MATT KNIGHT // BASTILLE NETWORKS

DECODING LORA

EXPLORING NEXT-GENERATION WIRELESS NETWORKS

WHO'S THIS GUY

- Matt Knight
- Software Engineer and Threat Researcher @ Bastille
- BE & BA from Dartmouth
- Background in electrical engineering, embedded software, etc.
- Applied RF security research

matt@Bastille.net
@embeddedsec

BEFORE WE GET STARTED…

- No 0-days or exploits
- We're going to take apart a cutting edge protocol

- Quick show of hands…

WHY IS THIS RELEVANT

- Cisco IBSG: 50 billion devices by 2020
  - Fewer wires every year

- Wireshark and Monitor Mode weren't always a thing
- Low-level access to interfaces is paramount for enabling comprehensive security research

AGENDA

1. Introduce LPWANs
2. Review technical radio concepts
3. Demodulate and decode the LoRa PHY with SDR
4. Introduce gr-lora

WHAT´S LORA?

WIRELESS IOT PROTOCOL

IOT == EMBEDDED

802.15.4

(ZIGBEE)

802.11

BLUETOOTH

BLE

AND [. . .]

WHAT'S WRONG WITH [YOUR FAVORITE PROTOCOL]?

- Require local provisioning
- Some require gateways to connect out
- [802.11] Thirsty battery requirements

- What's ideal then?

HOW ABOUT CELLULAR?

1. IT WORKS EVERYWHERE
2. EASY TO INSTALL

IT'S THIRSTY

AND IT'S GOING AWAY*

*2G, THAT IS

DEPRECATION: A DEVELOPER'S CONUNDRUM

- AT&T to sunset 2G on January 1, 2017
  - Telstra (Australia) sunset on December 1, 2016!
- Other major carriers to follow

- 2G advantages: ubiquitous, battery-conscious, somewhat inexpensive
  - Exactly what IoT devices require

REPLACING 2G

- 3G
  - More expensive
  - Harder power requirements
- 3GPP IoT Standards: LTE-M/NB-LTE Release 13
  - 3 separate IoT focused cellular protocols
  - Not ready by the sunset date, which means…

VOID IN MARKET

INTRODUCING

LPWANS

http://www.gsm-modem.de/M2M/wp-content/uploads/2015/10/LWPAN-versus-LTE.png

LPWAN

- LPWAN: Low Power Wide Area Network
- Like cellular, but optimized for IoT/M2M
  - Network of basestations worldwide
  - Star network to endpoints, UL/DL traffic
  - Range in miles

EMERGING STANDARDS

http://www.itwissen.info/bilder/lpwan-konzepte-fuer-das-internet-of-things-iot.png

AGGRESSIVE INVESTMENT

- SIGFOX raised €150MM last month
  - WSJ: Possible US IPO soon
- Senet and Actility, LoRa backers, raised a combined 51MM
- LoRa Alliance membership tripled last year

http://img.scoop.it/pkne9jh9_fl-PFplrbFgL4XXXL4j3HpexhjNOf_P3YmryPKwJ94QGRtDb3Sbc6KY
https://pbs.twimg.com/profile_images/615790171116601345/Q1PU_OuW.png

OPTIMIZED FOR IOT

- Battery-conscious
  - SIGFOX advertises 10 years on 1 AA battery
- Long range
  - LoRa advertises up to 13.6 miles

HOW!?

- Compare this with…
  - 2G: typically 1-2 miles, max 22 miles, a few days
  - 802.15.4: 10-100 meters, months-years
  - WiFi: 30 meters, a few days

COMPROMISES

EMBRACING COMPROMISE

- Conservative duty-cycling and listening
- Very sparse datagrams
- Highly rate-limited

EMBRACING COMPROMISE

- Examples
  - SIGFOX limits devices to 140 12-byte datagrams per day
  - Weightless-N is uplink-only
  - LoRa Class A devices can only receive downlink momentarily after uplinking

INTRODUCING

LORA

HISTORY

- LPWAN developed by Semtech
- PHY patented in June 2014
- LoRaWAN MAC/NWK stack released in January 2015
- Supported by LoRa Alliance

NOMENCLATURE

- LoRa vs. LoRaWAN
  - LoRa: PHY layer
  - LoRaWAN: MAC, NWK, and APP built on LoRa

LORAWAN NETWORK TOPOLOGY

- Gateway: low-IQ basestation / LoRa concentrator
- Network Server: routes between gateways and application servers
  - Roaming being defined
- Application Servers

http://www.nnnco.com.au/uploads/6/0/7/9/60799127/4996599_orig.png

LORAWAN SECURITY ARCHITECTURE

- NwkSKey: end device to network server
- AppSKey: end device to application server
- Keys can be preconfigured or sent OTA

http://www.nnnco.com.au/uploads/6/0/7/9/60799127/4996599_orig.png

THAT'S IT FOR LORAWAN

ALL LORA
ALL THE TIME

LICENSURE, OR LACK THEREOF

- LoRa uses 900 MHz ISM
  - US: 902-928 MHz
  - EU: 868 MHz
- No special license required

