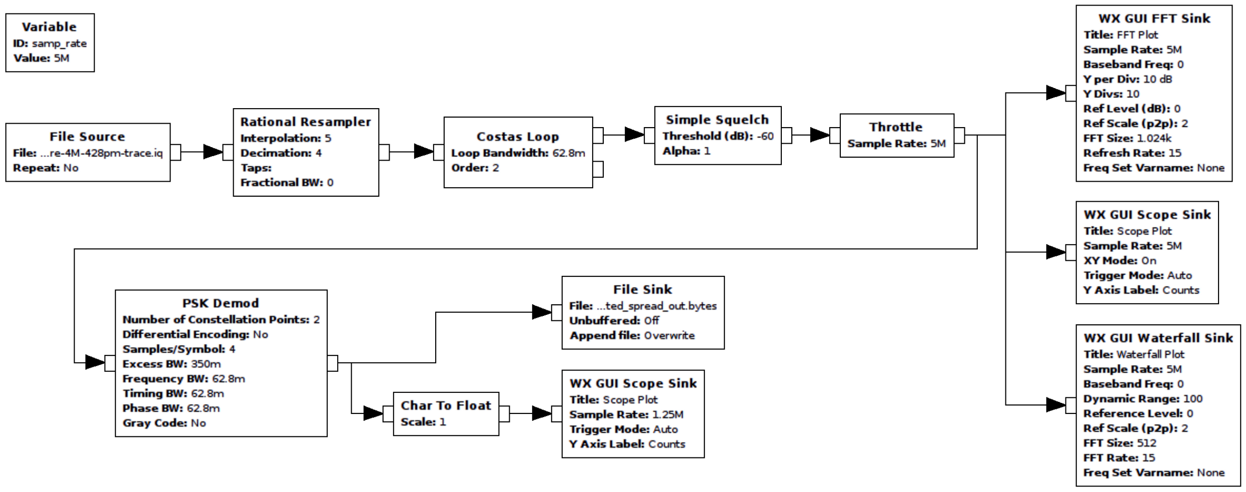
This research was conducted using GNURadio and Python for data processing and an Ettus Research B200 for RF work. Custom proof-of-concept toolsets were written for DSSS and packet decoding. Devices tested include a SPOT Generation 3, a SPOT Trace, and a SmartOne A.

3.3 Frequencies and Antennas

Four frequencies are allocated for the simplex data uplink. Current testing has only shown operation on channel A.

Channel	Frequency
A	1611.25 MHz
B	1613.75 MHz
C	1616.25 MHz
D	1618.78 MHz

[7] `http://productsupport.globalstar.com/2009/02/09/are-simplex-messages-secure/`



Variable
ID: samp_rate
Value: 5M

File Source
File: ...re-4M-428pm-trace.iq
Repeat: No

Rational Resampler
Interpolation: 5
Decimation: 4
Taps:
Fractional BW: 0

Costas Loop
Loop Bandwidth: 62.8m
Order: 2

Simple Squelch
Threshold (dB): -60
Alpha: 1

Throttle
Sample Rate: 5M

WX GUI FFT Sink
Title: FFT Plot
Sample Rate: 5M
Baseband Freq: 0
Y per Div: 10 dB
Y Divs: 10
Ref Level (dB): 0
Ref Scale (p2p): 2
FFT Size: 1.024k
Refresh Rate: 15
Freq Set Varname: None

WX GUI Scope Sink
Title: Scope Plot
Sample Rate: 5M
XY Mode: On
Trigger Mode: Auto
Y Axis Label: Counts

WX GUI Waterfall Sink
Title: Waterfall Plot
Sample Rate: 5M
Baseband Freq: 0
Dynamic Range: 100
Reference Level: 0
Ref Scale (p2p): 2
FFT Size: 512
FFT Rate: 15
Freq Set Varname: None

PSK Demod
Number of Constellation Points: 2
Differential Encoding: No
Samples/Symbol: 4
Excess BW: 350m
Frequency BW: 62.8m
Timing BW: 62.8m
Phase BW: 62.8m
Gray Code: No

File Sink
File: ...ted_spread_out.bytes
Unbuffered: Off
Append file: Overwrite

Char To Float
Scale: 1

WX GUI Scope Sink
Title: Scope Plot
Sample Rate: 1.25M
Trigger Mode: Auto
Y Axis Label: Counts

Globalstar uses left-hand circular-polarized antennas for transmission of simplex data from the user to the satellite. The Globalstar GSP-1620 antenna, designed for transmitting from the user to a satellite, has proven adequate for experimentation.

