Karsten Nohl, Chris Paget – 26C3, Berlin

GSM – SRSLY?
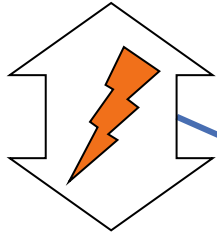
H4RDW4RE

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)

Security expec-tations divert from reality

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 demon-strate its weakness

- 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 Cracking Project"

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

'97	'00	'03	'05	'06	'03/'08

Tables never released

Too expensive

Not enough known data in GSM packets
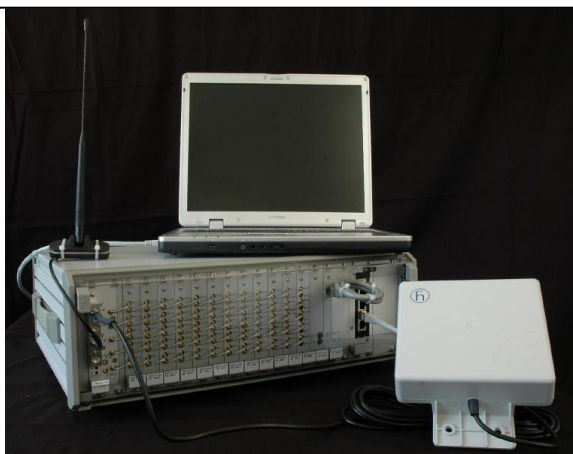
… that didn't work.

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 but never released

- 15 years of of A5/1 research have not produced a single PoC

- (until today)

Meanwhile …

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

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

Two flavors of attack devices

A

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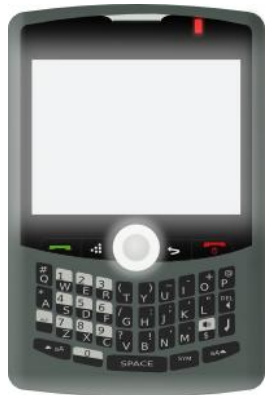- Easily spottable (but nobody is looking)

B

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

IMSI catching

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

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

H4RDW4RE

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 Tx 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