LICENSURE, OR LACK THEREOF

DMA[1]	Call Sign	Move Off-Air	Move to Low VHF	Move to High VHF
New York, NY	WABC-TV	602,638,380	351,539,055	NA
New York, NY	WCBS-TV	900,000,000	675,000,000	360,000,000
New York, NY	WDVB-CD	424,772,100	318,579,075	169,908,840
New York, NY	WEBR-CD	376,508,700	282,381,525	150,603,480
New York, NY	WEDW	448,103,700	336,077,775	179,241,480
New York, NY	WEPT-CD	129,438,900	97,079,175	51,775,560
New York, NY	WFTY-DT	438,170,400	328,627,800	175,268,160
New York, NY	WFUT-DT	788,135,400	591,101,550	315,254,160
New York, NY	WJLP	206,954,325	NA	NA
New York, NY	WLIW	672,446,700	504,335,025	268,978,680
New York, NY	WLNY-TV	483,707,700	362,780,775	193,483,080
New York, NY	WMBC-TV	805,447,800	604,085,850	322,179,120
New York, NY	WMBQ-CD	445,109,400	333,832,050	178,043,760
New York, NY	WMUN-CD	364,589,100	273,441,825	145,835,640
New York, NY	WNBC	869,160,600	651,870,450	347,664,240
New York, NY	WNET	443,269,260	258,573,735	NA
New York, NY	WNJB	475,608,780	277,438,455	NA
New York, NY	WNJN	775,742,400	581,806,800	310,296,960
New York, NY	WNJU	819,342,000	614,506,500	327,736,800
New York, NY	WNYE-TV	770,259,600	577,694,700	308,103,840
New York, NY	WNYJ-TV	625,505,400	469,129,050	250,202,160
New York, NY	WNYW	824,922,000	618,691,500	329,968,800
New York, NY	WPIX	435,535,920	254,062,620	NA
New York, NY	WPXN-TV	745,848,900	559,386,675	298,339,560
New York, NY	WRNN-TV	846,911,700	635,183,775	338,764,680
New York, NY	WTBY-TV	683,951,400	512,963,550	273,580,560
New York, NY	WVVH-CD	31,156,200	23,367,150	12,462,480
New York, NY	WWOR-TV	810,039,600	607,529,700	324,015,840
New York, NY	WXTV-DT	832,842,000	624,631,500	333,136,800
New York, NY	WZME	483,669,000	362,751,750	193,467,600
Los Angeles, CA	KABC-TV	305,469,360	178,190,460	NA
Los Angeles, CA	KAZA-TV	557,458,200	418,093,650	222,983,280
Los Angeles, CA	KBEH	617,519,700	463,139,775	247,007,880
Los Angeles, CA	KCAL-TV	299,840,400	174,906,900	NA
Los Angeles, CA	KCBS-TV	544,991,400	408,743,550	217,996,560
Los Angeles, CA	KCET	492,480,900	369,360,675	196,992,360
Los Angeles, CA	KCOP-TV	333,967,860	194,814,585	NA
Los Angeles, CA	KDOC-TV	539,584,200	404,688,150	215,833,680
Los Angeles, CA	KFTR-DT	573,801,300	430,350,975	229,520,520
Los Angeles, CA	KHTV-CD	457,959,600	343,469,700	183,183,840

- Compare this with cellular
  - Regulatory authorities auctions cellular spectrum licenses for billions
  - Restricts building infrastructure to biggest telcos
- Left: opening bid list for FCC TV whitespace reverse auction

LORAWAN NETWORK PROVIDERS

- Senet
  - Commercial network
- The Things Network
  - Crowdsourced
- LoRaHAM
  - Travis Goodspeed & friends

- **No licensed spectrum required…!!**

http://img.scoop.it/pkne9jh9_fl-PFplrbFgL4XXXL4j3HpexhjNOf_P3YmryPKwJ94QGRtDb3Sbc6KY
http://thenextweb.com/insider/2015/08/19/the-things-network-wants-to-make-every-city-smart-starting-with-amsterdam/#gref

RADICALLY DIFFERENT

OFFENSIVELY

~~OBSCENELY~~ SHORT

RADIO CRASH COURSE

PHY LAYER

- Lowest layer in communication stack
- In wired protocols: voltage, timing, and wiring defining 1s and 0s
- In wireless: patterns of energy being sent over RF medium

http://www.tech-faq.com/wp-content/uploads/2009/01/osimodel.png

WHAT IS RF?

