

Confidence in SMS

Workshop

0sec – 0x736563
October 17th 2010



Short introduction

Pierre Pronchery, coder/auditor as:

- **DUEKIN CONSULTING** *IT-Security consultant*
- *loopb-ack GbR with other independent consultants*
- *DeforaOS an ambitious Open Source project*

Enough said...



GSM Primer: Background

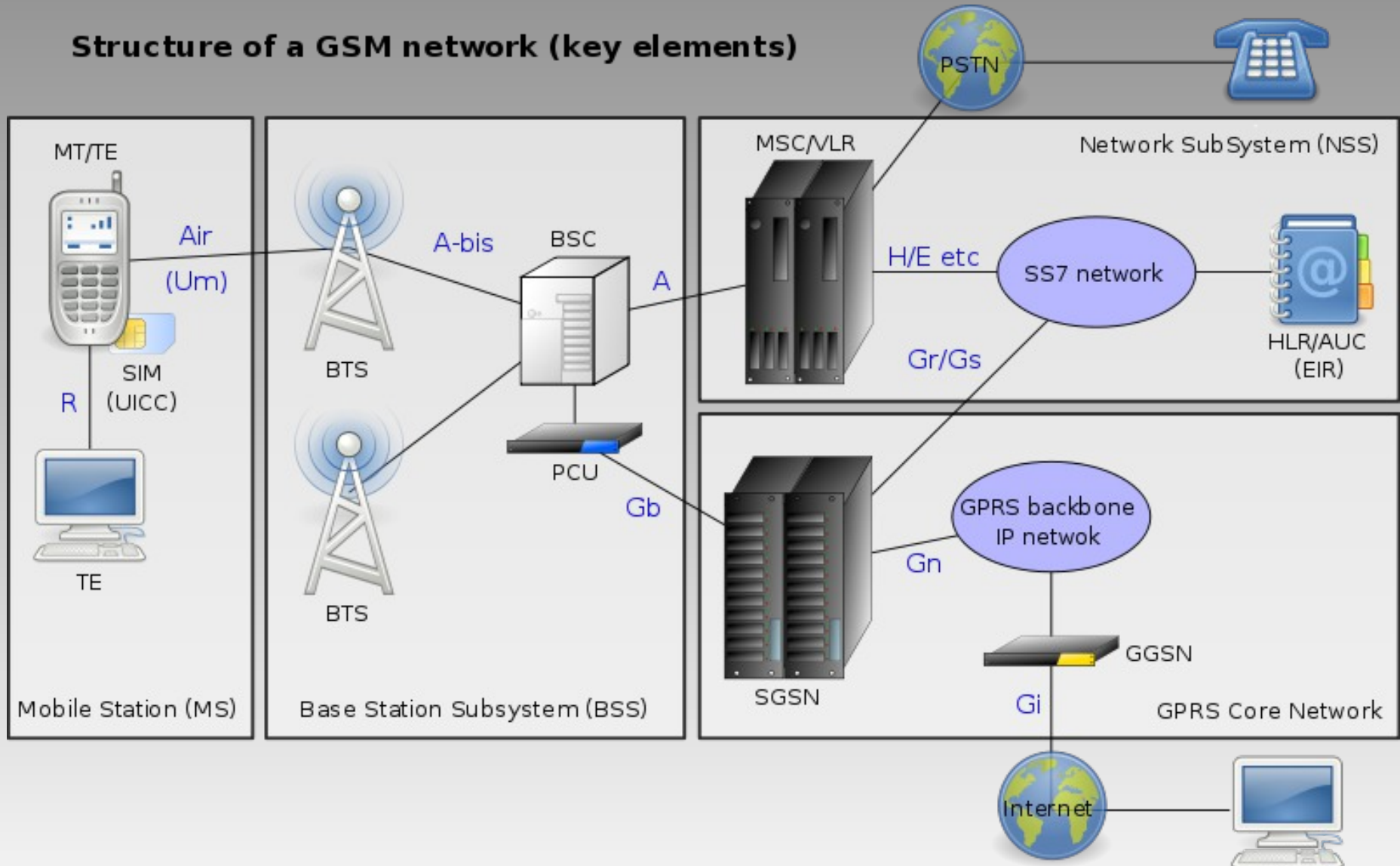
- “Groupe Spécial Mobile”, or “Global System for Mobile communications”
- 1.5 billion people, 212+ countries
- European project from 1982, first launch 1991
- Cellular network, ranges up to 35 km
- Operates in 850, 900, 1800 MHz (2G) or 2100 MHz (3G)
- “Moderate level of service security”

