Eavesdropping on Satellite Telecommunication Systems

DRAFT

Benedikt Driessen
Horst-Goertz Institute for IT Security
Ruhr-University Bochum, Germany
benedikt.driessen@rub.de

February 2, 2012

Abstract

While communication infrastructures rapidly intertwine with our daily lives, public understanding of underlying technologies and privacy implications is often limited by their closed-source nature. Lacking the funding and resources of corporations and the intelligence community, developing and expanding this understanding is a sometimes tedious, but nonetheless important process. In this sense, we document how we have eavesdropped on our own communication in the Thuraya satellite network. We have used open-source software to build on recent work which reverse-engineered and cryptanalized both stream ciphers currently used in the competing satellite communication standards GMR-1 and GMR-2. To break Thuraya's encryption (which implements the GMR-1 standard) in a real-world scenario, we have enhanced an existing ciphertext-only attack. We have used common and moderately expensive equipment to capture a live call session and executed the described attack. We show that, after computing less than an hour on regular PC-hardware, we were able to obtain the session key from a handful of speech data frames. This effectively allows decryption of the entire session, thus demonstrating that the Thuraya system (and probably also SkyTerra and TerreStar, who are currently[1] implementing GMR-1) is weak at protecting privacy.

1 Introduction

In a time were even casual communication is gradually shifting towards the digital realm, the need to understand underlying technologies, protocols and privacy guarantees becomes ever more pressing. While the notion of "digital realm" captures virtually everything from "classic" telecommunication technology (e.g., GSM, UMTS) to Email, Skype and Facebook, GSM was (and still is) a compelling example why it is necessary for the public to openly discuss and understand closed-source systems. Historically, GSM relied on a set of undisclosed

[1] According to a Hughes press release from April 2, 2009, see http://www.hughes.com/HNSAgreement_with_SkyTerra_and_TerreStar_to_Implement_GMR1-3G_Satellite_Air_Interface.htm

encryption algorithms and was widely believed to be secure since it *did* provide "voice privacy". However, soon[2] after the algorithms were reverse-engineered [4], subsequent cryptanalysis [2,3,13] revealed serious weaknesses, casting doubt on the level of privacy and alerting security-conscious users.

In this sense, it is our firm belief that security weaknesses need to be exposed and demonstrated in order to asses their impact, to establish awareness and enforce evolution. This is, again, best demonstrated by GSM and its successors: Following the series of attacks on GSM, it is observable that a "shift of consciousness" has indeed happened in the telecommunication world: ZUC [9,10] is a stream cipher, developed and endorsed by China for inclusion in the emerging LTE standard, UMTS' long-term successor. The cipher has been made available for public scrutiny for over a year, a strategy clearly motivated by attacks on GSM [1]. The publication of ZUC is accompanied by a series of workshops created specifically for the purpose of evaluating the security of the cipher (and derived security mechanisms).

In line with our philosophy on disclosure and demonstration we continue recent work [6] on reverse-engineering of the encryption methods in contemporary satellite telephony systems. In this paper we document an attack on the Thuraya satellite network, detail hard- and software required for its implementation and report on the results. Since Thuraya is only one incarnation of the GMR-1 standard family, we assume that our attack also applies to other systems such as SkyTerra and TerreStar.

2 Background

In this section we introduce the background necessary for our attack. We shortly explain the basic notions of satellite telephony, and review the cipher implemented in the Thuraya system.

2.1 Thuraya and the GMR-1 Network Architecture

The Thuraya network implements the GEO-Mobile Radio (GMR-1) standard and provides satellite telephony for most of Europe, the Middle East, North, Central and East Africa, Asia and Australia. Thuraya offers a diverse range of products for fixed installations, handhelds (i.e., satphones) and even solutions for the maritime environment. With the help of Thuraya, voice, fax and IP-based data can be transmitted were "traditional" infrastructures (e.g., GSM, UMTS, WLAN, etc.) are not available. The Thuraya network consists of two operational[3] satellites (Thuraya-2 and Thuraya-3) and a set of terrestrial gateways and one primary gateway (located in Sharjah, UAE) handling the entire network as depicted in Figure 1. Gateway stations provide the connectivity to tethered networks, e.g., telephone calls to a landline are forwarded to the Public Switched Telephone Network (PSTN), and enable maintenance and configuration purposes. For this so called ground segment, conventional wavelength (C-Band) signals are used.

