osmo-gmr: What's up with sat-phones ?

Piecing together the missing bits

Sylvain Munaut

31C3, December 27th, 2014

Introduction

Outline

About the speaker

- Linux and free software "geek" since 1999
- M.Sc. in C.S. + some E.E.
- General orientation towards low level
  - Embedded, Kernel, Drivers and such.
  - Hardware (Digital stuff, FPGA or RF)
- Interest in various telecom and SDR projects for several years
  - Osmocom projects (OpenBSC, Osmocom-BB, ...)
  - Airprobe, OpenBTS ...
  - In my spare time

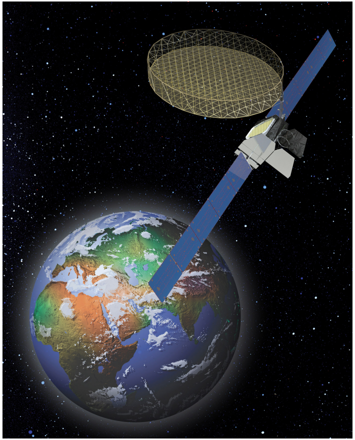
GMR Introduction

What is GMR ?

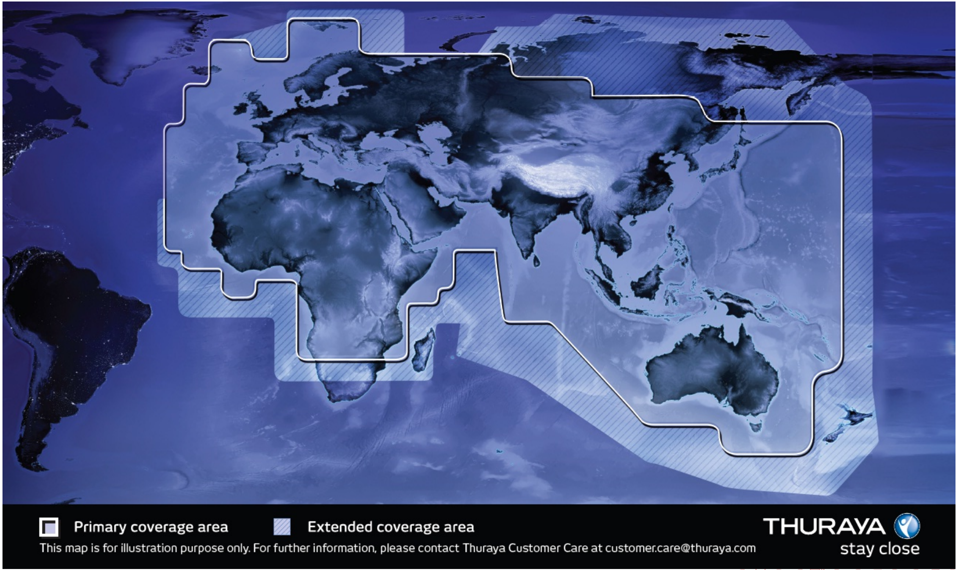
- "GEO-Mobile Radio Interface"
  (GEO stands for Geostationary Earth Orbit)
- ETSI standard for satellite phones
- Heavily based on GSM
- Multiple standards :
  - GMR-1 (ETSI TS 101 376)
    - GMR-1 *(the one described in this talk)*
    - GmPRS
    - GMR-1 3G
  - GMR-2 (ETSI TS 101 377)

Deployment

- GMR-1
  - Thuraya
    - Thuraya 2 (44E) and Thuraya 3 (98.5E)
    - Main focus of our attention so far
  - SkyTerra ?
  - TerreStar ?
  - ICO
  - (Inmarsat R-BGAN)
  - Solaris Mobile *(future)*
- GMR-2
  - Inmarsat "IsatPhone"
  - ACes



Deployment
Thuraya



Comparison to GSM
Features

- New names
  - BTS $\rightarrow$ GTS, BSC $\rightarrow$ GSC, BSS $\rightarrow$ GSS, ...
  - MS $\rightarrow$ MES(-MS)
- New Specialized features
  - Terminal-to-Terminal calls
  - High Penetration Alerting (HPA)
- Tight links to GPS
  - Almanac and Ephemeris sent by the satellite
  - Position reported in RACH (Channel Request)
- New speech codec: AMBE
- New cipher

