Free Software for GSM cellular telephony

OpenBSC, OsmoBTS, OsmoSGSN, OpenGGSN

Harald Welte

gnumonks.org
osmocom.org
sysmocom.de

DORS/CLUC, June 2014, Zagreb

Outline

About the speaker

- Using + playing with GNU/Linux since 1994
- Kernel / bootloader / driver / firmware development since 1999
- IT security expert, focus on network protocol security
- Core developer of Linux packet filter netfilter/iptables
- Trained as Electrical Engineer
- Always looking for interesting protocols (RFID, DECT, GSM)

Success of Free Software
depending on area of computing

- Free Software has proven to be successful in many areas of computing
  - Operating Systems (GNU/Linux)
  - Internet Servers (Apache, Sendmail, Exim, Cyrus, ...)
  - Desktop Computers (gnome, KDE, Firefox, LibreOffice, ...)
  - Mobile Devices
  - Embedded network devices (Router, Firewall, NAT, WiFi-AP)
- There are more areas to computing that people tend to forget. Examples in the communications area:
  - Cellular telephony networks (GSM, 3G, LTE)
  - Professional Mobile Radio (TETRA, TETRAPOL)
  - Cordless telephones (DECT)

Free specs / Free implementations

- Observation
  - Both GSM/3G and TCP/IP protocol specs are publicly available
  - The Internet protocol stack (Ethernet/Wifi/TCP/IP) receives lots of scrutiny
  - GSM networks are as widely deployed as the Internet
  - Yet, GSM/3G protocols receive no such scrutiny!
- There are reasons for that:
  - GSM industry is extremely closed (and closed-minded)
  - Only about 4 proprietary protocol stack implementations
  - GSM chip set makers never release any hardware documentation

The closed GSM industry
Handset manufacturing side

- Only very few companies build GSM/3.5G baseband chips today
  - Those companies buy the operating system kernel and the protocol stack from third parties
- Only very few handset makers are large enough to become a customer
  - Even they only get limited access to hardware documentation
  - Even they never really get access to the firmware source

The closed GSM industry

Network manufacturing side

- Only very few companies build GSM network equipment
  - Basically only Ericsson, Nokia-Siemens, Alcatel-Lucent and Huawei
  - Exception: Small equipment manufacturers for picocell / nanocell / femtocells / measurement devices and law enforcement equipment
- Only operators buy equipment from them
- Since the quantities are low, the prices are extremely high
  - e.g. for a BTS, easily 10-40k EUR
  - minimal network using standard components definitely in the 100,000s of EUR range

The closed GSM industry
Operator side

From my experience with Operators (prove me wrong!)

- Operators are mainly finance + marketing today
- Many operators outsources
  - Network servicing / deployment, even planning
  - Other aspects of business like Billing
- Operator just knows the closed equipment as shipped by manufacturer
- Very few people at an operator have knowledge of the protocol beyond what's needed for operations and maintenance

The closed GSM industry
Security implications

The security implications of the closed GSM industry are:

- Almost no people who have detailed technical knowledge outside the protocol stack or GSM network equipment manufacturers
- No independent research on protocol-level security
  - If there's security research at all, then only theoretical (like the A5/2 and A5/1 cryptanalysis)
  - Or on application level (e.g. mobile malware)
- No free software protocol implementations
  - which are key for making more people learn about the protocols
  - which enable quick prototyping/testing by modifying existing code

Security analysis of GSM

How would you get started?

If you were to start with GSM protocol level security analysis, where and how would you start?

- On the handset side?
  - Difficult since GSM firmware and protocol stacks are closed and proprietary
  - Even if you want to write your own protocol stack, the layer 1 hardware and signal processing is closed and undocumented, too
  - Known attempts
    - The TSM30 project as part of the THC GSM project
    - MADos, an alternative OS for Nokia DTC3 phones
  - none of those projects successful so far

Security analysis of GSM

How would you get started?

If you were to start with GSM protocol level security analysis, where and how would you start?