[2] Interestingly, even before the A5/1 and A5/2 algorithms were reverse-engineered by Briceno [4], Golic presented cryptanalysis [11] on what he calls the "alleged A5/1 cipher", which was indeed very close to the real cipher.

[3] Thuraya-1 has ceased to operate in May 2007 and has been moved to "junk orbit" [14].

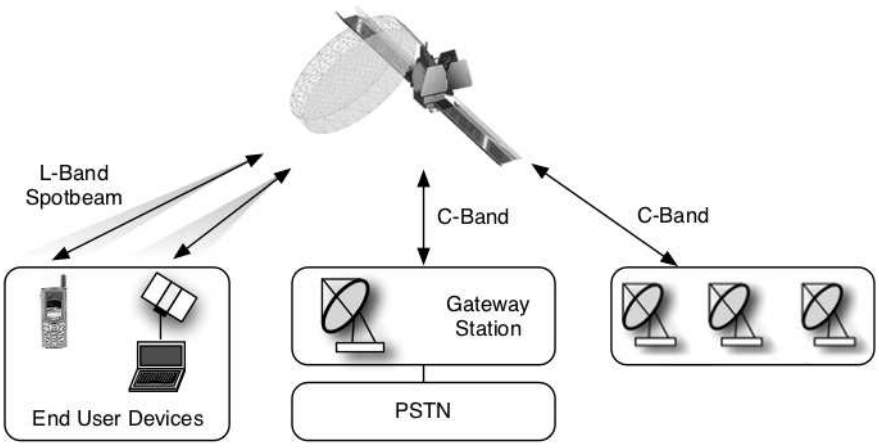

Figure 1: Layout of the GMR-1 network

The user segment operates on L-band carriers assigned to spotbeams, which are Thuraya's equivalent to cells in GSM (albeit covering far more area). In the L-band the 34 MHz frequency band from 1.525 GHz to 1.559 GHz is assigned for space-to-earth (downlink) communication while the uplink operates between 1.6265 GHz and 1.6605 GHz. Uplink and downlink are divided into 1087 paired carrier frequencies, with a spacing of 31.25 KHz. Just like in GSM, the Time Division Multiple Access (TDMA) time slot architecture is employed which partitions a carrier into disjunct timeslots of a fixed length.

Several logical channels (called channels from now on) can share a carrier frequency by being mapped on different timeslots. Due to the this architecture, a channel is uniquely determined by a frequency and a sequence of Timeslot Numbers (TN). There are different types of channels, but all are either Traffic CHannels (TCH) for voice or fax data, or Control CHannels (CCH). Data is sent over any channel in the form of frames, each with a distinct 19-bit Frame Number (FN), which are encoded to add redundancy and protect against transmission failures. For some channels, the encoded data is subsequently encrypted. The encoded (and encrypted) data is finally modulated before it is transmitted via the phone's antenna. The encoding scheme differs from channel to channel and is dependent on the respective reliability requirements as defined in the various standards.

Specific channels relevant for our attack are the Frequency Correction CHannel (FCCH), the Common Control CHannel (CCCH) and the Traffic CHannel 3 (TCH3). The FCCH is initially used by the satphone to determine its relative time and frequency error in order to synchronize with the system. The CCCH is used to send information to the phone when a new channel (e.g. TCH3) needs to be established[4]. These assignments message contain an Absolute Radio-Frequency Channel Number (ARFCN) and a TN, which is, as explained above, all that is required to use the channel. After TCH3 is set up on the up- and downlink, it can be used to transmit speech data.

[4]TCH3 is typically established at the beginning of a call.

2.2 Encryption in Thuraya

When a call is established in GMR-1, after setting up an appropriate channel (e.g. TCH3), the satellite initiates authentication by sending a request to the phone. This request contains a random number R, which is sent to the phone's SIM card where a specific secret key S is stored. Given (R, S), the SIM card derives a session key K and an authentication token. After authentication, encryption is switched on and all subsequent frames on the relevant channel are encrypted by a stream cipher. Keystream is generated for each frame individually, using the session key K and the frame number N of the frame as input for a black-box algorithm denoted as A5-GMR-1 in the GMR-1 specifications [7].