Comparison to GSM
Protocol Stack

- Layer 0/1: Completely different
  - Different bursts and TDMA multiplex / multi-frame
  - Different modulation
  - More channels types
- Layer 2: LAPSat vs LAPDm
  - Both simplified version of LAPD
  - Shorter header
  - k=16 window size for outstanding unacknowledged segments
- Layer 3:
  - RR different
  - MM/CM common
- Same core network
- Packet Data:
  - RLC/MAC different
  - LLC and above common/shared

Previous work

osmo-gmr

- Hardware setup
  - Antennas, LNAs, filters, SDR receivers
- SDR processing
  - Channelization, Demodulation
- Channel coding
  - Channel types, Interleaving, Viterbi, CRC
- Demo receive app
  - Synchronization, minimal higher level support
- Wireshark dissector
- Two big missing pieces for basic RF to Audio chain
  - Voice Codec
  - Cipher

Refer to the 28C3 talk for more details

GMR-1 Speech codec

GMR-1 Speech codec

The problem

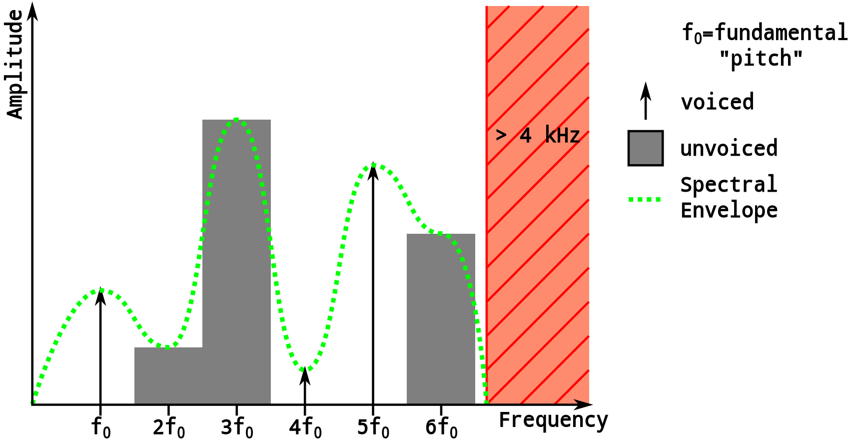
- AMBE: Advanced Multi-Band Excitation
- Not documented in the standard
  - Barely a high level description
  - No reference code
- Proprietary codec by *DVSI Inc.*
  - Not supported by their "cheap" hardware USB decoder
  - Cheapest hardware is the NET-2000 appliance (2kEUR)
- But :
  - `mbelib`: Code for other documented IMBE/AMBE variants (P25)
  - Implemented in SO2510 phone DSP (`TI C55x`)

AMBE Codec
Description

- Highly specialized for voice (vocoder)
- Divides speech in small segments
  - For GMR-1: 20 ms frames subdivided into co-quantized 10 ms sub-frames
- Represent each speech (sub)frame as a set of parameters
  - f_0 : Fundamental frequency *(pitch)*
  - G : Gain *(volume)*
  - Voiced / Unvoiced decision (per band)
  - Spectral Magnitudes
- Decoding can be summarized as 3 steps:
  - **Unpacking**: Unpack the raw frame bits into quantized parameters
  - **De-Quantization**: From quantized parameters to actual values
  - **Synthesis**: From the parameters set to actual audio

AMBE Codec

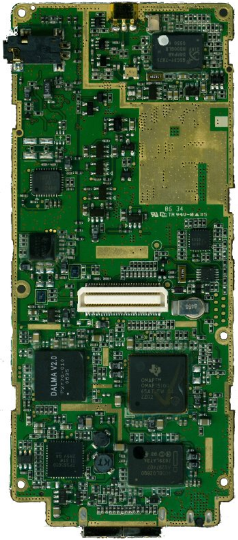
Synthesis



AMBE Reversing

DSP Code analysis

- Target: SO-2510 phone
- Codec has to be in the DSP, nowhere else it could be !
- DSP firmware extracted from firmware update package
  - Supported by IDA
- But where ?
  - 250k binary blob
  - No strings
  - Obscure `TI C55x` assembly
- Dieter Spaar to the rescue !
  - Identified entry points for encode/decode functions
  - Look for Audio DMA / Interrupts
  - Search for constants
  - Stack Switching