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

H4RDW4RE

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

Two flavors of attack devices

A

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

B

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

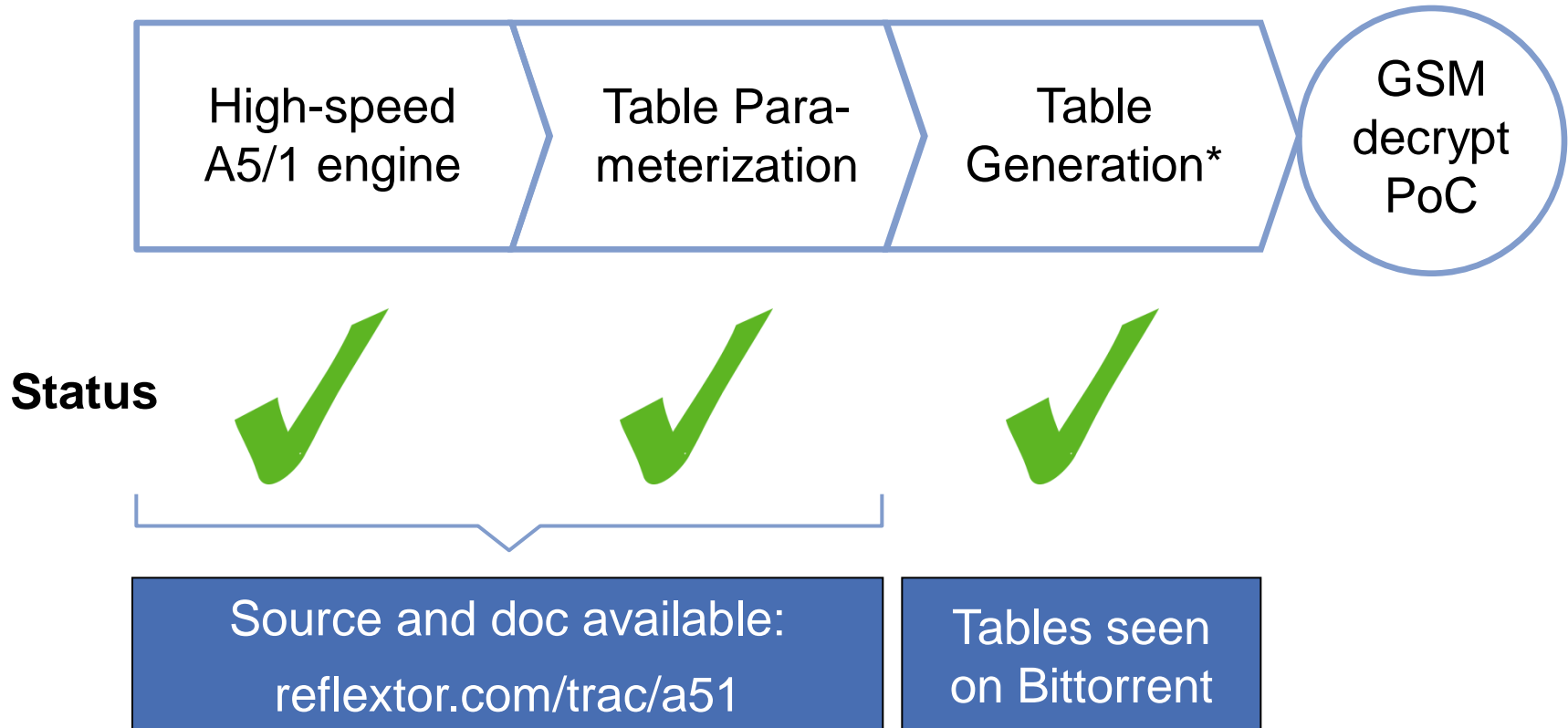# A5/1 is vulnerable to generic pre-computation attacks

Code book attacks

- For ciphers with small keys, code books allow decryption

Secret state	Output
A52F8C02	52E91001
62B9320A	52E91002
C309ED0A	52E91003

- 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

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

Groundwork for table generation is complete and released as open source

High-speed A5/1 engine	Table Para-meterization	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