- On the network side?
  - Difficult since equipment is not easily available and normally extremely expensive
  - However, network is very modular and has many standardized/documented interfaces
  - Thus, if equipment is available, much easier/faster progress
  - Also, using SDR (software defined radio) approach, special-purpose / closed hardware can be avoided

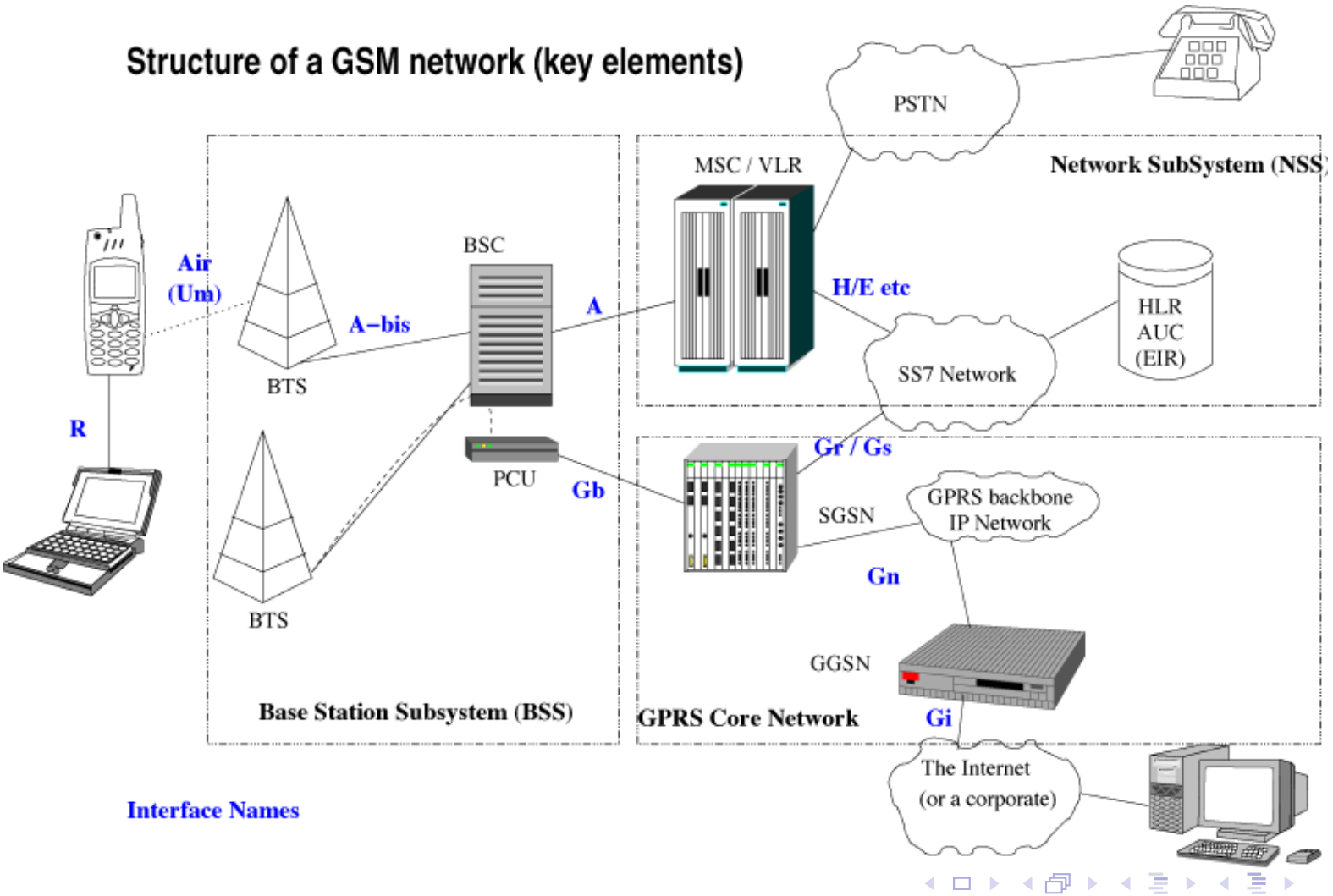
Security analysis of GSM
The bootstrapping process

- Read GSM specs day and night (> 1000 PDF documents)
- Gradually grow knowledge about the protocols
  - OpenBSC: Obtain actual GSM network equipment (BTS)
  - OpenBTS: Develop SDR based GSM Um Layer 1
- Try to get actual protocol traces as examples
- Start a complete protocol stack implementation from scratch
- Finally, go and play with GSM protocol security