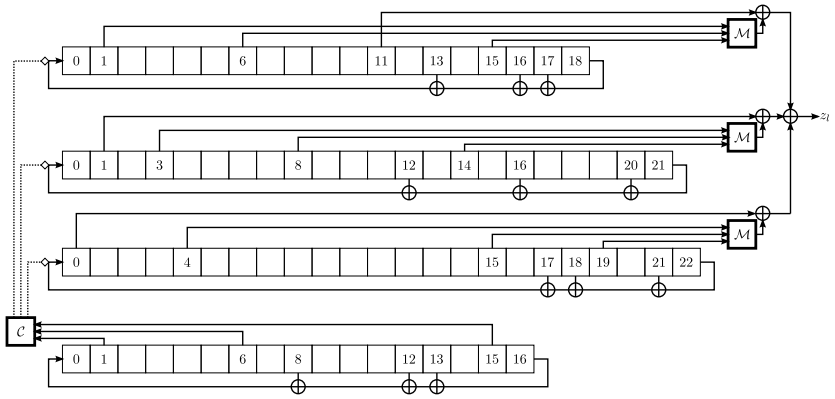

Figure 2: The A5-GMR-1 cipher

Although communication between satellites and the user segment is standardized and documented by ETSI, no details about the actual cryptographic algorithms used for voice encryption are disclosed. However, the cipher has recently been reverse-engineered and published in a paper, were the authors also propose different attacks [6]. As can be seen in Figure 2, A5-GMR-1 is a variant of the (weak) A5/2 cipher, which is still employed in GSM-systems worldwide. This discovery is only a mild surprise, since the GMR-standards are closely related to GSM.

Since A5/2 is well known and its derivative A5-GMR-1 is thoroughly described in afore mentioned paper, we will now only briefly re-examine the cipher. A5-GMR-1 is a stream-cipher with a 64-bit key, it consists of four LFSRs R_1, R_2, R_3 and R_4 (c.f. Figure 2, the LFSRs are enumerated from top to bottom) which are of different lengths. Each of the LFSRs has a pentanomial as feedback polynomial, which is used to update the LFSR's state when the LFSR is clocked. Clocking of R_1-R_3 is controlled by R_4, which, in combination with the clock-controller function $\mathcal{C}(\cdot)$, implements the irregular clocking scheme known from A5/2. After R_1-R_3 have been clocked, four bits of each LFSR are extracted of which three are fed into a majority-function

$$\mathcal{M}(x_2, x_1, x_0) = x_2x_1 + x_2x_0 + x_0x_1$$

which returns 1 if more then one input bit is 1 and 0 otherwise. The outputs of the majority-function and the other extracted bits are combined by XORing them to form one bit of keystream.

The cipher is operated in two modes, *initialization* and *generation* mode. Running the cipher in former mode includes setting the initial state of the cipher, which is done in the following way:

1. All four registers are set to zero.
2. A 64-bit initialization vector $\alpha = (\alpha_0, ..., \alpha_{63})$ is computed by XORing the bits of the 19-bit frame-number N and 64-bit session key K, i.e.,

$$\begin{aligned}\alpha = \mathcal{F}(K, N) = (&K_0, K_1, K_2, K_3 + N_6, K_4 + N_7, K_5 + N_8, K_6 + N_9,\\ &K_7 + N_{10}, K_8 + N_{11}, K_9 + N_{12}, K_{10} + N_{13},\\ &K_{11} + N_{14}, K_{12} + N_{15}, K_{13} + N_{16}, K_{14} + N_{17},\\ &K_{15} + N_{18}, K_{16}, K_{17}, ..., K_{21}, K_{22} + N_4,\\ &K_{23} + N_5, K_{24}, ..., K_{59}, K_{60} + N_0, K_{61} + N_1,\\ &K_{62} + N_2, K_{63} + N_3)\end{aligned}$$

3. α is clocked into all four registers, i.e., R_1 is clocked and one bit of α is XORed with the feedback-bit, R_2 is clocked and XORed with the same bit of α, etc. While doing this, no irregular clocking takes place.
4. The least-significant bits of all four registers are set to 1, i.e., $R_{1,0} = R_{2,0} = R_{3,0} = R_{4,0} = 1$.