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

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

C 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

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
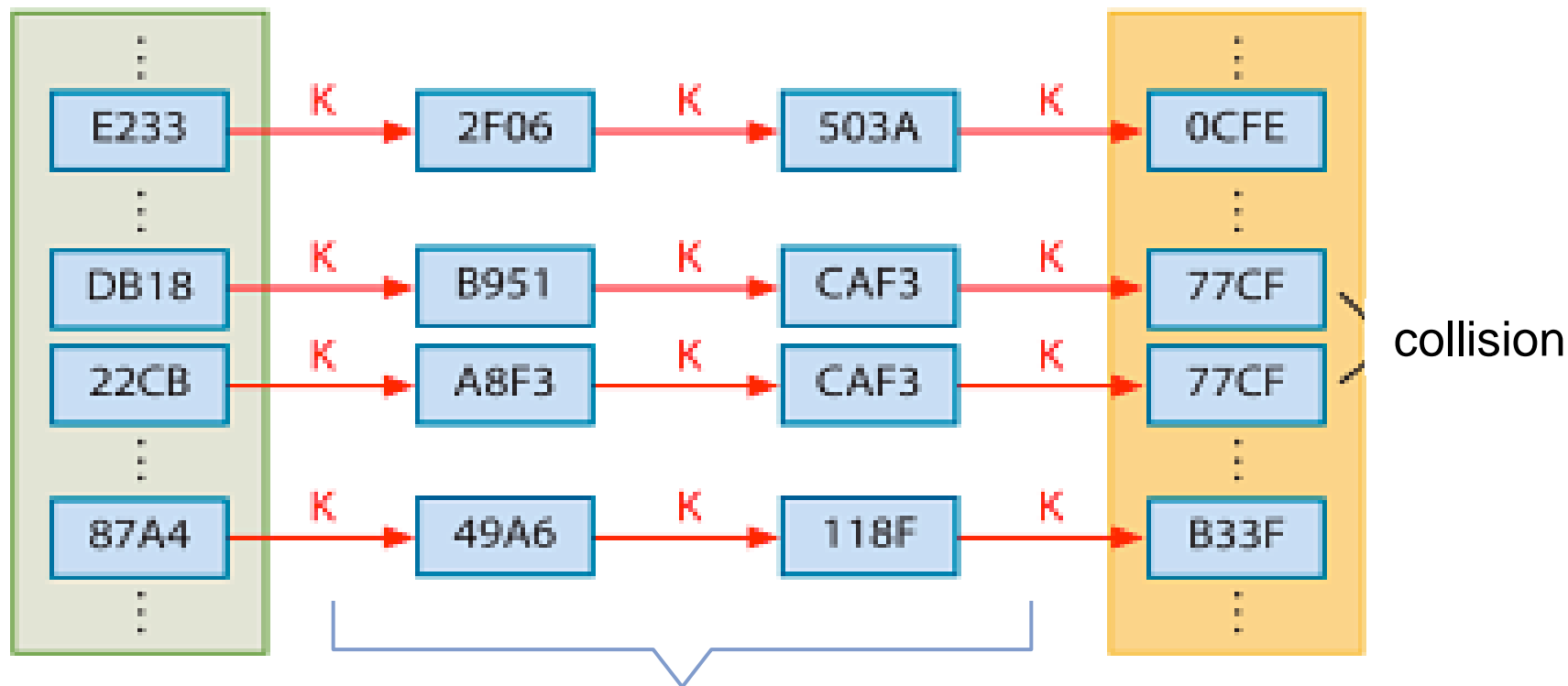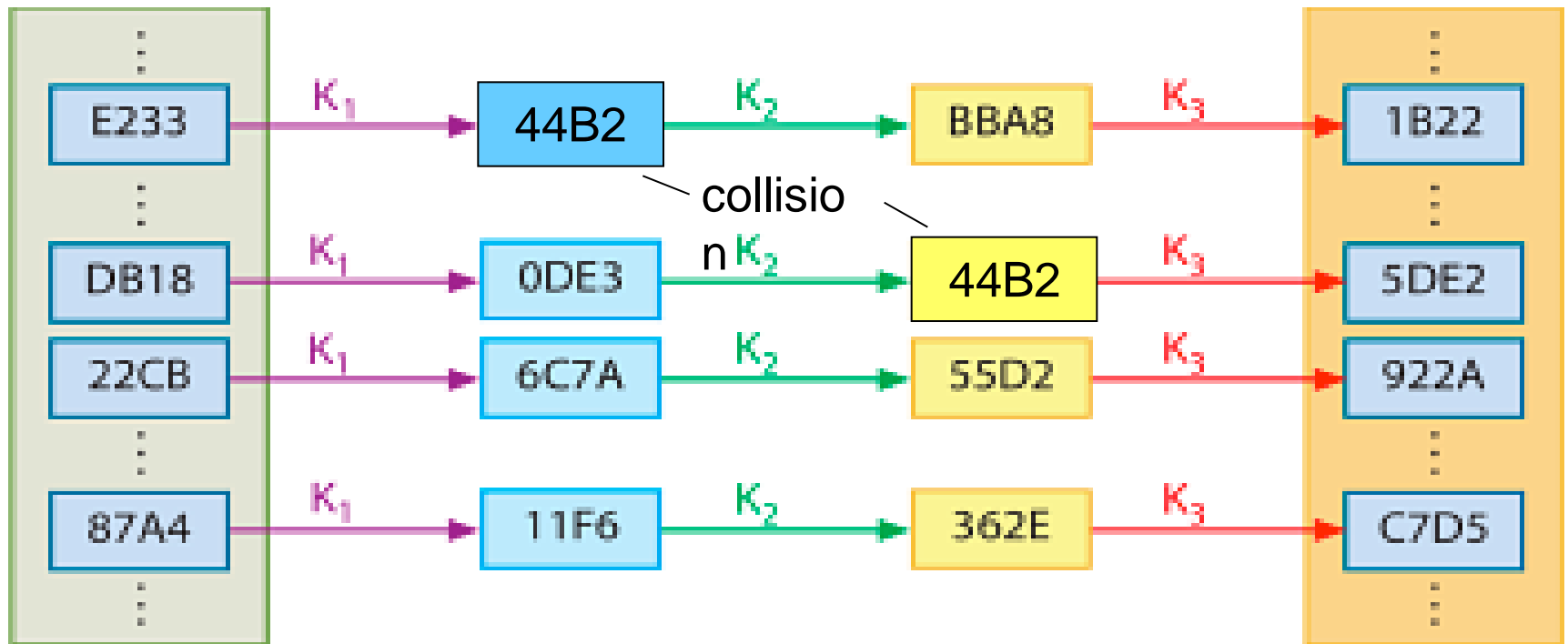