- "Radio Frequency"
- Electromagnetic waves
- Energy

MANIPULATING RF

- Done with a radio
- Hardware defined
  - RF and protocol in silicon
- Software defined (SDR)
  - Flexible silicon handles RF
  - Protocol-specific components implemented in software (CPU or FPGA)

PHY COMPONENTS

- Modulation
  - How digital values are mapped to RF energy
- RF parameters that can be modulated:
  - Amplitude
  - Frequency
  - Phase
  - some combination of the above

http://xenon.colorado.edu/spotlight/kb/gps_basics/modulations.001.png

MODULATION

- Modulators can modulate analog or digital information

- Digital modulation
  - Symbols: discrete RF energy state representing some quantity of information

SYMBOLS ILLUSTRATED

- Top: FSK
- Bottom: OOK/ASK

- Compare with analog modulation
  - Analog = infinite possible symbols
  - Digital = finite number of possible symbols, defined by modulation

http://3.bp.blogspot.com/-w66qwKucSJI/UgWKKmPUP2I/AAAAAAAADYA/B9NMGYzqJVk/s1600/Screenshot-2013-08-03-04-41-52.png

MORE COMPLICATED IOT PHYS

- Spread spectrum
  - Data bits are encoded at a higher rate and occupy more spectrum
  - Resilient to RF noise
- Examples
  - 802.15.4
  - Bluetooth
  - Bluetooth Low Energy

TOOLS USED IN THIS TALK

- Transmitter: Microchip LoRa RN2903 Module
  - Hardware-defined Semtech LoRa radio
- Receiver: Ettus B210
  - Software-defined radio
  - Python, GNURadio, and Baudline to process

LAST THING… FFT

- Fast Fourier Transform

- Decomposes a signal into its component frequencies
  - Any periodic signal can be modeled as the sum of harmonic sine waves

- Allows analysis and visualization of frequency domain

Spectrogram

TECHNICAL DETAILS

LORA

LORA'S PROPRIETARY PHY

- Modulation: Chirp Spread Spectrum (CSS)
- What's a chirp?
  - A signal of continuously increasing or decreasing frequency
  - i.e. a "sweeped tone"

CSS CHIRPS

- **Upchirp**
  - Increasing frequency
- **Downchirp**
  - Decreasing frequency

CSS ADVANTAGES

- Great link budget
  - Resilience to interference
  - Performance at low power
- Resistant to:
  - Multi-path (urban applications)
  - Doppler effect (mobile applications)

Interesting set of pros… where else are chirps used?

RADAR

CHIRPS IN RADAR

- Military and marine radars
  - Wideband and pulse compression

- Open source GNU Chirp Sounder
  - Ionospheric radars
  - Space weather

SEEKING LORA

- December 2015: No LoRa sightings in major US cities
- I encountered Senet at a Meetup event in Cambridge, MA
- While watching one of their marketing videos…

Portsmouth, NH!

ROAD TRIP

York
Portsmouth, NH
Haverhill
Lawrence
Rockport
Gloucester
1 h 13 min
61.2 miles
1 h 14 min
57.6 miles
Cambridge, MA
Brookline

EXAMINING THE LORA PHY FRAME

- Repeated upchirps
  - Preamble/Training Sequence
- 2.25 downchirps
  - Start of frame delimiter (SFD)
- Choppy upchirps of varying length
  - Data!

PHY DATA UNIT STRUCTURE

- Chirp frequency is static
- Chirp "jumps" throughout band

- Instantaneous frequency changes are result of data being modulated onto the chirps
  - Frequency-modulated chirp

DEMODULATING

LORA

BEFORE WE GET STARTED… OSINT

- Technical documentation
  - Semtech European patent application 13154071.8
  - LoRa Alliance LoRaWAN spec (MAC/NWK only, not a PHY spec)
  - Semtech app notes AN1200.18 and AN1200.22
- Prior art
  - Partial implementation in open source rtl-sdrangelove
  - Observations at https://revspace.nl/DecodingLora

SOME DEFINITIONS…

- Bandwidth: width of spectrum occupied by chirp
- Spreading factor: number of bits encoded per symbol (RF state, remember?)
- Chirp rate: first derivative of chirp frequency

SOME DEFINITIONS…

- Bandwidth: width of spectrum occupied by chirp
- Spreading factor: number of bits encoded per symbol (RF state, remember?)
- Chirp rate: first derivative of chirp frequency

SOME NUMBERS…

- US: 125kHz, 250kHz, 500kHz
- US: [7-12] bits per symbol
- bandwidth/(2**spreading_factor)

SO WHAT'S A SYMBOL?

- Instantaneous change in frequency
- FM modulated chirps

DEMODULATING THE PHY

1. Identify the beginning of a frame
2. Find the beginning of the PHY data unit
3. Extract data from instantaneous frequency transitions

- How? We need to quantify the frequency transitions

TRANSFORMING THE SIGNAL

- De-chirping the signal makes analysis easier
- Generate local upchirp and downchirp at the appropriate chirp rate

MATH

$$f_0 * f_1 = f_0 + f_1$$

$$f_0 * -f_0 = 0$$

(complex conjugate)

Upchirp

*

Downchirp

=

Constant Frequency

Why not necessarily DC?
Chirp phase!

*

Upchirp

= ??

*

Downchirp

= ??

