GSM – SRSLY?
Summary:
GSM Encryption needs to be shown insecure

GSM is constantly under attack:
- A5/1 cipher shown insecure repeatedly
- Lack of network authentication allow MITM intercept (IMSI Catcher)

However, GSM is used in a growing number of sensitive applications:
- Voice calls, obviously
- SMS for banking
- Seeding RFID/NFC secure elements for access control, payment and authentication

To rectify the perception of GSM’s security, we demonstrate its weaknesses

The community has computed the cryptographic base for a public demonstration of cracking GSM

This presentation details motives, approach and next steps of the “A5/1 Security Project”

Source: H4RDW4RE
GSM is global, omnipresent and insecure

80% of mobile phone market

200+ countries

4 billion users!

GSM encryption introduced in 1987 ...

... then disclosed and shown insecure in 1994

We need to publicly demonstrate that GSM uses insufficient encryption

Public break attempts

- A5/1 shown *academically* broken
- A5/1 shown more …
- … and more …
- … and more broken.
- Broken with massive computation
- *Rainbow table computation*
- *Tables never released*
- Too expensive
- Not enough known data in GSM packets
- … that didn't work.

Source: H4RDW4RE
GSM encryption is constantly being broken, just not publicly

All public break attempts of A5/1 have failed so far

- Academic breaks of A5/1 cipher are not practical [EC1997, FSE2000, Crypto2003, SAC2005]
- Cracking tables computed in 2008 were never released

15 years of A5/1 research have not produced a proof of concept (until today)

Meanwhile …

… A5/1 is constantly being circumvented by intelligence, law enforcement, and criminals

Source: H4RDW4RE
Active and passive intercept is common as attack devices are readily available.

Two flavors of attack devices

Active intercept:
- Phones connect through fake base station
- Easily spottable (but nobody is looking)

Passive key cracking:
- Technically challenging
  - Non-trivial RF setup
  - Heavy pre-computation
- Allows hidden operation

This talk demonstrates that GSM intercept is practical to raise awareness.

Source: H4RDW4RE, DeepSec GSM training
IMSI catching routes calls through a fake base station

- Advertise base station on beacon channel

IMSI: Subscriber Identity (~= username)
- Sort-of secret (replaced by TMSI asap)

MCC*: Mobile Country Code
- 262 for .de, 310-316 for USA
MNC*: Mobile Network Code
- Country-specific, usually a tuple with MCC
- 262-01 for T-Mobile Germany

Phones will connect to any base station with spoofed MNC/MCC
- If you claim it, they will come
- Strongest signal wins
- IMSI catching is detectable from phone, but no detect apps exists!
- Crypto is completely optional and set by the base station!!

* Full list of MNC/MCCs available on Wikipedia

Source: H4RDW4RE
IMSI catcher could even be built from open source components

**A** Setup
- **OpenBTS + USRP + 52MHz clock**
  - Easy to set up, Asterisk is hardest part
  - On-board 64MHz clock is too unstable
- **Software side is easy**
  - ./configure && make
  - Libraries are the only difficulty

**B** Configure
- **Set MCC/MNC to target network**
- **Find and use an open channel (ARFCN in GSM-ese)**

**C** Collect, Decode
- **Wireshark has a Built-in SIP analyzer**
- **Or: capture data on air with Airprobe and decode GSM packets**
The iPhone that wouldn’t quit

What if we want to test and not “catch” IMSIs?

- Set MCC/MNC to 001-01 (Test/Test)
- Phones camp to strongest signal
  - Remove transmit antenna
  - Minimize transmit power
- GSM-900 in .eu overlaps ISM in USA
  - 902-928MHz is not a GSM band in the USA

Despite all of this we could not shake an iPhone 3G*…

* Other iPhones would not connect at all.

Source: H4RDW4RE
Fun bugs exposed by OpenBTS

During testing, we saw bugs in OpenBTS and phones:

- Persistent MNO shortnames
  - Chinese student spoofed local MNO
  - Classmates connected
  - Network name of “OpenBTS”, even after BTS was removed & phones hard rebooted!