Security analysis of GSM
The bootstrapping process

- Start to read GSM specs (> 1000 PDF documents!)
- Gradually grow knowledge about the protocols
- Obtain actual GSM network equipment (BTS)
- Try to get actual protocol traces as examples
- Start a complete protocol stack implementation from scratch
- Finally, go and play with GSM protocol security

The GSM network



GSM network components

- The BSS (Base Station Subsystem)
  - MS (Mobile Station): Your phone
  - BTS (Base Transceiver Station): The *cell tower*
  - BSC (Base Station Controller): Controlling up to hundreds of BTS
- The NSS (Network Sub System)
  - MSC (Mobile Switching Center): The central switch
  - HLR (Home Location Register): Database of subscribers
  - AUC (Authentication Center): Database of authentication keys
  - VLR (Visitor Location Register): For roaming users
  - EIR (Equipment Identity Register): To block stolen phones

GSM network interfaces

- Um: Interface between MS and BTS
  - the only interface that is specified over radio
- A-bis: Interface between BTS and BSC
- A: Interface between BSC and MSC
- B: Interface between MSC and other MSC

GSM networks are a prime example of an asymmetric distributed network, very different from the end-to-end transparent IP network.

GSM network protocols
On the Um interface

- Layer 1: Radio Layer, TS 04.04
- Layer 2: LAPDm, TS 04.06
- Layer 3: Radio Resource, Mobility Management, Call Control: TS 04.08
- Layer 4+: for USSD, SMS, LCS, ...

GSM network protocols
On the A-bis interface

- Layer 1: Typically E1 line, TS 08.54
- Layer 2: A variant of ISDN LAPD with fixed TEI's, TS 08.56
- Layer 3: OML (Organization and Maintenance Layer, TS 12.21)
- Layer 3: RSL (Radio Signalling Link, TS 08.58)
- Layer 4+: transparent messages that are sent to the MS via Um

OpenBSC software

OpenBSC is a Open Source implementation of (not only) the BSC features of a GSM network.

- Support A-bis interface over E1 and IP
- Support for BTS vendor/model is modular
- Multiple BTS models/vendors can be mixed!
- Can work as a *pure BSC* or as a full *network in a box*
- Supports mobility management, authentication, intra-BSC hand-over, SMS, voice calls (FR/EFR/AMR)
- GPRS + EDGE support if combined with OsmoSGSN and OpenGGSN

OpenBSC

- Supports various BTS brands/models (Siemens BS-11, Ericsson RBS2000, Nokia MetroSite, ip.access nanoBTS, sysmocom sysmoBTS)
- Has classic 2G signalling, voice and SMS support
- Implements various GSM protocols like
  - A-bis RSL (TS 08.58) and OML (TS 12.21)
  - TS 04.08 Radio Resource, Mobility Management, Call Control
  - TS 04.11 Short Message Service
- Telnet console with Cisco-style interface

OpenBSC software architecture

- Implemented in pure C, similarities to Linux kernel
  - Linked List handling, Timer API, coding style
- Single-threaded event-loop / state machine design
- Telnet based command line interface *Cisco-style*
- Input driver abstraction (mISDN, Abis-over-IP)

OpenBSC: GSM network protocols

The A-bis interface

Layer 1	Typically E1 line, TS 08.54
Layer 2	A variant of ISDN LAPD with fixed TEI's, TS 08.56
Layer 3	OML (Organization and Maintenance Layer, TS 12.21)
Layer 3	RSL (Radio Signalling Link, TS 08.58)
Layer 4+	transparent messages that are sent to the MS via Um

OpenBSC: How it all started

- In 2006, I bought a Siemens BS-11 microBTS on eBay
  - This is GSM900 BTS with 2 TRX at 2W output power (each)
  - A 48kg monster with attached antenna
  - 200W power consumption, passive cooling
  - E1 physical interface
- I didn't have much time at the time (day job at Openmoko)
- Started to read up on GSM specs whenever I could
- Bought a HFC-E1 based PCI E1 controller, has mISDN kernel support
- Found somebody in the GSM industry who provided protocol traces

OpenBSC: Timeline

- November 2008: Dieter+Harald started the development of OpenBSC
- December 2008: we did a first demo at 25C3
- January 2009: we had full voice call support
- Q1/2009: Add support for ip.access nanoBTS
- June 2009: I started with actual security related stuff
- August 2009: We had the first field test with 2BTS and > 860 phones
- Q1/2010: The first 25 OpenBSC instances running in a commercial network