REMEMBER SYMBOLS

- Symbol: RF state representing some quantity of information
- LoRa spreading factor: number of bits encoded into each symbol

- How many possible symbols are there?
  - 2**spreading_factor

EXTRACTING SYMBOLS

- Channelize and resample signal to chirp bandwidth
- De-chirp with locally generated signal
- Take FFT of de-chirped signals, where length of FFT is equal to the number of possible symbols
- Most powerful component in each FFT is the symbol!

20
30
40
50

DEMODULATION SUMMARY

1. Identify the beginning of a frame
2. Find the beginning of the PHY data unit
3. Extract data from instantaneous frequency transitions

DEMODULATION SUMMARY

1. Identify the beginning of a packet
   - Preamble signified by continuous up-chirp
   - == same symbol being transmitted over and over
   - Look for some number of consecutive FFTs with maximum power in the same bin

DEMODULATION SUMMARY

2. Find the beginning of the PHY data unit
   - Repeat same process looking for SFD down-chirps
   - Down-chirp is complex conjugate of the up-chirp
   - PHY data unit begins 2 symbols after the SFD

BUT WAIT!

- Accurately finding SFD is essential for receiver synchronization
- Bad sync can spread symbol energy between adjacent FFTs
  - == incorrect data!

30
40
50

SFD SYNC SOLUTION

- Increase FFT time-based precision once preamble is found
- Overlapping FFT sample buffers!

OVERLAPPING FFTS

- Use overlapping FFTs to synchronize to first sample in the first SFD symbol

300
400

40
50
40
50

Unaligned, Non-Overlapped FFTs
Aligned, Non-Overlapped FFTs
0
10
20
30
40
50
0
10
20
30
40
50

OVERLAPPING FFTS

- Use overlapping FFTs to synchronize to first sample in the first SFD symbol
- Re-compute with non-overlapping FFTs to get your data!

DEMODULATION SUMMARY

3. Extract data from chirp frequency transitions
   - Use described FFT method
   - Normalize data about the value of the preamble (always value 0)

WE´RE DONE, RIGHT?

LOL!

DATA ENCODING

- Symbols represent encoded data
  - What?
- Data is transformed before it is transmitted
  - Why?
- Encoding increases OTA resiliency

900 MHz spectrum in the US

RF IS A BRUTAL ENVIRONMENT

- All systems see interference from weather, geomagnetic activity, etc.
- Some systems have protected/reserved frequencies
- LoRa is ISM – TONS OF CHATTER
  - RF contention/collision is guaranteed
- Encoding scrambles and replicates data within frame

WHAT KIND OF ENCODING?

CLOSED SOURCE

WHAT KIND OF ENCODING?

- Semtech European patent application clues:
  1. Symbol "gray indexing" — Adds error tolerance
  2. Data whitening — Induces randomness
  3. Interleaving — Scrambles bits within frame
  4. Forward Error Correction — Adds correcting parity bits
- 4 distinct operations comprise encoding

EASY ENOUGH?

NOPE

Why not?

DOCUMENTATION LIES

CRACKING THE DECODER

- Decoding stages:

1. Symbol “gray indexing”	Gray Coding
2. Data whitening	PRNG algorithm in app note
3. Interleaving	Interleaver algorithm in patent
4. Forward Error Correction	Standard Hamming(N,4) algorithm

First challenge: figure out the whitening PRNG

WHITENING

- Transmitter XORs frame with a pseudorandom sequence
- Receiver XORs RX'd frame with same sequence
  - Returns original frame, b/c XOR is its own inverse
- Why? Randomizing data helps receiver synchronization
  - Similar to Manchester encoding
  - Manchester reduces bit rate, whitening does not

FINDING THE WHITENING SEQUENCE

- Several whitening algos defined in Semtech AN1200.18
- Other examples in rtl-sdrangelove
- None of them worked
- Solution: derive it empirically
  - How can we solve for whitening when there is interleaving and FEC?

MAKING REASONABLE ASSUMPTIONS

- Assuming FEC is Hamming(N,4)
  - (from documentation)
- Chose to use Spreading Factor 8, N=8 –> exactly 1 codeword per symbol
  - SF8 == 8 bits per symbol
  - Hamming(8,4) == 8 bits per codeword

HAMMING(8,4)

- Hamming(8,4) codewords: 4 data bits + 4 parity bits each
  - 0x1 – 0xE: 4 ‘1’s and 4 ‘0’s each
  - 0x0: 0b00000000
  - 0xF: 0b11111111

FEC adds no bits if payload is [0x00,…]

LEVERAGING ASSUMPTIONS

Transmitting [0x00, …]

- Decoding stages:
  1. Symbol “gray indexing”
  2. Data whitening
  3. Interleaving
  4. Forward Error Correction

LEVERAGING ASSUMPTIONS

Transmitting [0x00, ...]