We denote the whole initialization process by

$$(\underbrace{\beta_0, ..., \beta_{18}}_{R_1}, \underbrace{\beta_{19}, ..., \beta_{40}}_{R_2}, \underbrace{\beta_{41}, ..., \beta_{63}}_{R_3}, \underbrace{\beta_{64}, ..., \beta_{80}}_{R_4}) = \mathcal{G}(K, N),$$

where β is a 81-bit string, comprised of the consecutive bits of the four initialized registers. After all registers are initialized, irregular clocking is activated and the cipher is clocked for 250 times. The resulting output bits are discarded. After initialization, the cipher is ready for generating *actual* keystream bits $z_{250}^{(N)}, z_{251}^{(N)}, z_{252}^{(N)}, ...$

In Thuraya, a frame of length l, sent on the downlink, is encrypted by keystream bits $z_{250}^{(N)}, ..., z_{250+l}^{(N)}$, whereas the corresponding uplink frame uses bits $z_{250+l+1}^{(N)}, ..., z_{250+2l}^{(N)}$.

3 A Ciphertext-Only Attack on TCH3

The attack we present in the following is a variant of the ciphertext-only attack presented in [6], which itself was inspired by previous attacks [2, 13] on A5/2. We now briefly review the proposed attack which exploits several weaknesses which are either due to the design of the cipher or due to the use of the cipher in GMR-1:

1. Given R_4, the clocking behavior of A5-GMR-1 is uniquely determined.
2. Since the inputs to each majority-component are only from one register, one bit of keystream can always be expressed as a quadratic equation over $GF(2)$, which can easily be linearized.

3. In GMR-1, encryption is applied after encoding and scrambling[5], which are all linear operations in $GF(2)$.

4. For each two keystreams generated by the same session key but different frame-numbers, the respective initial states are linearly related by the XOR-differences of the frame-numbers.

Due to the first and second observation and given some keystream bits for a particular frame N we can guess R_4, clock the entire cipher for several times and generate a linearized system of equations

$$\mathbf{A} \cdot x = z^{(N)}, \tag{1}$$

describing keystream bits as linear combinations of the initial state of R_1, R_2 and R_3. If we guess R_4 correctly and $\mathbf{A}$ has full rank, solving the equation system gives the correct initial state which can easily be used to obtain the session key. Please note that, even if the session key is fixed, for different frame-numbers not only the keystream but also the initial state and the matrix describing its relation to the keystream will be different. The size of x (i.e., the number of variables in Equation (1)) and hence the minimum[6] number of known keystream bits can be computed by

$$v = \binom{18-k_1}{2} + \binom{21-k_2}{2} + \binom{22-k_3}{2} + (18-k_1) + (21-k_2) + (22-k_3)$$

where k_1, k_2 and k_3 are the number of bits we additionally may guess for R_1-R_3. By replacing variables by constants we can decrease the size of the equation system and the number of required keystream bits, but also increase the average amount of bits to guess for the whole attack to $2^{15+k_1+k_2+k_3}$.

We now use the principle we have outlined above (and the fact that encryption is applied to encoded data) for a ciphertext-only attack which explicitly targets the TCH3 channel in Thuraya. Encoding, scrambling and encrypting a 160-bit speech-frame $d^{(N)}$ with frame-number N can be expressed as

$$c^{(N)} = d^{(N)} \cdot \mathbf{G} + s + z^{(N)},$$

where $\mathbf{G}$ is the generator matrix of the code, s is a 208-bit pseudo-random scrambling sequence, $z^{(N)}$ the keystream generated for this frame and $c^{(N)}$ the resulting 208-bit codeword. $\mathbf{G}$ and s are known, additionally a parity-check matrix $\mathbf{H}$ can be derived from $\mathbf{G}$ with $\mathbf{H} \cdot w = 0$ iff $w = v \cdot \mathbf{G}$. Due to this property, if we invert scrambling for a codeword $c^{(N)}$, we get

$$\mathbf{H} \cdot \left(c^{(N)} + s\right) = \mathbf{H} \cdot \left(d^{(N)} \cdot \mathbf{G} + z^{(N)}\right) = \mathbf{H} \cdot z^{(N)}.$$

Given a syndrome $r^{(N)} = \mathbf{H} \cdot \left(c^{(N)} + s\right)$, we can again set up an equation system in variables $x_0, x_1, ..., x_{v-1}$ of the initial state by guessing R_4, clocking the cipher 250 times (to account for the warm-up phase) and another 208 times, i.e.

$$\mathbf{H} \cdot (\mathbf{A} \cdot x) = \mathbf{S} \cdot x = r^{(N)}.$$

[5] While encoding adds redundancy, scrambling is used to "[...] randomize the number of 0s and 1s in the output bit stream." [8].