OpenBSC: Field Test at HAR2009

OpenBSC in NITB mode

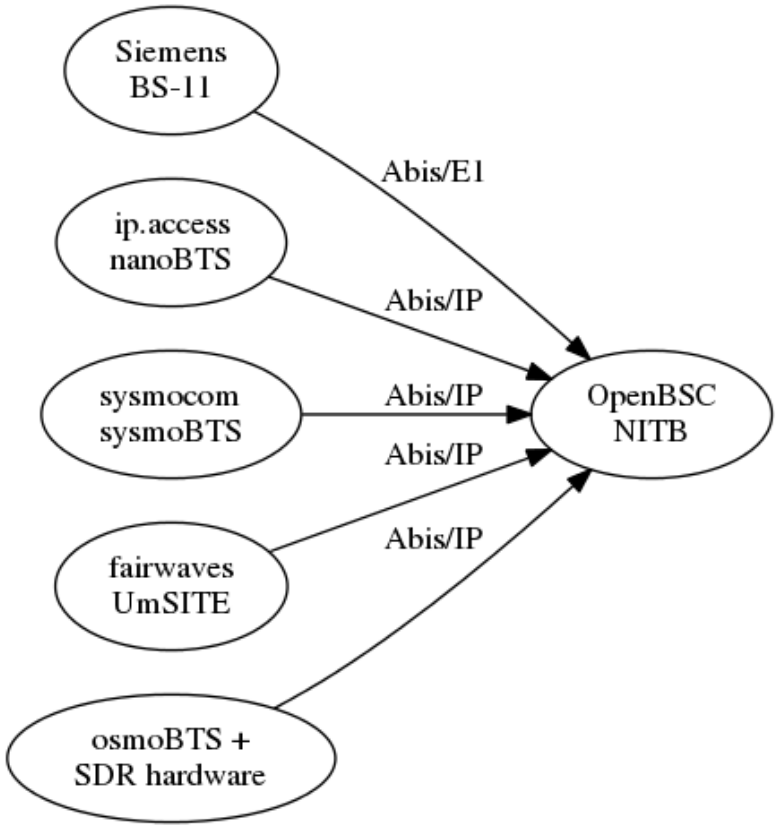
Network In a Box Mode

The `osmo-nitb` program

- implements the A-bis interface towards any number of BTS
- provides most typical features of a GSM network in one software
- no need for MSC, AuC, HLR, VLR, EIR, ...
  - HLR/VLR as SQLite3 table
  - Authentication + Ciphering support
  - GSM voice calls, MO/MT SMS
  - Hand-over between all BTS
  - Multiple Location Areas within one BSC

OpenBSC in NITB mode

Network In a Box Mode



OpenBSC NITB features

OpenBSC NITB features

- Run a small GSM network with 1-n BTS and OpenBSC
- No need for MSC/HLR/AUC/...
- No need for your own SIM cards (unless crypto/auth rqd)
- Establish signalling and voice channels
- Make incoming and outgoing voice calls between phones
- Send/receive SMS between phones
- Connect to ISDN PBX or public ISDN via Linux Call Router

OpenBSC in NITB mode

Network In a Box Mode

The `osmo-nitb` program

- does not implement any other GSM interfaces apart from A-bis
- no SS7 / TCAP / MAP based protocols
- no integration (roaming) with existing traditional GSM networks
- wired telephony interfacing with ISDN PBX `lcr` (Linux Call Router)
- Has been tested with up to 800 subscribers on 5 BTS
- Intended for R&D use or private PBX systems