AMBE Reversing

Simulator

- TI Code Composer Studio - Simulator
  - Accurately simulates supported DSP
  - Arbitrary memory layout
  - fread()/fwrite() from host
  - Tracing of all memory access
  - Windows only :(
- Use the original firmware to decompress audio for us
  - DSP dump converted to a valid COFF .OBJ file for linking
  - Custom linker script
  - Simple main() that fread() frames and fwrite() audio
- Success !
  - It took quite a few tries, lots of traps
  - But it works and we get audio out
  - Slow (not real-time) and not practical though

AMBE Reversing

Hardware

- Real HW would be faster and more convenient. But :
  - Code has to run at the physical address it has been linked for
  - OMAP has a DSP MMU, but standalone DSP don't
  - Need a cheap board with a compatible memory map
- Dieter found one with SDRAM where needed and Ethernet
  - Success ! About 16x faster than real-time
  - SDRAM is not fast, relocate some data tables to SRAM
- I indented to buy the same board
  - But in my haste ... I ordered the wrong one ... ***facepalm***
  - No SDRAM, more SRAM, but at the wrong physical address
  - Easy, just relocate the code ! Can't be that hard, right ?
  - Use `IDAPython` + simulator trace mode
  - Success !

AMBE Reversing
Software

- Hardware USB decoder is nice, but not enough
- Decompression process:
  - **Unpacking**
    - Early, simple bit manipulation, easy to follow
  - **Dequantization**
    - Easily 95% of the work
    - Hard to follow fixed point math in DSP assembly
  - **Synthesis**
    - Started by just re-using `mbelib` code
    - Then rewrote using P25 specs and some guessing
- Resulting PoC/reference code in GIT
  - Not same audio quality as the original, but perfectly intelligible

```
amar     *+AR0(#857h)
call     sub_103540
mov      *AR6(#100h), AR1
btst     @0, AR1, TC1
mov      #0, T2
xccpart !TC1
bcc      loc_10B148, !TC1
amar     *AR7, XAR0
amar     *AR5, XAR1
|| mov T2, T0
call     sub_109EC8
mov      T0, T2


amar     *AR5, XAR3
amar     *AR5, XAR4
amar     *+AR3(#684h)
amar     *AR3, XAR2
amar     *+AR4(#684h)
amar     *AR5, XAR3
amar     *+AR3(#584h)
rpt      #0FFh
mov      *AR4+, *AR3+
mov      *AR5(#785h), AR1
mov      AR1, *AR5(#784h)
amar     *SP(#var_0), XAR3
|| rpt #0FFh
mov      *AR3+, *AR2+
add      #102h, mmap(@SP)
mov      T3, *AR5(#785h)
popboth XAR6
```

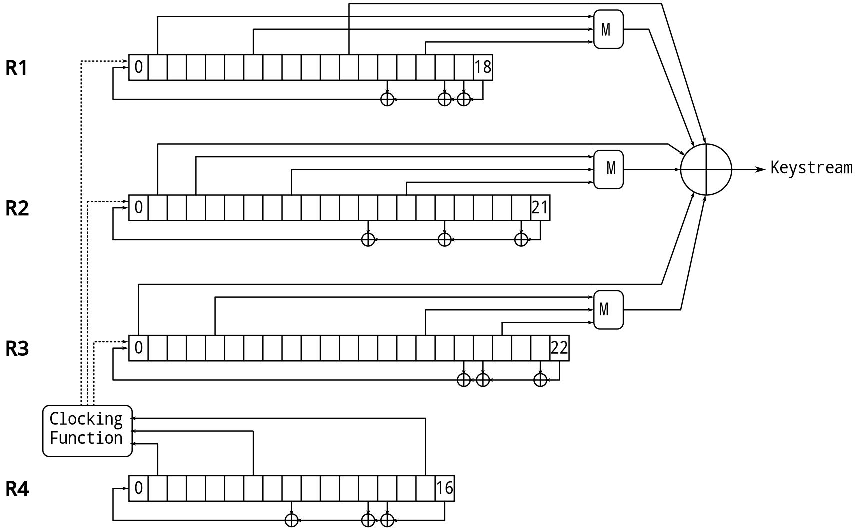
GMR-1 Cipher

Extraction from DSP

- Perfomed by a team at Bochum university (RUB)
  - Led by Benedikt Driessen
- Published beginning 2012
- Validated against on-the-air data using `osmo-gmr`
- Process:
  - Extract DSP image by running the ARM DSP download code in QEMU
  - Analyze DSP image using heuristics
  - (But really, if you look for `XOR` for a few minutes, that works too)
  - Refer to `http://gmr.crypto.rub.de` for more details and the papers
- Also includes a cipher-text only attack