Source: Wikipedia



GSM Primer: Infrastructure

Structure of a GSM network (key elements)



GSM Primer: Security

- Authentication of the subscriber
- Optional encryption:
 - A5/1, primarily in Europe and US
broken since 12/2009, rainbow tables available
 - A5/2, more recent than A5/1 but also weaker
real-time decryption since 2008
 - A5/3 “Kasumi” for 3G
partly broken already



GSM Primer: Security

- Make your own base station: OpenBTS
- Monitor mode: DeforaOS Phone/Openmoko
- Scan and listen to the network: GNU Radio
- The same with the ability to call: OsmocomBB

A 30€ standard phone is enough to achieve this.



SMS Sub-primer

- “Short Message Service”, first message in 1992
- Slow start while fixing billing issues
- Uses the signaling channel
- Limited to 128 and then 140 bytes, meaning:
 - 160 ASCII 7-bit characters
 - 140 8-bit characters (or binary data)
 - 70 UTF-16 characters
- Longer messages can be concatenated



The price you pay

- The cost to users has essentially been stable
- Average price of an SMS today: \$0.11
- Some providers are rising the price...
- Flat-rate plans in Germany for 10€/month
- Lots of silly ring-tones, logos and ads
- Remote configuration data
- Like calls, only encrypted between MS and BTS

Read

<http://gthing.net/the-true-price-of-sms-messages>



Who you are, where you've been, what you know, where you go...

- Messages are sent to an SMSC (Short Message Service Center)
- Queued there for delivery
- Further transmission using the Mobile Application Part (MAP) of the SS7 protocol
- There is no guarantee of actual delivery
- You can request a delivery notification



Talking SMS

Using particular AT commands:

- *AT+CMGF format used for modem interaction*
- *AT+CMGS sending messages*
- *AT+CMGL listing messages*
- *AT+CMGR reading messages*
- *AT+CMGW storing messages*



Talking SMS: text mode

- Looks pretty simple:
AT+CMGF=1
AT+CMGS=+491337666404
Insert text here [CTRL+z]
- Limited character-set allowed
- No control over compression, message type...
- Not implemented on every phone



Talking SMS: PDU mode

- “Protocol Description Unit”
- Many different alphabets available (AT+CSCS)
- Looks like that:
AT+CMGF=0
AT+CMGS=23
>0011000B916407281553F80000AA0AE8329BFD4697D9EC
37 [CTRL+z]
- Quite complex and partly backwards...

See <http://www.dreamfabric.com/sms/>



Talking SMS: different encodings

- 7-bit default alphabet: see src/gsm.c line 88
- 8-bit: usually used for data
- UTF-16 (UCS2) for Japanese/Arabic...



Existing solutions

- Cryptophone from GSMK
- cryptosms.org, Open Source project
- TextSecure from Whisper Systems
- DeforaOS Phone, Open Source project
- Do you know more?



Proprietary solution: Cryptophone

- Developed by a German company, GSMK
- Open Source for your eyes only
- Runs on specific hardware as-is



Aging solution: CryptoSMS

- Open Source project (not active anymore?)
- Asymmetrical cryptography
- Written in Java for Mobile phone stacks
- Easy to install (open the .jad file in a browser)
- In practice almost only works on Symbian (no Blackberry, no Windows Mobile, no Android)
- Need to start a separate application



Proprietary solution: TextSecure

- Developed by Whisper Systems (Moxie Marlinspike)
- Proprietary application for Android only
- Asymmetrical cryptography, like OTR
- Open specifications or source code “soon”
- Easy to install (Android Market)
- Need to start a separate application
- Does not tell when messages are sent unencrypted



Existing solution: DeforaOS Phone

- Open Source project, one developer so far (hi!)
- Transparent user interface
- Messages are stored encrypted
- Written in C and Gtk+
- Only packaged for desktops or the Openmoko Freerunner so far
- Designed to not cause extra billing cost
- (Weak?) symmetrical cryptography



Choose your own workshop

Use it

- Installation
- Deployment
- Feedback

Break it

- Design review
- Step-by-step cracking
- How to fix it

Code it

- Development
- Cross-compilation
- Contributing

Dislike it

- Other ideas?