osmo-nitb LCR integration

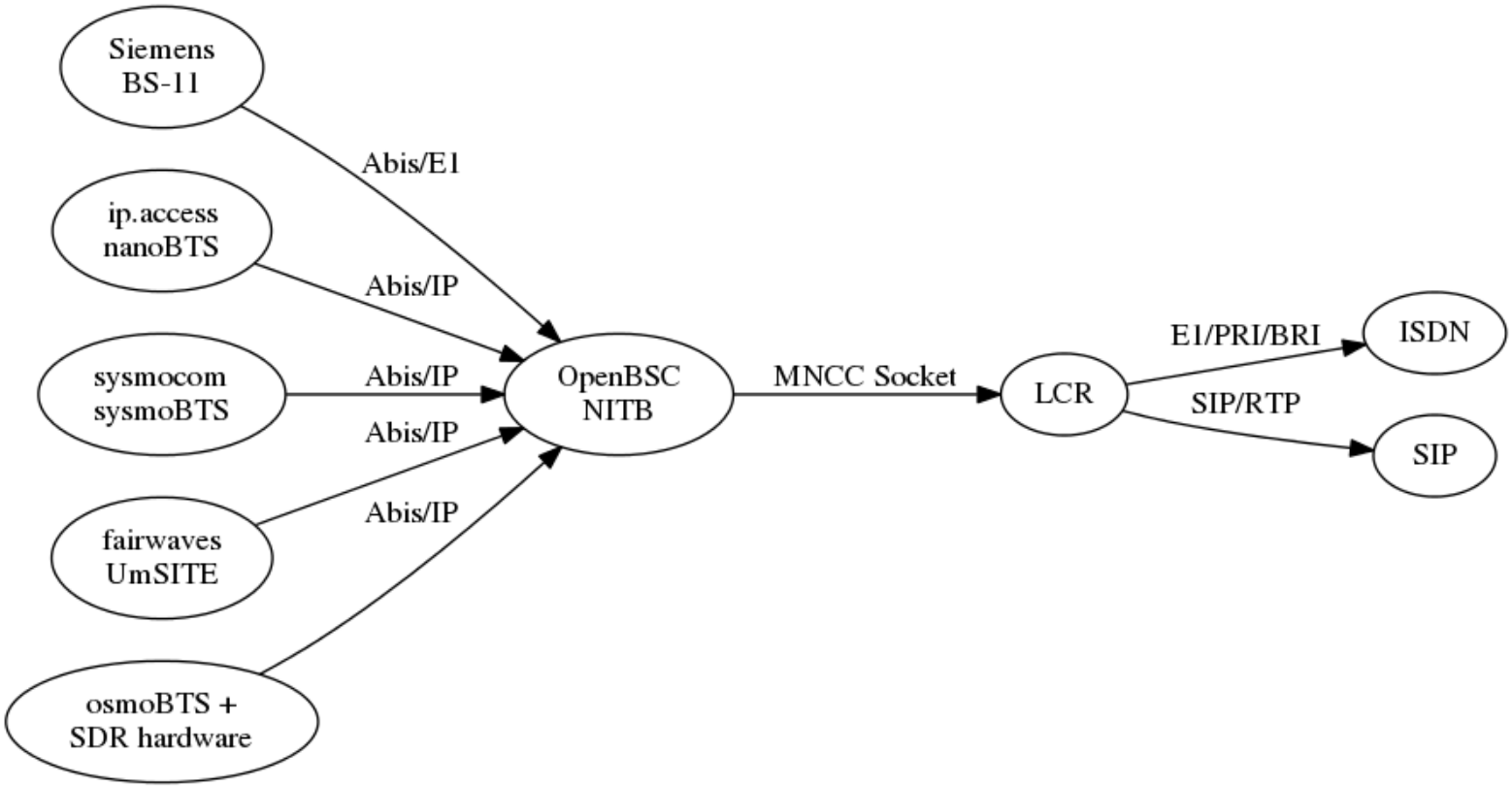
Interfacing with wired telephony

OpenBSC (NITB mode) can be connected to Linux Call Router (`lcr`)

- osmo-nitb exposes a MNCC interface (on unix domain socket)
- lcr attachs to that MNCC interface
- All call control inside osmo-nitb is disabled
- Dialling plan, etc. is now configured in `lcr` like for any other wired phones
- lcr supports VoIP (SIP), E1 (ISDN) and other interfaces