- Open / Closed registration
  - Separate from SIP-level HLR auth
  - Supposed to send “not authorized” message
  - Instead sent “You’ve been stolen” message
  - Hard reboot required, maybe more

- Still many bugs in GSM stacks
- They are being found thanks to open source

Source: H4RDW4RE
Active and passive intercept is common as attack devices are readily available

Two flavors of attack devices

Active intercept:
- Phones connect through fake base station
- Easily spottable (but nobody is looking)

Passive key cracking:
- Technically challenging
  - Non-trivial RF setup
  - Heavy pre-computation
- Allows hidden operation

Source: H4RDW4RE, DeepSec GSM training
A5/1 is vulnerable to generic pre-computation attacks

**Code book attacks**

- For ciphers with small keys, code books allow decryption
- Code book provides a mapping from known output to secret state
- An A5/1 code book is 128 Petabyte and takes 100,000+ years to be computed on a PC

<table>
<thead>
<tr>
<th>Secret state</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A52F8C02</td>
<td>52E91001</td>
</tr>
<tr>
<td>62B9320A</td>
<td>52E91002</td>
</tr>
<tr>
<td>C309ED0A</td>
<td>52E91003</td>
</tr>
</tbody>
</table>

This talk revisits techniques for computing and storing a A5/1 code book efficiently

Source: H4RDW4RE
Groundwork for table generation is complete and released as open source

High-speed A5/1 engine → Table Parameterization → Table Generation*

GSM decrypt PoC

Status

Source and doc available: reflextor.com/trac/a51

Tables seen on Bittorrent

* Community provided: fast graphics cards (NVidia or ATI) and Cell processors (Playstation)
Key requirement of code book generation is a fast A5/1 engine

Time on single threaded CPU: 100,000+ years

- **Parallelization**
  - GPUs*: hundreds of threads
  - FPGA: thousands of engines

- **Algorithmic tweaks**
  - GPUs*: compute 4 bits at once
  - FPGA: minimize critical path

- **Implementation tweaks**
  - GPUs*: bitslicing

3 months on 40 CUDA nodes

---

Latest speeds [chains/sec]
- Nvidia GTX280 – 500
- ATI HD5870 – 500
- PS3 Cell – 120

* NVidia CUDA and ATI Brook GPUs are supported

Source: H4RDW4RE
Cracking to be demonstrated on Wednesday

- The first tables started showing up on the congress FTPs and Bittorrents;
  - check reflextor.com/trac/a51 for up-to-date details
- We want more!
  - Please sort your tables before uploading
    (tutorial on reflextor.com/trac/a51)
  - After the congress, keep sharing through Bittorrent

- We continue to collect tables until Tuesday evening
- Current state to be demonstrated in **workshop**
  - **Wednesday Dec 30, 13:00**, Large workshop room (A03)
  - Bring encrypted GSM sniffs you want to decrypt
Pre-computation tables store the code book condensed

Longer chains := a) less storage, b) longer attack time

Source: H4RDW4RE, c't
Distinguished point tables save hard disk lookups

Hard disk access only needed at distinguished points

Source: H4RDW4RE, c't
Rainbow tables mitigate collisions

Rainbow tables have no mergers, but an exponentially higher attack time

Source: H4RDW4RE, c't
The combination of both table optimizations is optimal.

### Assumptions
- 2 TB total storage
- 99% success rate when collecting all available keystream*
- 50% success rate when only obvious keystream is used
- Near real-time decryption with distributed cracking network

### The most resource efficient table for A5/1 is:
- 32 DP segments of length $2^{15}$
- Merged into one rainbow
- 380 such tables with height $2^{28.5}$ needed

* Collecting all available key stream requires data from a registered phone

Source: H4RDW4RE
GSM discloses more known keystream than assumed in previous crack attempts

<table>
<thead>
<tr>
<th>Frame with known or guessable plaintext</th>
<th>Assignment</th>
<th>Timing known through</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Very early</td>
<td>Early</td>
</tr>
<tr>
<td>1. Empty Ack after ‘Assignment complete’</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>2. Empty Ack after ‘Alerting’</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>3. ‘Connect Acknowledge’</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>4. Idle filling on SDCCH (multiple frames)</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>5. System Information 5+6 (~1/sec)</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td></td>
<td>Known Channel</td>
<td>Unknown Channel</td>
</tr>
</tbody>
</table>