[6] We need at least as many keystream bits as we have variables and thus equations. However, since not all equations we obtain by clocking the cipher based on R_4 are necessarily linearly independent, we may need even more keystream bits.

Here, $\mathbf{A}$ is the $208 \times v$ matrix that describes the linear relation between x and the bits $z_{250}, z_{251}..., z_{457}$ generated by the cipher. Please note that $\mathbf{H}$ is a 48×208 matrix and subsequently $\mathbf{S}$ is a $48 \times v$ matrix which implies that for $v > 48$ this system is not uniquely solvable.

Now we describe the actual steps of our attack for which we assume that we are in possession of one descrambled codeword $c^{(N_0)'} = c^{(N_0)} + s$ and n 48-bit syndromes $r^{(N_0)} = \mathbf{H} \cdot c^{(N_0)'}, r^{(N_1)}, ..., r^{(N_{n-1})}$ which correspond to TCH3 downlink data encrypted under the same session key. Our attack is parameterized by (n, k_1, k_2, k_3) and recovers the initial state $\beta = \mathcal{G}(K, N_0)$. Before we proceed, we need to introduce the helper-function $\mathcal{V}(\cdot)$ which can be applied to extract certain bits of the 81-bit state of A5-GMR-1. Depending on the configuration of the attack, $\mathcal{V}(\cdot)$ will extract a bitstring which corresponds to these positions of the overall state whose bits we have guessed (i.e., R_4 and parts of the other registers).

1. Systematically guess the bitstring γ which has $20 + k_1 + k_2 + k_3$ bits (also incorporating the fixed bit per LFSR). For each syndrome $0 \leq i < n$ do the following:
   (a) Compute the 81-bit difference $\delta = \mathcal{G}(0, N_0) + \mathcal{G}(0, N_i)$ in the initialization state for frame-number N_0 and N_i.
   (b) Modify γ by XORing it with the corresponding positions of δ, i.e., $\gamma' = \gamma + \mathcal{V}(\delta)$.
   (c) Based on γ' and δ generate a linearized $458 \times v$ matrix $\mathbf{B}$ which describes the linear relation between the initial state for N_0 and the 458 keystream bits generated for $r^{(N_i)}$.
   (d) Discard the first 250 rows of $\mathbf{B}$ to obtain a $208 \times v$ matrix $\mathbf{B}'$.
   (e) Compute the $48 \times v$ matrix $\mathbf{S}' = \mathbf{H} \cdot \mathbf{B}'$ and add those rows of $\mathbf{S}'$ (and the corresponding bits from $r^{(N_i)}$) to the equation system $\mathbf{S} \cdot x = r$, which are linearly dependent from all previously existing rows of $\mathbf{S}$.
   (f) Abort if $\mathbf{S}$ has full rank.
2. Solve the equation system by computing $x = \mathbf{S}^{-1} \cdot r$ and combine the guessed bits and x appropriately to obtain the 81-bit initialization state candidate β.
3. Initialize A5-GMR-1 with β and clock it to obtain 208 bits of keystream $z^{(N_0)}$ for $c^{(N_0)'}$ and verify that
$$\mathbf{H} \cdot \left(c^{(N_0)'} + z^{(N_0)'} \right) \stackrel{!}{=} 0.$$
   If this equation holds, decrypting $c^{(N_0)'}$ produced a valid codeword. This implies we have produced the correct keystream and therefore (most likely) the correct initial state.

Once we have $\beta = \mathcal{G}(K, N_0)$ we can set up another equation system

$$\mathbf{L} \cdot \alpha = \beta \qquad \text{with} \qquad \alpha = \mathbf{L}^{-1} \cdot \beta = \mathcal{F}(K, N_0)$$

where $\mathbf{L}$ describes the process of clocking α into all four LFSRs (and setting the lowest bit per LFSR to 1). Then we can easily derive the session key, i.e., $K = \mathcal{F}(\alpha, N_0)$.