osmo-nitb LCR integration

Interfacing with wired telephony

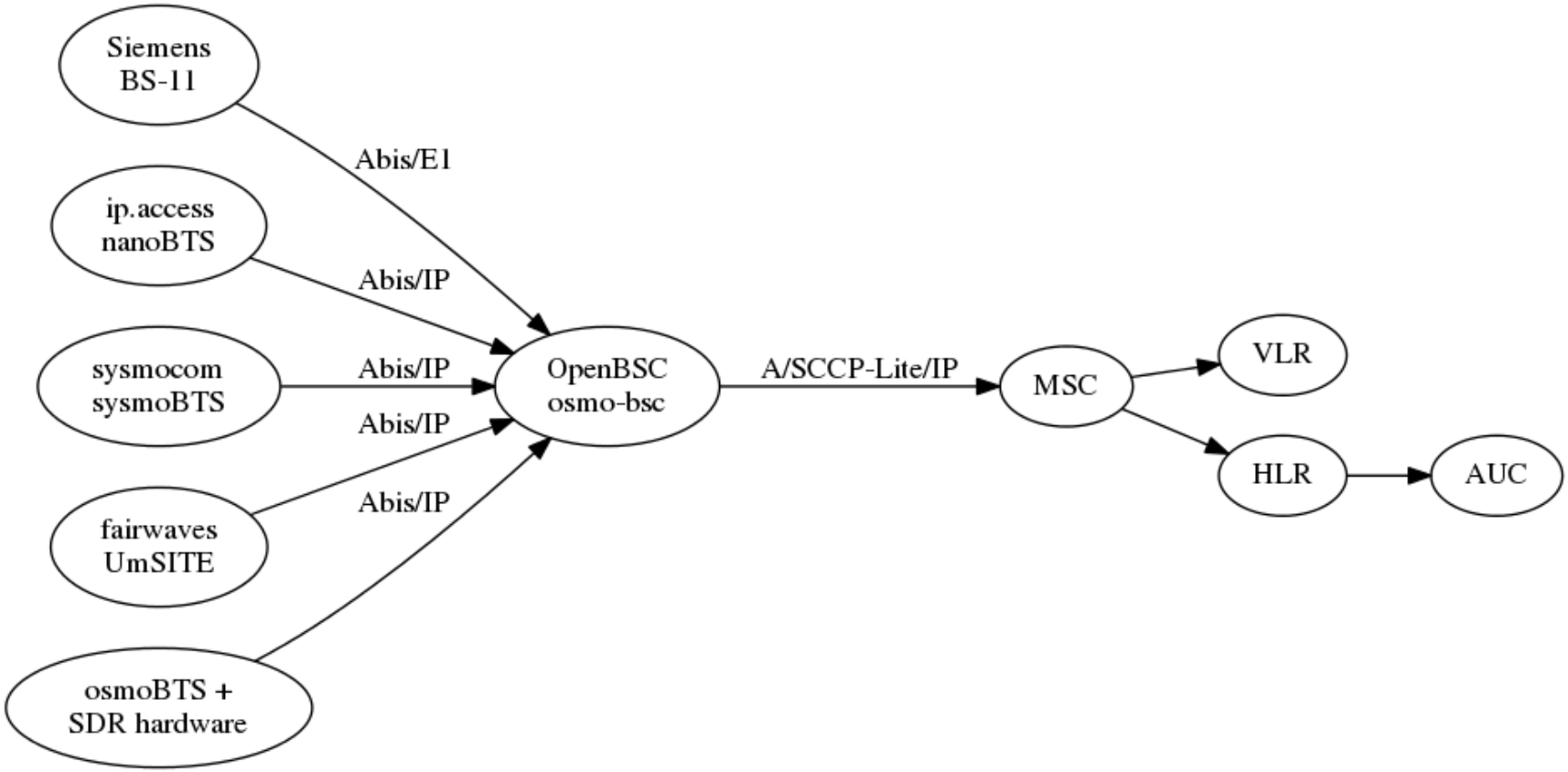


OpenBSC in BSC-only mode

The `osmo-bsc` program

- behaves like a classic GSM BSC
- uses SCCP-Lite (ip.access multipex) to any SoftMSC like ADC
- used in production/commercial deployments (75 BSCs)
- mainly intended to replace proprietary BSC in traditional GSM networks

OpenBSC in BSC-only mode



GPRS and OpenBSC

- The BSC doesn't really do anything related to GPRS
- GPRS implemented in separate SGSN and GGSN nodes
- GPRS uses its own Gb interface to RAN, independent of A-bis
- OpenBSC can configure the nanoBTS for GPRS+EDGE support via OML
- Actual SGSN and GGSN implemented as OsmoSGSN and OpenGGSN programs

OsmoSGSN

The Osmocom SGSN program implements

- basic/minimal SGSN functionality
- the Gb interface (NS/BSSGP/LLC/SNDCP)
- mobility management, session management