Mobile terminated calls

- 1. Empty Ack after ‘Cipher mode complete’
- 2. ‘Call proceeding’
- 3. ‘Alerting’
- 4. Idle filling (multiple frames)
- 5. ‘Connect’
- 6. System Information 5+6 (~1/sec)

Network terminated calls

- 1. Empty Ack after ‘Assignment complete’
- 2. ‘Alerting’
- 3. ‘Connect’
- 4. System Information 5+6 (~1/sec)

Source: GSM Standards
Industry responds by creating a new challenge

“… the GSM call has to be identified and recorded from the radio interface. […] we strongly suspect the team developing the intercept approach has underestimated its practical complexity.

A hacker would need a radio receiver system and the signal processing software necessary to process the raw radio data.” – GSMA, Aug.‘09

These remaining components of an interceptor could be repurposed from open source projects.

Source: GSMA press statement
Hypothetically, an interceptor can be built from open source components

<table>
<thead>
<tr>
<th>Component</th>
<th>“Radio receiver system”</th>
<th>“Signal processing software”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current capabilities</td>
<td>- USRP2 -</td>
<td>Decoding and recording one GSM channel</td>
</tr>
<tr>
<td>Needed Improvements</td>
<td></td>
<td>- Interpreting control channel data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Decoding several channels (v.3.0)</td>
</tr>
</tbody>
</table>

Bottleneck: USRP programming

Source: H4RDW4RE

Karsten Nohl - A5/1 Cracking
GSM’s security must be overhauled

Upgrading GSM’s encryption function should be a mandatory security patch

However, replacing A5/1 with A5/3 may not be enough:

A. The A5/3 cipher “Kasumi” is academically broken

B. The same keys are used for A5/1 and A5/3 (weakest link security)

Source: HARDWARE
Summary of the Attack on Kasumi:

- Data complexity: $2^{26}$ plaintexts/ciphertexts
- Space complexity: $2^{30}$ bytes (one gigabyte)
- Time complexity: $2^{32}$ (hardest part: ex search)
- Completely practical complexities
- Attack verified by actual software simulation
A5/3 can be cracked in a semi-active attack

1. Intercept an encrypted call
   \[ \text{start_cipher(A5/3, rand)} \]
   - Encrypted call data

2. Ask phone to reuse key
   \[ \text{start_cipher(A5/1, rand)} \]
   - Encrypted IDLE frames*
   - \text{Fake BTS}

3. Decrypt data
   - Decrypt call or SMS with key cracked from A5/1 trans-actions
   - GSM cracker done ✓ (but not passive and not realtime)

* IDLE frames contain known plaintext

Source: H4RDW4RE
All tools needed for the semi-active attack are openly available

**Intercept an encrypted call**
- Record encrypted call data
- Airprobe

**Ask phone to reuse key**
- `start_cipher(A5/1, rand)`
- Record encrypted frames
- Airprobe

**Decrypt data**
- Crack key from A5/1-encrypted IDLE frames
- A5/1 rainbow tables

- Decrypt calls
- OpenBTS/OpenBSC

* IDLE frames contain known plaintext

Source: H4RDW4RE
A5/1 cracking is just the first step …

- Pre-computation framework build to be generic
  - Any cipher with small key space
  - Flexible table layout
  - Various back ends: CPU, CUDA, ATI, FPGA
- All tools released open source

- Please get involved
  - Port table generator to cipher in your projects
  - Find data to be decrypted (i.e., through programming the USRP’s FPGA)
Questions?

<table>
<thead>
<tr>
<th>Documentation, Source</th>
<th>reflextor.com/trac/a51</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mailing list</td>
<td>tinyurl.com/a51list</td>
</tr>
<tr>
<td>c’t article</td>
<td>tinyurl.com/ct-rainbows</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Karsten Nohl</th>
<th><a href="mailto:nohl@virginia.edu">nohl@virginia.edu</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chris Paget</td>
<td><a href="mailto:chris@h4rdw4re.com">chris@h4rdw4re.com</a></td>
</tr>
</tbody>
</table>

Many thanks to Sascha Krißler, Frank A. Stevenson and David Burgess!