Downlink is a bit more complicated, and far more faint. Channels vary by satellite, but are within the 6875–7055 MHz range. Both RHCP and LHCP are used for downlink.

3.4 Direct Sequence Spread Spectrum

Devices using the simplex data service implement direct sequence spread spectrum (DSSS) modulation to reliably transmit data using low power. DSSS is a modulation scheme that works by mixing a slow data signal with a very fast Pseudo Noise (PN) sequence. Since the pseudo-random sequence is known, the resulting signal retains all of the original data information but spread over a much wider spectrum. Among other benefits, this process makes the signal more tolerant to interference.

In Globalstar's implementation of DSSS, packet data is first modulated as non-differential BPSK at 100.04 bits/second, then spread using a repeating 255 chip PN sequence at a rate of 1,250,000 chips/second. Here "chip" refers to one bit of a PN sequence, so that it is not confused with actual data bits.

3.5 Pseudo Noise Sequence / M-Sequences

Pseudo Noise (PN) sequences are periodic binary sequences known by both the transmitter and receiver. Without this sequence, data cannot be received. The simplex data service uses a specific type of PN sequence called an *M-Sequence*.

M-Sequences have the unique property of having a strong autocorrelation for phase shifts of zero but very poor correlation for any other phase shift. This makes the detection of the PN in unknown data, and subsequently locking on to a DSSS signal, relatively simple.

All simplex data network devices examined use the same PN sequence to transmit data. By knowing one code, all network data can be intercepted.

3.6 Obtaining The M-Sequence

In order to intercept network data, the PN sequence must be recovered. For each bit of data transmitted, the PN sequence repeats 49 times. Data packets contain 144 bits.

```
1,250,000 chips      1 second       1 PN sequence
--------------- x -------------- x --------------- = 49 PN sequences/bit
   1 second        100.04 bits       255 chips
```

The PN sequence never crosses a bit boundary, so it can be inferred that

```
xor(PN, data) == PN
```

By decoding the transmitted data stream as BPSK,[8] we can demodulate a spread bitstream. Note that demodulation in this manner negates any processing gain provided from DSSS and thus can only be received over short distances, so for long distances you will need to use a proper DSSS implementation.

Viewing the demodulated bitstream, a repeating sequence is observed. This is the PN, the spreading code key to the kingdom.

The simplex data network PN code is 1111111100101101011011101010101110010011011010011001101-00011101101100010001001111010010010000111100010100111000111110101111001110100001010110010-100010110000011001000110000110111111011100001000001001010100101111110000001110011000110101-00000001011101111101100.

3.7 Despreading

DSSS theory states that to decode a DSSS-modulated signal, a received signal must be mixed once again with the modulating PN sequence; the original data signal will then fall out. However, for this to work, the PN sequence needs to be phase-aligned with the mixed PN/data signal, otherwise only noise will emerge.

Alignment of the PN sequence to the data stream if accomplished by correlating the PN sequence against the incoming datastream at each sample. When aligned, the correlation will peak. To despread, this correlation peak is tracked and the PN is mixed with the sampled RF data. The resulting signal is the 100.04 bit/second non-differential BPSK modulated packet data.

3.8 Decoding and Locations

Once the signal is despread, a BPSK demodulator is used to recover data. The result is a binary stream, 144 bytes in length, representing one data packet. The data packet format is as follows:

Field	Bits	Description
Preamble	(10)	0000001011 signifies start of packet
ESN	(26)	3 bits for manufacturer ID and 23 bits for unit ID
Message #	(4)	message number modulo 16, saved in non-volatile memory
Packet #	(4)	number of packets in a message
Packet Seq. #	(4)	sequence number for each packet in a message
User Data	(72)	9 bytes of user information, MSB first
CRC24	(24)	CRC is 24 bits with polynomial: 114377431

Simplex data packets can technically transmit any 72 bits of user defined data. However, the network is predominantly used for asset tracking and thus many packets contain GPS coordinates being relayed from tracking devices. This data scheme for GPS coordinates can be interpreted with the following Python code.

```
latitude = int(user_data[8:32],2) * 90 / 2**23
longitude = 360 - int(user_data[32:56],2) * 180 / 2**23
```

[8]DSSS theory shows us that DSSS is the same as BPSK for a BPSK data signal.