A5-GMR-1

Structure



A5-GMR-1

Description

- Based off GSM's A5/2
  - Feedback, output and clocking taps changed
  - Output function and Initialization tweaked
  - But globally the same structure
- 4 maximal length LFSRs (R1=19, R2=22, R3=23 and R4=17)
- Initialization:
  - Mix the key and frame-number with a linear function
  - Clock all LFSRs 64 times while mixing-in the result from above
  - Force the MSB of each LFSRs to 1
  - Clock all LFSRs 250 times
- Bitstream generation:
  - Clock the cipher as many times as needed to generate enough bits
  - R4 is tapped into a clocking function that drives R1,R2,R3 clocking
  - R1, R2 and R3 are tapped and combined into an output bit

Linear algebra over $GF(2)$ quick refresher

- $GF(2)$ is a fancy term for *binary*
  - Addition is the logic `XOR`
  - Multiplication is the logic `AND`
- Linear: each term is a constant or a variable/unknown multiplied by a constant
- Linear equations systems :
  - Can be represented as matrix operations

$$\begin{matrix} a \cdot x + b \cdot y = c \\ d \cdot x + e \cdot y = f \end{matrix} \iff \begin{pmatrix} a & b \\ d & e \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} c \\ f \end{pmatrix}$$

  - Can be solved efficiently
    - Number of independent equations vs number of unknowns determines the number of possible solutions (over/under determination)
- Other linear operations can also be rewritten as matrix operations
  - State change of a LFSR for instance: $S_{t+1} = A \cdot S_t$
  - They can also be combined: $S_{t+n} = A^n \cdot S_t$
  - If they add redudancy (like FEC), a parity-check matrix can check the result

RUB attack

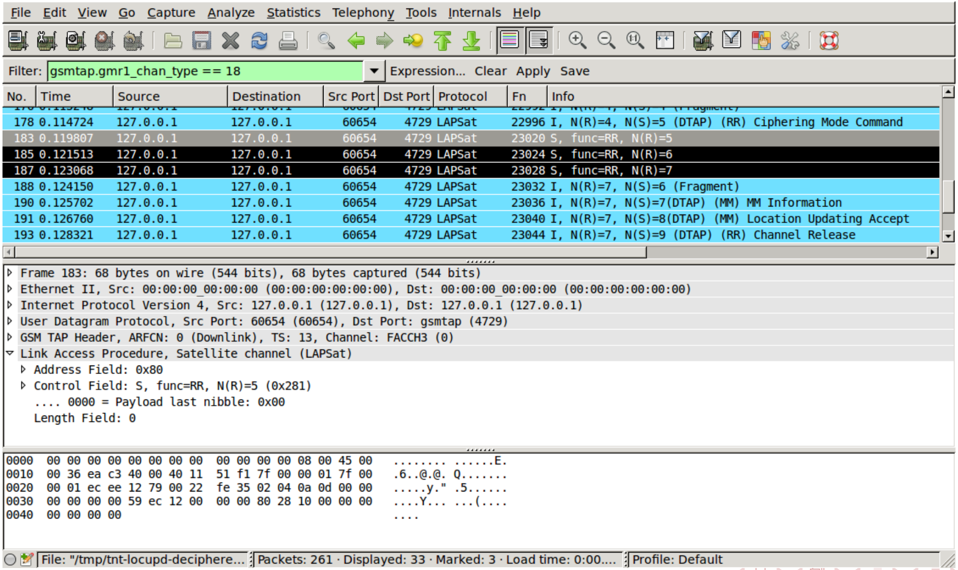
- Cipher-text only
- Targets `TCH3` traffic/voice frame
- Based off previous A5/2 work
  - "Instant Ciphertext-Only Cryptanalysis of GSM encrypted communication" by E.Barkan, E.Biham, and N.Keller
  - Tweaked for GMR-1 and its `TCH3` frames
  - Adds more "guessed" bits in each register to reduce the unknowns
- Results :
  - 350 Gb of off-line data
  - 32 `TCH3` frames
  - About 40 minutes on-line phase