Longer chains := a) less storage, b) longer attack time

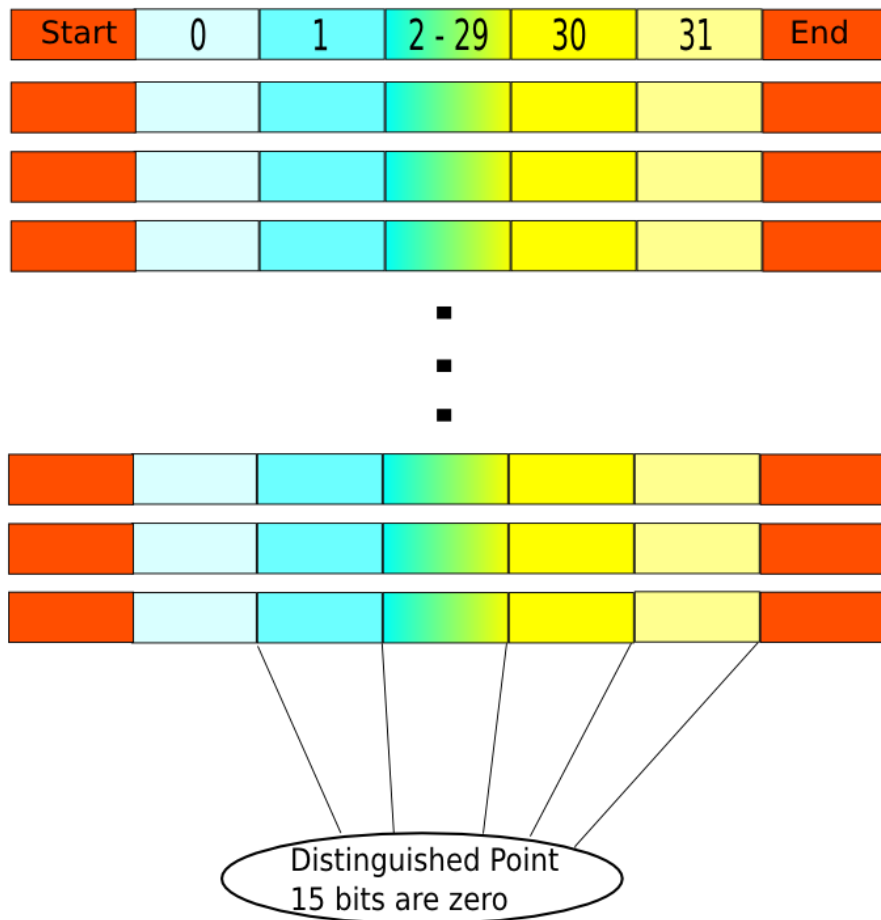# Distinguished point tables save hard disk lookups
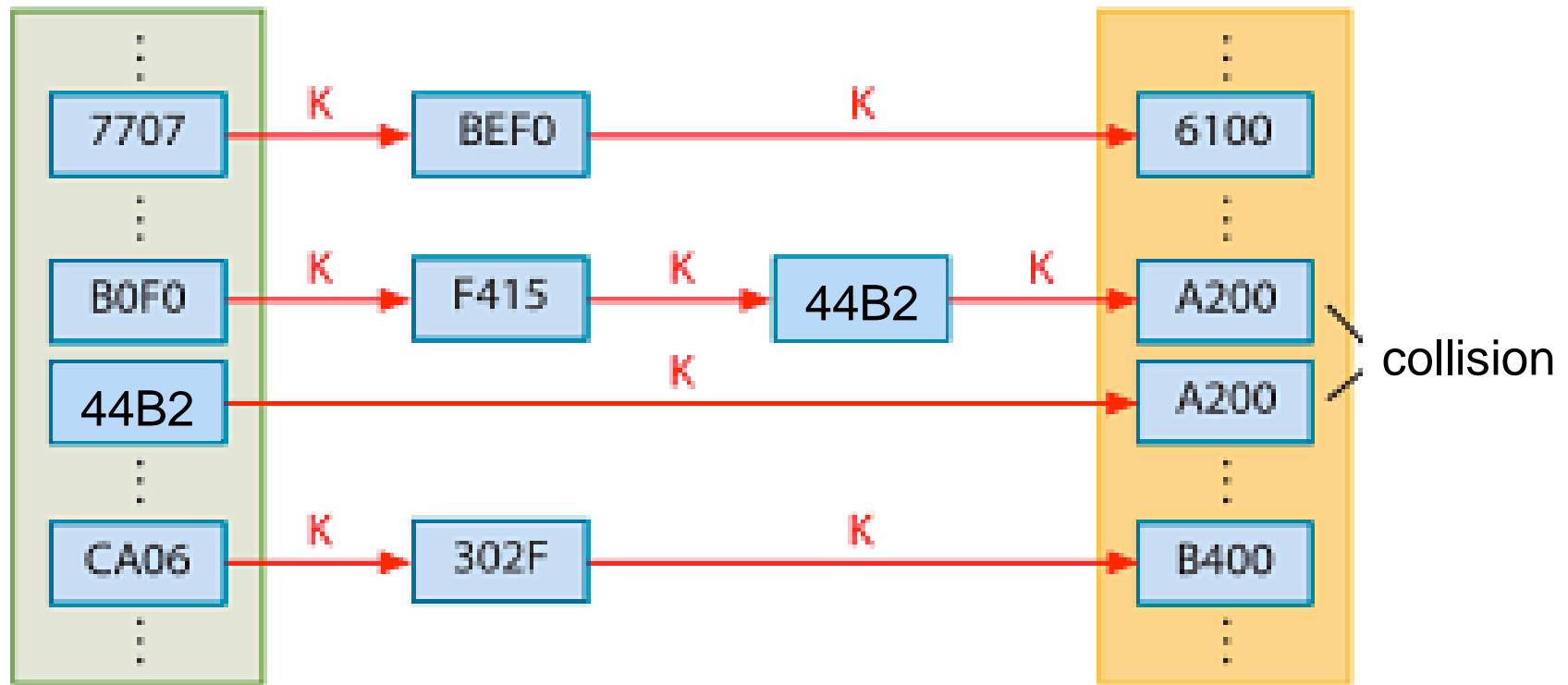
Hard disk access only needed at distinguished points