- Decoding stages:
  1. Symbol “gray indexing”
  2. Data whitening
  3. Interleaving
  4. ~~Forward Error Correction~~

Hamming(8,4)

LEVERAGING ASSUMPTIONS

Transmitting [0x00, …]

- Decoding stages:
  1. Symbol "gray indexing"
  2. Data whitening
  3. ~~Interleaving~~
  4. ~~Forward Error Correction~~

Assume interleaver is non-additive

Hamming(8,4)

$$S_{j,mod(i+j,PPM)} = C_{i,j} \text{ , for } 0 \leq i < PPM \text{ and } 0 \leq j < 4 + RDD$$

LEVERAGING ASSUMPTIONS

Transmitting [0x00, …]

- Decoding stages:
  1. Symbol “gray indexing”
  2. Data whitening
  3. ~~Interleaving~~
  4. ~~Forward Error Correction~~

Only 3 permutations

Assume interleaver is non-additive

Hamming(8,4)

LEVERAGING ASSUMPTIONS

Transmitting [0x00, …]

- Decoding stages:
  1. Symbol "gray indexing"
  2. Data whitening
  3. ~~Interleaving~~
  4. ~~Forward Error Correction~~

Only 3 permutations

Assume interleaver is non-additive

Hamming(8,4)

Derive the whitening sequence from the transmitter!

FINDING THE INTERLEAVER

- Semtech European patent application defines diagonal interleaver

$$S_{j,mod(i+j,PPM)} = C_{i,j} \text{ , for } 0 \leq i < PPM \text{ and } 0 \leq j < 4 + RDD$$

- Also doesn't work!

DERIVING THE INTERLEAVER

- This was hard
- Exploit properties of Hamming FEC to reveal patterns
- Hamming(8,4) codewords again:
  - 0x1 – 0xE: 4 ‘1’s and 4 ‘0’s each
  - 0x0: 0b00000000
  - 0xF: 0b11111111

DERIVING THE INTERLEAVER, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

```
Payload: 0x0000000F   Payload: 0x000000F0   Payload: 0x00000F00   Payload: 0x0000F000
 00100011              11000000              00001001              11010000
 00010011              00100101              00000111              00001001
 00001001              00010001              00000011              00000101
 00000111              00001101              00000011              00000110
 00000000              00001100              01000010              00001000
 00000100              00000000              10000001              01000010
 01000011              00000001              00100001              10000000
 10000101              01000111              00010000              00100101

Payload: 0x000F0000   Payload: 0x00F00000   Payload: 0x0F000000   Payload: 0xF0000000
 00000011              01000100              01000001              00001000
 00000011              00000011              10000010              01000101
 01000001              00000000              00100001              10000011
 10000010              01000101              00010010              00100011
 00100010              10001001              00001010              00010011
 00010001              00100010              00000111              00001011
 00001001              00010000              00000011              00000111
 00000000              00001111              00000101              00000111
```

DERIVING THE INTERLEAVER, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

```
Payload: 0x0000000F   Payload: 0x000000F0    Payload: 0x00000F00  Payload: 0x0000F000
 00100011              11000000               00001001             11010000
 00010011              00100101               00000111             00001001
 00001001              00010001               00000011             00000101
 00000111              00001101               00000011             00000110
 00000000              00001100               01000010             00001000
 00000100              00000000               10000001             01000010
 01000011              00000001               00100001             10000000
 10000101              01000111               00010000             00100101

Payload: 0x000F0000   Payload: 0x00F00000    Payload: 0x0F000000  Payload: 0xF0000000
 00000011              01000100               01000001             00001000
 00000011              00000011               10000010             01000101
 01000001              00000000               00100001             10000011
 10000010              01000101               00010010             00100011
 00100010              10001001               00001010             00010011
 00010001              00100010               00000111             00001011
 00001001              00010000               00000011             00000111
 00000000              00001111               00000101             00000111
```

DERIVING THE INTERLEAVER, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

```
Payload: 0x0000000F   Payload: 0x000000F0    Payload: 0x00000F00   Payload: 0x0000F000
 00100011              11000000              00001001              11010000
 00010011              00100101              00000111              00001001
 00001001              00010001              00000011              00000101
 00000111              00001101              00000011              00000110
 00000000              00001100              01000010              00001000
 00000100              00000000              10000001              01000010
 01000011              00000001              00100001              10000000
 10000101              01000111              00010000              00100101

Payload: 0x000F0000   Payload: 0x00F00000    Payload: 0x0F000000   Payload: 0xF0000000
 00000011              01000100              01000001              00001000
 00000011              00000011              10000010              01000101
 01000001              00000000              00100001              10000011
 10000010              01000101              00010010              00100011
 00100010              10001001              00001010              00010011
 00010001              00100010              00000111              00001011
 00001001              00010000              00000011              00000111
 00000000              00001111              00000101              00000111
```

ALIGNING CODEWORDS, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

Payload: **0xDEADBEEF**

```
bit 76543210
    00110011
    10111110
    11111010
    11011101
    10000010
    10000111
    11000000
    10000010
```

	Top							Bot
D								
E								
A								
D								
B								
E								
E								
F								

ALIGNING CODEWORDS, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

Payload: **0xDEADBEEF**

```
bit 76543210
    00110011
    10111110
    11111010
    11011101
    10000010
    10000111
    11000000
    10000010
```

	Top							Bot
D	1	0	1	0	0	0	0	1
E								
A								
D								
B								
E								
E								
F								