A better attack
Overview

- Based on the same A5/2 GSM attack
  - Don't do anything fancy, just tweak for A5-GMR-1
- Both known-plaintext and ciphertext-only variant
- Targets `FACCH3` control frames instead of `TCH3` voice frames
- `FACCH3` advantages :
  - Simpler modulation and better training sequence → less bit-errors
  - Predictable plaintext → known-plaintext attacks
  - Much more redudancy (more FEC) → less bursts needed for ciphertext-only attacks
  - Used to negotiate `TCH6`/`TCH9` channels → attack works for CSD/Fax

A better attack

Known plaintext



A better attack

Plaintext attack

- Goal is to describe cipher as a linear operation: $A \cdot x = b$
  - A = matrix describing cipher, x = internal state and b = cipher stream
  - Each row of A and b is a bit of the output
- Internal cipher state dependency on FN and Kc is linear
  - Possible to combine equations from different bursts at different FN
  - Can recover Kc from the state
- Non-linear elements:
  - Majority function: $\mathcal{M}(a, b, c) = a + b.c$
    - Introduces quadratic terms
    - Linearize by adding one new unknown for every possible quadratic term
    - 594 new unknowns
  - Irregular clocking depending on R4 value
    - R4 is 17 bits but one is forced to '1' at init. Small enough for brute force !
    - Assume a given value for R4
    - Repeat 65536 times

A better attack

R4 quick scan

- In $A_n \cdot x = b$, some equations are redundant
- We can get a parity-check matrix H_n such that $H_n \cdot b = \mathbf{0}$
- Those 65536 H_n matrices can be precomputed offline
- With a single matrix-multiply we can check if a given R4 value is even a possibility
  - If result is non-zero, we can skip that R4 value
  - If result is zero, then we try to solve the system
  - In practice, only a few R4 value ever matches

A better attack

Ciphertext only

- Channel coding operation: $m = d \cdot G + g$
- Let H be the parity-check matrix so that $H \cdot (m + g) = \mathbf{0}$
- Encryption operation: $y = m + b$
- H can be used to derive equations from the ciphertext y:

$$\begin{aligned} H \cdot (y + g) &= H \cdot (m + b + g) \\ &= H \cdot b + \underbrace{H \cdot (m + g)}_{\mathbf{0}} \\ &= H \cdot A \cdot x \end{aligned}$$

- The same R4 quick-scan technique can also be used here
- To get enough equations for a unique solutions, multiple frames are needed

A better attack

Results

- Known-plaintext variant
  - Requires between 4 and 8 bursts depending on alignement
  - Space: 50 Mb
  - Time: 500 ms
- Ciphertext-only variant
  - Requires 8 consecutive bursts belonging to 2 FACCH3 L2 frames
  - Space: 5 Gb
  - Time: 1 s

Final words

Future

- C-band
- Packet Data (GmPRS)
- Upper layers implementation
- CSN.1 and 04.008 code generators
- TX side

Help welcome :)

Other satellite phone systems

- We choose Thuraya because :
  - Visible from Europe
  - Cheapest sat phone on ebay
  - Specifications mostly available
- Don't think other are better without proof
  - Availability of commerical intercepts tend to say otherwise

Thanks

Thanks to anyone who contributed to this projects and related ones. Most notably:

- Dimitri "horizon" Stolnikov
- Dieter Spaar
- RUB team

Further reading

- GMR-1 in general
  - OsmocomGMR http://gmr.osmocom.org/
  - 28C3 talk http://gmr.osmocom.org/trac/blog/28c3-recording
  - GMR1 Specs http://www.etsi.org/standards-search
  - GSM Specs http://webapp.etsi.org/key/queryform.asp
- AMBE Codec
  - DVSI Inc. http://www.dvsinc.com/
- GMR-1 Cipher
  - RUB GMR page http://gmr.crypto.rub.de/
  - Paper http://cryptome.org/gsm-crack-bbk.pdf