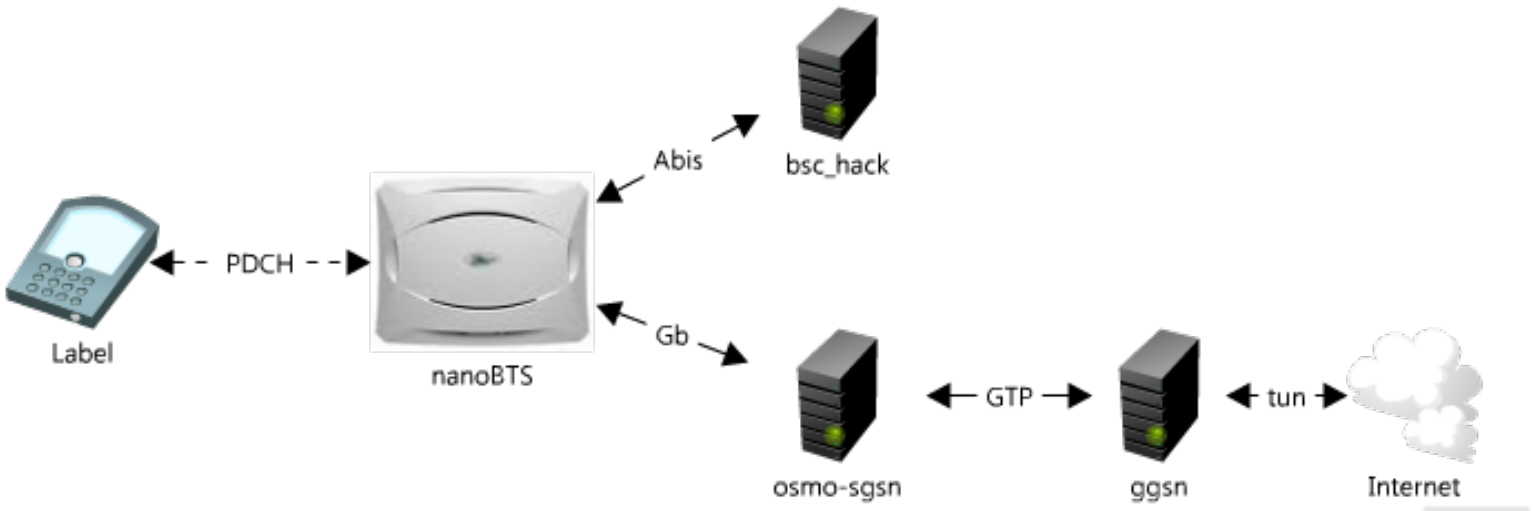
It's a work in progress, many missing features

- no HLR integration yet
- no paging coordination with MSC/BSC
- no encryption support yet

OpenGGSN

- GPL licensed Linux program implementing GGSN node
- Implements GTP-U protocol between SGSN and GGSN
- User-configurable range/pool of IPv4 addresses for MS
- Uses `tun` device for terminating IP tunnel from MS
- provides GTP implementation as libgtp
- Experimental patches for IPv6 support

OpenBSC and OsmoSGSN based network



Summary
What we've learned

- The GSM industry is making security analysis very difficult
- It is well-known that the security level of the GSM stacks is very low
- We now have multiple solutions for sending arbitrary protocol data
  - From a rogue network to phones (OpenBSC, OpenBTS)
  - From a FOSS controlled phone to the network (OsmocomBB)
  - From an A-bis proxy to the network or the phones

TODO
Where we go from here

- The tools for fuzzing mobile phone protocol stacks are available
- It is up to the security community to make use of those tools (!)
- Don't you too think that TCP/IP security is boring?
- Join the GSM protocol security research projects
- Boldly go where no (free) man has gone before

Current Areas of Work / Future plans

- UMTS(3G) support for NodeB and femtocells
- SS7 / MAP integration (Erlang and C)
- Playing with SIM Toolkit from the operator side
- Playing with MMS
- More exploration of RRLP + SUPL

Further Reading

- http://laforge.gnumonks.org/papers/gsm_phone-anatomy-latest.pdf
- http://bb.osmocom.org/
- http://openbsc.osmocom.org/
- http://openbts.sourceforge.net/
- http://airprobe.org/