3.9 CRC

Packets are verified using a 24 bit CRC. The data packet minus the preamble and CRC are fed into the CRC algorithm in order to verify or generate a CRC. The following Python code implements the CRC algorithm.

```
def crcTwentyfour(TX_Data):

    k = 0
    m = 0

    TempCRC = 0
    Crc = 0xFFFFFF

    for k in range(0,14):  #calc checksum on 14 bytes starting with ESN

        #offset to skip part of the preamble (dictated by algorithm)
        TempCRC = int(TX_Data[ (k*8)+8 : (k*8)+8+8 ], 2)

        if 0 == k:
            #skip 2 preamble bits in byte0
            TempCRC = TempCRC & 0x3f


        Crc = Crc ^ (TempCRC)<<16


        for m in range(0,8):
            Crc = Crc << 1

            if Crc & 0x1000000:
                #seed CRC
                Crc = Crc ^ 0114377431L


    Crc = (~Crc) & 0xffffff;
    #end crc generation. lowest 24 bits of the long hold the CRC

    #first CRC byte to TX_Data
    byte14 = (Crc & 0x00ff0000) >> 16

    #second CRC byte to TX_Data
    byte15 = (Crc & 0x0000ff00) >> 8

    #third CRC byte to TX_Data
    byte16 = (Crc & 0x000000ff)

    final_crc = (byte14 << 16) | (byte15 << 8) | byte16

    if final_crc != int(TX_Data[120:144], 2):
        print "Error: CRC failed"
        sys.exit(0)
```

3.10 Transmitting

DISCLAIMER: It is most likely illegal to transmit on Globalstar's frequencies where you live. Do so at your own risk. Remember, no one likes late night visits from the FCC and it would really suck if you interrupted someone's emergency communication!

By knowing the secret PN code, modulation parameters, data format, and CRC, it is possible to craft custom data packets and inject them back into the satellite network. The process is as follows:

- Generate a custom packet

- Calculate and affix the packet's CRC
- Spread the packet using the Globalstar PN sequence
- BPSK modulate the spread data and transmit on the RF carrier

Various SDR boards should have enough power to communicate with the network, however COTS amplifiers are available for less than a few hundred dollars. Specifications suggests a transmit power of about 200 milliwatts.

3.11 Spoofing

SPOT produces a series of asset trackers called SPOT Trace. SPOT also provides `SPOT_Device_Updater.pkg`, an OS X update utility, to configure various device settings. This utility contains development code that is never called by the consumer application.

The updater app package contains `SPOT3FirmwareTool.jar`. Decompilation shows that a UI view calls a method `writeESN()` in `SPOTDevice.class`. You read that correctly, they included the functionality to program arbitrary serial numbers to SPOT devices!

This UI can be called with a simple Java utility.

```java
import com.globalstar.SPOT3FirmwareTool.UI.DebugConsole;

public class SpotDebugConsole {
     public static void main(String[] args) {
        DebugConsole.main(args);
    }
}
```

Upon execution, a debug console is launched, allowing the writing of arbitrary settings including ESNs, to the SPOT device. (This functionality was included in Spot Device Updater 1.4 but has since been removed.)

3.12 Impact

The simplex data network is implemented in countless places worldwide. Everything from SCADA monitoring to emergency communications relies on this network. To find that there is no encryption or authentication on the services examined is sad. And to see that injection back into the network is possible is even worse.

Using the specifications outlined here, it is possible—among other things—to intercept communications and track assets over time, spoof an asset's location, or even cancel emergency help messages from personal locator beacons.

One could also enhance their own service, create their own simplex data network device, or use the network to transmit their own covert communications.

3.13 PoC and Resources

This work was presented at BlackHat USA 2015 and proof-of-concept code is available both by Github and within this PDF file.[9]

[9] `git clone https://github.com/synack/globalstar`
`unzip pocorgtfo09.pdf globalstar.tar.bz2`