Rainbow tables mitigate collisions

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

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

GSM discloses more known keystream than assumed in previous crack attempts

Known Channel · Unknown Channel

	Frame with known or guessable plaintext	Assignment			Timing known through
		Very early	Early	Late	
Mobile terminated calls	1. Empty Ack after 'Assignment complete'	● Known	Unknown	Unknown	"Stealing bits"
	2. Empty Ack after 'Alerting'	Unknown	Unknown	Unknown	
	3. 'Connect Acknowledge'	Unknown	Unknown	Unknown	
	4. Idle filling on SDCCH (multiple frames)			● Known	
	5. System Information 5+6 (~1/sec)	◐	Unknown	◐	
Network terminated calls	1. Empty Ack after 'Cipher mode complete'	● Known	● Known	● Known	**Counting frames**
	2. 'Call proceeding'	● Known	● Known	● Known	
	3. 'Alerting'	● Known	Unknown	● Known	"Stealing bits"
	4. Idle filling (multiple frames)			● Known	
	5. 'Connect'	● Known	Unknown	Unknown	
	6. System Information 5+6 (~1/sec)	◐	Unknown	◐	

H4RDW4RE

19

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 <u>radio receiver system</u> and the **<u>signal processing software</u>** necessary to process the raw radio data." — GSMA, Aug.'09

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

Hypothetically, an interceptor can be built from open source components

| Compo-nent	"Radio receiver system"	"Signal processing software"
	– USRP2 –	**OpenBTS**
Current capabi-lities	■ Capture 25 MHz of GSM spectrum – Typically enough for one operator – Need separate boards for up- and down-link	■ Decode and record one GSM channel ■ Interpret control channel data
Needed Improve-ments	■ Decrease data rate (from 40MB/s) – Ability to detect active channels (energy detector) – Execute steps of decode chain in FPGA (Tune and Decimate) **Bottleneck: USRP programming**	■ Decrypt A5/1 packets given the correct key (v.3.0?) ■ Decode several channels (v.3.0)