ALIGNING CODEWORDS, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

Payload: **0xDEADBEEF**

```
bit 76543210
    00110011
    10111110
    11111010
    11011101
    10000010
    10000111
    11000000
    10000010
```

	Top							Bot
D	1	0	1	0	0	0	0	1
E	0	1	1	1	0	1	0	0
A								
D								
B								
E								
E								
F								

ALIGNING CODEWORDS, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

Payload: **0xDEADBEEF**

```
bit 76543210
    00110011
    10111110
    11111010
    11011101
    10000010
    10000111
    11000000
    10000010
```

	Top							Bot
D	1	0	1	0	0	0	0	1
E	0	1	1	1	0	1	0	0
A	0	1	0	1	1	0	0	0
D								
B								
E								
E								
F								

ALIGNING CODEWORDS, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

Payload: **0xDEADBEEF**

```
bit 76543210
    00110011
    10111110
    11111010
    11011101
    10000010
    10000111
    11000000
    10000010
```

	Top							Bot
D	1	0	1	0	0	0	0	1
E	0	1	1	1	0	1	0	0
A	0	1	0	1	1	0	0	0
D	1	0	1	1	0	0	0	0
B								
E								
E								
F								

ALIGNING CODEWORDS, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

Payload: **0xDEADBEEF**

```
bit 76543210
    00110011
    10111110
    11111010
    11011101
    10000010
    10000111
    11000000
    10000010
```

	Top							Bot
D	1	0	1	0	0	0	0	1
E	0	1	1	1	0	1	0	0
A	0	1	0	1	1	0	0	0
D	1	0	1	1	0	0	0	0
B	1	1	0	0	0	0	1	0
E								
E								
F								

ALIGNING CODEWORDS, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

Payload: **0xDEADBEEF**

```
bit 76543210
    00110011
    10111110
    11111010
    11011101
    10000010
    10000111
    11000000
    10000010
```

	Top							Bot
D	1	0	1	0	0	0	0	1
E	0	1	1	1	0	1	0	0
A	0	1	0	1	1	0	0	0
D	1	0	1	1	0	0	0	0
B	1	1	0	0	0	0	1	0
E	0	1	1	1	0	1	0	0
E								
F								

ALIGNING CODEWORDS, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

Payload: **0xDEADBEEF**

```
bit 76543210
    00110011
    10111110
    11111010
    11011101
    10000010
    10000111
    11000000
    10000010
```

	Top							Bot
D	1	0	1	0	0	0	0	1
E	0	1	1	1	0	1	0	0
A	0	1	0	1	1	0	0	0
D	1	0	1	1	0	0	0	0
B	1	1	0	0	0	0	1	0
E	0	1	1	1	0	1	0	0
E	0	1	1	1	0	1	0	0
F								

ALIGNING CODEWORDS, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

Payload: **0xDEADBEEF**

```
bit 76543210
    00110011
    10111110
    11111010
    11011101
    10000010
    10000111
    11000000
    10000010
```

	Top							Bot
D	1	0	1	0	0	0	0	1
E	0	1	1	1	0	1	0	0
A	0	1	0	1	1	0	0	0
D	1	0	1	1	0	0	0	0
B	1	1	0	0	0	0	1	0
E	0	1	1	1	0	1	0	0
E	0	1	1	1	0	1	0	0
F	1	1	1	1	1	1	1	1

ALIGNING CODEWORDS, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

Payload: **0xDEADBEEF**

```
bit 76543210
    00110011
    10111110
    11111010
    11011101
    10000010
    10000111
    11000000
    10000010
```

	Top							Bot
D	1	0	1	0	0	0	0	1
E	0	1	1	1	0	1	0	0
A	0	1	0	1	1	0	0	0
D	1	0	1	1	0	0	0	0
B	1	1	0	0	0	0	1	0
E	0	1	1	1	0	1	0	0
E	0	1	1	1	0	1	0	0
F	1	1	1	1	1	1	1	1

ALIGNING CODEWORDS, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

Payload: **0xDEADBEEF**

	Bin	(8,4) Parity
0xD	1101	1000
0xE	1110	0001
0xA	1010	1010
0xD	1101	1000
0xB	1011	0100
0xE	1110	0001
0xE	1110	0001
0xF	1111	1111

	Top							Bot
D	1	0	1	0	0	0	0	1
E	0	1	1	1	0	1	0	0
A	0	1	0	1	1	0	0	0
D	1	0	1	1	0	0	0	0
B	1	1	0	0	0	0	1	0
E	0	1	1	1	0	1	0	0
E	0	1	1	1	0	1	0	0
F	1	1	1	1	1	1	1	1

ALIGNING CODEWORDS, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

Payload: **0xDEADBEEF**

	Bin	(8,4) Parity
0xD	1101	1000
0xE	1110	0001
0xA	1010	1010
0xD	1101	1000
0xB	1011	0100
0xE	1110	0001
0xE	1110	0001
0xF	1111	1111