Use it: Installation (sources)

Depends on:

- libSystem
- Gtk+ 2.0

Optionally:

- Pulseaudio
- Panel
- Keyboard

Official releases:

<http://www.defora.org/>

From CVS:

```
$ cvs
```

```
-d:pserver:anonymous@anoncvs.defora.org:/Data/CVS co Defora0S
```

In particular:

- System/src/libSystem
- Apps/Desktop/src/libDesktop
- Apps/Desktop/src/Phone



Use it: Installation (packages)

For Debian:

```
# cat >> /etc/sources.list << EOF
deb http://build.hackable1.org/debian wip main
EOF
# apt-get update
# apt-get install phone
```

From pkgsrc-wip: <http://pkgsrc.sf.net/>

- wip/deforaos-libsystem
- wip/deforaos-phone
- wip/deforaos-panel (optional)
- wip/deforaos-keyboard (optional)



Use it: Deployment (configuration)

- Configuration file in `~/.phone`
`device=/dev/ttyUSB0`
`plugins=panel,smcrypt,debug`
`hwflow=1`
- ...or with the configuration interface



Use it: Deployment (configuration)

For the SMS encryption plug-in in particular:

- Enable the “smscrypt” plug-in
- Inside the “[smscrypt]” section:
 - Either add one secret per phone number,
 - Or set a global, fallback secret

```
[smscrypt]
```

```
1234=first_secret
```

```
+491234=another one
```

```
secret=fallback!
```



Use it: Deployment (configuration)

- It's up to you to exchange symmetrical keys securely
- ...and choose them wisely
- ...and change them regularly



Use it: Quick manual

- Does not create any window at start (except if a plug-in does)
- Use the following executables to show some:
 - phone-contacts
 - phone-dialer
 - phone-logs
 - phone-messages
 - phone-settings
- Panel applet available



Use it: Feedback

- How is it so far?



Code it: Development

- Hosted by and part of the DeforaOS project
- Happens in a CVS tree:
\$ cvs
-d:pserver:anonymous@anoncvs.defora.org:/Data/CVS co DeforaOS
- Web interface and daily archive available
- Official releases and screenshots too
- Single developer so far... (hi?)
- Phone depends on OpenSSL, libSystem, Gtk+ 2
- Optionally on Pulseaudio, Panel and Keyboard



Code it: Compilation #1

```
$ make bootstrap  
[...]
```

```
=====  
The source tree is now configured for your  
environment. Essential libraries and tools will  
now be installed onto your system unless you exit  
this script now with the CTRL+C key combination.  
Otherwise, press ENTER to proceed.  
=====
```

CTRL+C

```
$ (cd System/src/libSystem && make install)  
$ cd Apps/Desktop/src/Phone && make install  
$ phone
```



Code it: Helper structure

```
struct _PhonePluginHelper
{
    Phone * phone;
    char const * (*config_get)(Phone * phone, char const *
section, char const * variable);
    /* [...] more configuration helpers */
    int (*error)(Phone * phone, char const * message, int
ret);
    int (*event)(Phone * phone, PhoneEvent event, ...);
    int (*queue)(Phone * phone, char const * command);
    int (*register_trigger)(Phone * phone, PhonePlugin *
plugin, char const * trigger, PhoneTriggerCallback
callback);
}
```



Code it: Compilation #2

- Installation defaults to /usr/local:
\$ make PREFIX=/opt/Defora0S CPPFLAGS=' -
DPREFIX=\"/opt/Defora0S\"'



Code it: Cross-compilation

- Easier from within hackable:1
- Install a cross-compilation environment:
 - Native, <http://trac.hackable1.org/trac/wiki>
 - Virtual host,
<http://people.defora.org/~khorben/>
- Helper script available:
trunk/packages\$./package.sh DEBIAN_ARCH=armel
phone
- Can be tricky with intermediate versions : (



Code it: Integration

- Primary target is the DeforaOS Desktop
- Can easily be extended via plug-ins:
 - Presence in the system tray
 - DBus interface...
- Packaged for hackable:1
 - Debian-based distribution
 - Handful of additional packages and scripts
 - Provides ready-to-flash images for actual devices
- Can be compiled from pkgsrc-wip (NetBSD...)