GSM's security must be overhauled

Upgrading GSM's encryption function should be a mandatory security patch
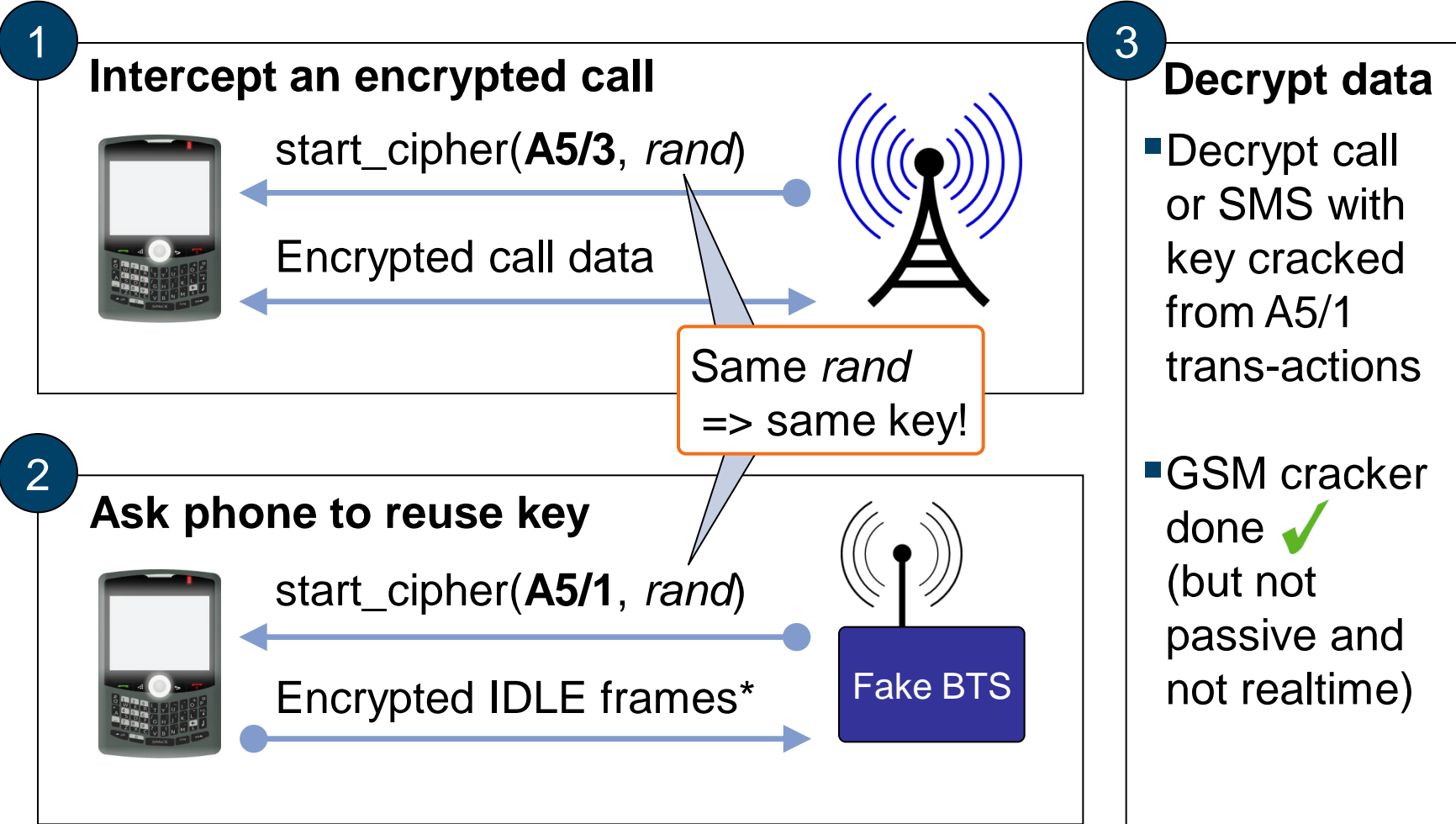
A5/1 ▶ A5/?
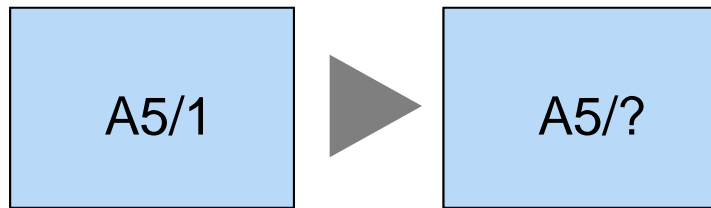
However, replacing A5/1 with A5/3 may not be enough:

A The A5/3 cipher Kasumi is academically broken

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

Summary of the Attack on Kasumi:
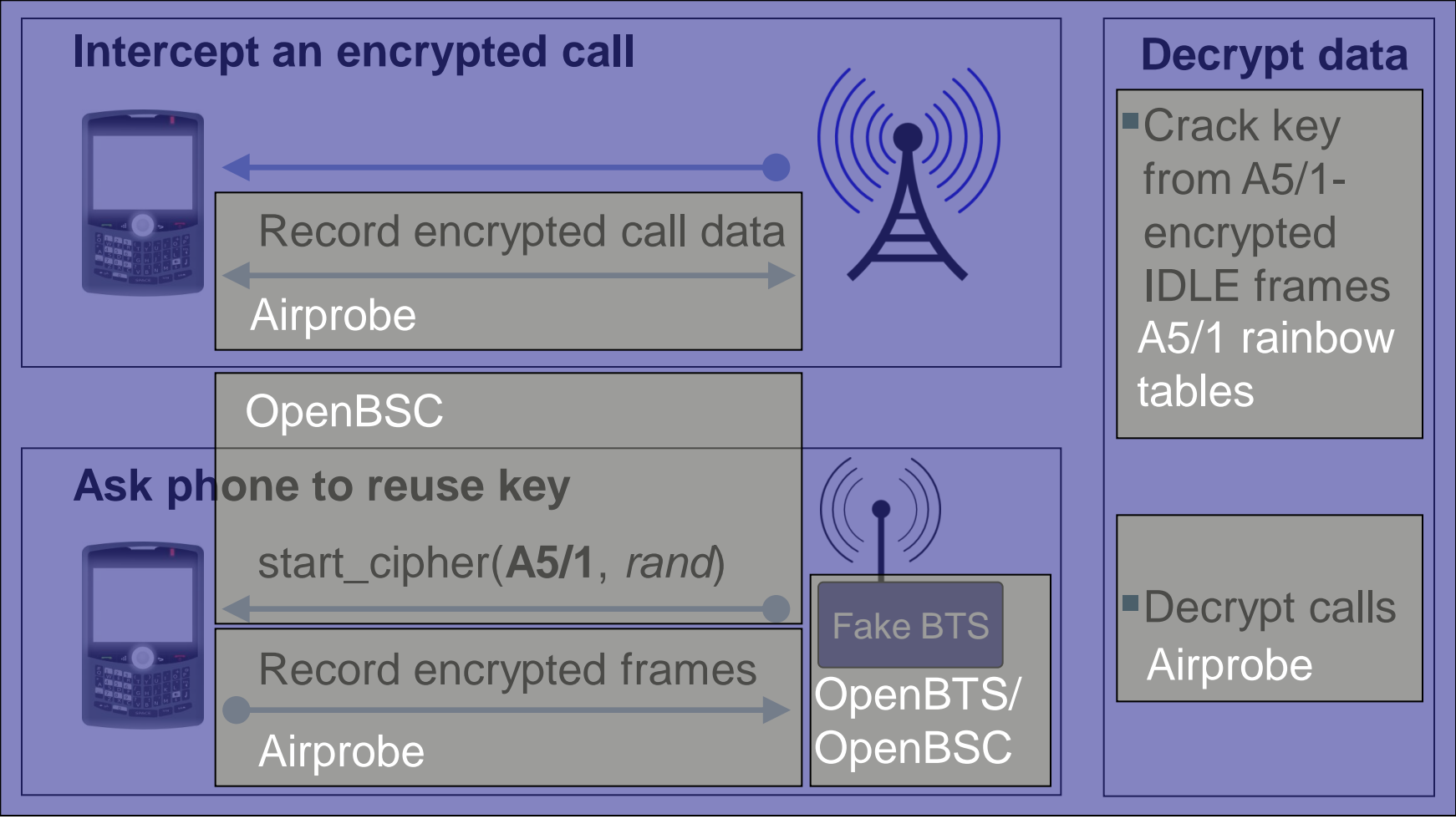
◆ Data complexity: 2^{26} plaintexts/ciphertexts

◆ Space complexity: 2^{30} bytes (one gigabyte)

◆ Time complexity: 2^{32} (hardest part: ex search)

◆ Completely practical complexities

◆ Attack verified by actual software simulation

1

Intercept an encrypted call

start_cipher(**A5/3**, *rand*)

Encrypted call data

Same *rand*
=> same key!

2

Ask phone to reuse key

start_cipher(**A5/1**, *rand*)

Encrypted IDLE frames*

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

B All tools needed for the semi-active attack are openly available

Intercept an encrypted call

Record encrypted call data

Airprobe

OpenBSC

Ask phone to reuse key

start_cipher(**A5/1**, *rand*)

Record encrypted frames

Airprobe

Fake BTS

OpenBTS/
OpenBSC

Decrypt data

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

- Decrypt calls
  Airprobe

*   IDLE frames contain known plaintext

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?

Documentation, Source	reflextor.com/trac/a51
Mailing list	tinyurl.com/a51list
c't article	tinyurl.com/ct-rainbows

Karsten Nohl	<nohl@virginia.edu>
Chris Paget	<chris@h4rdw4re.com>

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

H4RDW4RE