	Bot							Top
D	1	0	0	0	0	1	0	1
E	0	0	1	0	1	1	1	0
A	0	0	0	1	1	0	1	0
D	0	0	0	0	1	1	0	1
B	0	1	0	0	0	0	1	1
E	0	0	1	0	1	1	1	0
E	0	0	1	0	1	1	1	0
F	1	1	1	1	1	1	1	1

ALIGNING CODEWORDS, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

Payload: **0xDEADBEEF**

	Bin	(8,4) Parity
0xD	1101	1000
0xE	1110	0001
0xA	1010	1010
0xD	1101	1000
0xB	1011	0100
0xE	1110	0001
0xE	1110	0001
0xF	1111	1111

	Bot							Top
D	1	0	0	0	0	1	0	1
E	0	0	1	0	1	1	1	0
A	0	0	0	1	1	0	1	0
D	0	0	0	0	1	1	0	1
B	0	1	0	0	0	0	1	1
E	0	0	1	0	1	1	1	0
E	0	0	1	0	1	1	1	0
F	1	1	1	1	1	1	1	1

ALIGNING CODEWORDS, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

Payload: **0xDEADBEEF**

	Bin	(8,4) Parity
0xD	1101	1000
0xE	1110	0001
0xA	1010	1010
0xD	1101	1000
0xB	1011	0100
0xE	1110	0001
0xE	1110	0001
0xF	1111	1111

	p1	p2			d1	d2	d3	d4
	Bot							Top
D	1	0	0	0	0	1	0	1
E	0	0	1	0	1	1	1	0
A	0	0	0	1	1	0	1	0
D	0	0	0	0	1	1	0	1
B	0	1	0	0	0	0	1	1
E	0	0	1	0	1	1	1	0
E	0	0	1	0	1	1	1	0
F	1	1	1	1	1	1	1	1

ALIGNING CODEWORDS, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

Payload: **0xDEADBEEF**

	Bin	(8,4) Parity
0xD	1101	1000
0xE	1110	0001
0xA	1010	1010
0xD	1101	1000
0xB	1011	0100
0xE	1110	0001
0xE	1110	0001
0xF	1111	1111

	p1	p2	p4	p3	d1	d2	d3	d4
	Bot							Top
D	1	0	0	0	0	1	0	1
E	0	0	1	0	1	1	1	0
A	0	0	0	1	1	0	1	0
D	0	0	0	0	1	1	0	1
B	0	1	0	0	0	0	1	1
E	0	0	1	0	1	1	1	0
E	0	0	1	0	1	1	1	0
F	1	1	1	1	1	1	1	1

ALMOST THERE

APPLY FORWARD ERROR CORRECTION

- spreading_factor = 8, code_rate=4/8

Payload: **0xDEADBEEF**

	Bin	(8,4) Parity
0xD	1101	1000
0xE	1110	0001
0xA	1010	1010
0xD	1101	1000
0xB	1011	0100
0xE	1110	0001
0xE	1110	0001
0xF	1111	1111

	p1	p2	p4	p3	d1	d2	d3	d4
	Bot							Top
D	1	0	0	0	0	1	0	1
E	0	0	1	0	1	1	1	0
A	0	0	0	1	1	0	1	0
D	0	0	0	0	1	1	0	1
B	0	1	0	0	0	0	1	1
E	0	0	1	0	1	1	1	0
E	0	0	1	0	1	1	1	0
F	1	1	1	1	1	1	1	1

APPLY FORWARD ERROR CORRECTION

- spreading_factor = 8, code_rate=4/8

Payload: **0xDEADBEEF**

	Bin	(8,4) Parity
0xD	1101	1000
0xE	1110	0001
0xA	1010	1010
0xD	1101	1000
0xB	1011	0100
0xE	1110	0001
0xE	1110	0001
0xF	1111	1111

	p1	p2	p4	p3	d1	d2	d3	d4
	Bot							Top
D	1	0	0	0	1	1	0	1
E	0	0	1	0	1	1	1	0
A	1	0	0	1	1	0	1	0
D	1	0	0	0	1	1	0	1
B	0	1	0	0	1	0	1	1
E	0	0	1	0	1	1	1	0
E	0	0	1	0	1	1	1	0
F	1	1	1	1	1	1	1	1

APPLY FORWARD ERROR CORREC

- spreadin 8, cod e=4/8

load: ADBEEF

	Bi 4) Parity			p2	p	3	d 2	4
	110 1000	D		0	0		1	0 p
E	1110 001	E		0	1		1	1 0
	1010 010		1		0		1	1 0
	1101 00		1		0		1	0 1
0	1011 0		0				1 0	1 1
0x	110		0	0		0	1 1	1 0
0x	10	E	0	0	1	0	1 1	1 0
0	111 111	F	1	1	1	1	1 1	1 1

BOOM

THAT´S THE MODULATION

RED HERRINGS

- Decoding stages:
  1. Symbol “gray indexing”
  2. Data whitening
  3. Interleaving
  4. Forward Error Correction

Documentation:

European patent LIE

Data sheet LIE

European patent SUPER LIE

European patent

…data sheets actually ok

HOW DID THIS WORK?