Code it: Minimal plug-in

```
int plugin_init(PhonePlugin * plugin);
int plugin_destroy(PhonePlugin * plugin);
int plugin_event(PhonePlugin * plugin, PhoneEvent
event, ...);
void plugin_settings(PhonePlugin * plugin);
PhonePlugin plugin = {
    NULL,                /* set to helper */
    "Its name", "icon-name",
    plugin_init,        /* or NULL */
    plugin_destroy,
    plugin_event,
    plugin_settings,
    NULL                /* plug-in data */
};
```



Code it: Contributing

Via DeforaOS:

- Introduce yourself on devel@lists.defora.org
- Report bugs or wishes
- Send patches
- Gain my trust (and an account)
- Commit directly :)

Via hackable:1:

- Introduce yourself on devel@lists.hackable1.org
- Request an account
- Patch the packages
- Commit :)



Code it: Suggestions

- Anything I may have missed?



Break it: Design review #1

Encryption algorithm:

```
for(i = 0; i < *len; i++)
{
    (*buf)[i] ^= smscrypt->buf[j];
    smscrypt->buf[j++] = (*buf)[i];
    if(j != smscrypt->len)
        continue;
    SHA1_Init(&sha1);
    SHA1_Update(&sha1, smscrypt->buf,
smcrypt->len);
    SHA1_Final(smcraft->buf, &sha1);
    j = 0;
}
```



Break it: Design review #2

Decryption algorithm:

```
for(i = 0; i < *len; i++)
{
    (*buf)[i] ^= smscrypt->buf[j];
    smscrypt->buf[j++] ^= (*buf)[i];
    if(j != smscrypt->len)
        continue;
    SHA1_Init(&sha1);
    SHA1_Update(&sha1, smscrypt->buf,
smscrypt->len);
    SHA1_Final(smscrypt->buf, &sha1);
    j = 0;
}
```



Break it: Design review #3

Encryption sequence:

1. Buffers a SHA-1 hash of the secret
2. XORs the first 20 bytes of the message with it
3. Buffers a SHA-1 hash of the encrypted content
4. XORs the following 20 bytes of the message
5. Goto 3



Break it: Design review #4

Decryption sequence:

1. Buffers a SHA-1 hash of the secret
2. XORs the first 20 bytes of the message with it
3. Buffers a XOR of the decrypted content with it
4. XORs the following 20 bytes of the message
5. Goto 3



Break it: Step-by-step cracking #0

Master Bruce said: « *don't come up with your own encryption algorithm* »

He is right. Any ideas why?



Break it: Step-by-step cracking #1

Say, you intercept the following messages:

5c d9 09 0a 51 6f 18 9d 58 66 0f 50 94 5f c2 f0
68 60 22 ee 83 92 e1

4b d0 0e 59 08 69 1e 9d 4b 66 08 5e d5 4e 8c fa
6e 26

51 d4 01 11 51 6f 1f 9d 46 70 4e 55 db 4e 8c e0
75 39 36 f3 1c e0 df 0a e8 b4 d8

64 de 0c



Break it: Step-by-step cracking #2

Known/guessing plain-text:

- If 64de0c is “lol”, then we have:
 - Thi
 - Can
 - Yea
- I think this looks correct...
- From there, guess the rest of the key
 - Obvious or likely values combined
 - Statistical approach



Break it: Step-by-step cracking #3

Particular weaknesses:

- The length of the message is known
- First 20 bytes available for statistical analysis

Master Bruce also said « *don't encrypt arbitrary data!* »

- Just send a message with 20 constant characters...



Break it: How to fix it

- Use asymmetrical cryptography. Or:
- Hash each character with the preceding
- Pad messages with random data
- Use a real encryption algorithm (eg DES)
- Other ideas?
- Why not doing it now?



Dislike it: Other ideas?



Hope you had fun and learned some

Where to reach me:

- Independant <pierre.pronchery@duekin.com>
- Company <pierre.pronchery@loopb-ack.com>
- Open Source <khorben@defora.org>