- Decoding stages:

1. Symbol “gray indexing”	CONTROLLED
2. Data whitening	CONTROLLED
3. Interleaving	?????
4. Forward Error Correction	...pretty confident

Only 1 experimental variable!

LORA

PHY STRUCTURE

PHY HEADER

Sync Word & SFD
nPreamble Symbols
nHeader Symbols
Preamble
Header
CRC
(explicit mode only)
Payload
Payload CRC
CR = 4/8
CR = CodingRate
SF = SpreadingFactor

PHY HEADER MODE

- Implicit vs. Explicit
  - Explicit encodes modulation params into the header
  - Implicit assumes receiver knows them, uses header for data instead

Explicit Header Mode

This is the default mode of operation. Here the header provides information on the payload, namely:

- The payload length in bytes.
- The forward error correction code rate
- The presence of an optional 16-bits CRC for the payload.

- No problem, right? Let's derive it as before!

LOL!

DECIPHERING THE PHY HEADER

- Implicit and Explicit header modes use different whitening sequences
  - PHY header is obfuscated even if we know the implicit whitening sequence

- Let's see what we know

PHY HEADER MODULATION

Always sent at SF-2 and CR=4/8, even if header is implicit!

PHY DATA UNIT STRUCTURE

- First 8 symbols:
  - Spreading Factor == SF-2
  - Code Rate == 4/8
- Low Data Rate mode:
  - Remaining symbols also sent at Spreading Factor == SF-2

THAT'S THE PHY

INTRODUCING

GR-LORA

GR-LORA

- Out-of-tree GNU Radio module
- Open-source implementation of the PHY

MOTIVATION

- Existing interfaces to LoRa are at Layer 2 and above
  - IC interfaces and data sheets
  - LoRaMAC // LoRaWAN standards
- PHY layer security can't be taken for granted

802.15.4 PHY LAYER EXPLOITS

- Packet-in-packet – Travis Goodspeed, Sergey Bratus, Ricky Melgares, Rebecca Shapiro, Ryan Speers
  - Layer 7->1 compromise
- WIDS evasion – Ira Ray Jenkins, Rebecca Shapiro, Sergey Bratus, Travis Goodspeed, Ryan Speers
  - Evading IDS // network monitors by fingerprinting receiver PHYs and crafting packets for selective evasion

802.15.4 PHY LAYER EXPLOITS

- Packet-in-packet – Travis Goodspeed, Sergey Bratus, Ricky Melgares, Rebecca Shapiro, Ryan Speers
  - Layer 7->1 compromise
- IDS evasion – Ray Jenkins, Rebecca Shapiro, Sergey Bratus, Travis Goodspeed, Ryan Speers
  - Evading IDS/ network monitors by fingerprinting receiver PHYs and crafting packets for selective evasion

PHYS MATTER

GNURADIO/POTHOS PRIOR ART

- Josh Blum's LoRa-SDR for POTHOS
  - Implements a LoRa-like modulation
  - Encoding/decoding is incorrect – follows reference documents, which differ drastically from reality
- rpp0's gr-lora

GR-LORA ARCHITECTURE

- Modulation and encoding stages are modeled as separate blocks
  - Allows for modularity and experimentation
- Asynchronous PDU interface between blocks
  - Super simple socket interface

DEMODULATOR // DECODER

- Implements the decoding algorithm we just outlined
  - Dechirping, stacked FFTs, etc.

MODULATOR // ENCODER

- Modulator uses a pre-calculated c32 chirp LUT and a phase accumulator
  - More efficient than FM or IFFT

GR-LORA SOURCE

github.com/BastilleResearch/gr-lora

LIVE DEMO

SCENARIO

- Let's sniff some traffic!
- Transmitter
  - Adafruit Feather M0 w/ hardware-defined LoRa module
- Receiver
  - gr-lora
  - fed by Ettus USRP B210

GR-LORA TODOS

- Explicit PHY header, whitening, and CRC
- Alternate demod strategies, sync improvements

- Implement upper layers (LoRaWAN, LoRaHAM)

ADDITIONAL RESOURCES

- PoC||GTFO 0x13: “Reverse Engineering LoRa’s PHY”
- “So You Want to Hack Radios” with Marc Newlin

TO CONCLUDE

- LPWANs have momentum and are proliferating
- RF stacks are becoming more diverse
  - Wireless is not just WiFi anymore
- Shown how to go from obscure RF –> bits
- Added a new tool to the RF security researcher's arsenal

ACKNOWLEDGEMENTS

- Balint Seeber, Bastille Threat Research Team
- Josh Blum, hexameron, Bertrik Sikken, open source contributors
- Thomas Telkamp and Chuck Swiger for donating IQ
- Chaos Computer Club for organizing!

DANKE SCHÖN

github.com/BastilleResearch/gr-lora

matt@Bastille.net
@embeddedsec

QUESTIONS?

github.com/BastilleResearch/gr-lora

matt@Bastille.net
@embeddedsec