4 Executing the Attack

Here we describe our implementation of the attack, which includes the hard- and software we have used. We stress that our attack only targets the downlink, which is due to two reasons:

1. The downlink can be received at least everywhere in the area of the assigned spotbeam. Listening to the uplink on the other hand requires a close proximity to the targeted satphone.

2. It is not in our interest to develop a full-blown interceptor. We simply want to show that our attack is practicable. Furthermore, if we can extract the session key from the downlink, we can also decrypt the uplink.

We conclude this section with the results we have obtained by attacking our own speech data, intercepted from an actual satphone conversation.

4.1 Hard- and Software

The hardware necessary to eavesdrop on Thuraya's L-band frequencies consists of an appropriate antenna and a Software Defined Radio (SDR) system. In principal, a directional antenna can be built with a very low budget. However, it turned out that assembling a helical antenna with exact tuning is a delicate process, so we opted for a commercial antenna which can be bought as accessory for Thuraya docking-stations. A USRP-2 and a contemporary PC comprise the SDR system, where GNUradio-based [5] software is used for receiving and filtering data, and code from the OsmocomGMR project[7] [12] is responsible for demodulating and descrambling TCH3 frames. Eavesdropping on ARFCN 1007, the frequency identifier assigned to the spotbeam covering most of Germany, we were able to intercept TCH3 assignment messages. Tuning to the assigned channel we have captured downlink speech data from our own Thuraya SO-2510 phone which gave us a predictable timing behavior and a reliable source of data – without interfering with the privacy of uninvolved parties.

4.2 Implementation

For the implementation of our attack we have established[8] a set of parameters,

$$n = 32, \qquad k_1 = 0, \qquad k_2 = 2, \qquad k_3 = 4, \qquad N_0 = 0,$$

which implies that the equation systems we generate have $v = 532$ variables, therefore $\mathbf{S}$ is a matrix of dimension 532×532. In order to construct these matrices, we require a minimum of $\lceil 532/48 \rceil = 12$ TCH3 frames. This is typically

[7] OsmocomGMR is a subproject of Osmocom, see `http://www.osmocom.org`. The aim of the Osmocom project is to establish open-source implementations of a wide range of communication standards, e.g., GSM, TETRA and even GMR-1. The implementation of OsmocomGMR is still in its infancy but evolved enough for our purposes.

[8] Determining k_1, k_2 and k_3 was done in a heuristic fashion with the aim to minimize linear dependencies – thus keeping n low – and also the size of $\mathbf{S}$ in order to reduce computational overhead for solving equation systems. It remains an open question to mathematically determine how many and even more *which* bits of R_1-R_3 to guess, given constraints such as a fixed number of frames, to achieve optimal performance.

enough, but due to linear dependencies sometimes more frames are used, although never more than $n = 32$. Given these parameters, to obtain one session key we need to generate and solve 2^{21} equation systems and consequentially test as many state candidates (by decoding via another matrix operation) on average. In order to speed up the process we have divided our attack into two steps:

1. In the *offline* phase we generate and store all $\mathbf{S}$ matrices for a particular choice of (n, k_1, k_2, k_3) and a starting frame-number N_0 from which all following numbers are derived incrementally, i.e., $N_i = N_0 + i$ for $0 < i < n$. While adding rows to the matrix, we consider the frame-number induced XOR-differences in the initialization states and maintain an upper triangular form of the matrix. This feature of $\mathbf{S}$ allows to quickly test for linear dependencies, makes the matrix consume less space when stored on disk and enables faster solving in the online phase.

2. In the *online* phase, we iterate over all stored matrices, reconstruct r from captured TCH3 data (and auxiliary information) and solve the equation system, which requires only backward-substitution due to the form of the matrix.

Please note this subtlety: Due to the fact that we maintain upper triangular form in the offline phase, rows in $\mathbf{S}$ are often linear combinations of several rows which would also apply to the right-hand side of an actual equation system. Since the offline phase is (mostly) independent of any real data (i.e., we generate $\mathbf{S}$, but not r), we need to store information on how to combine the respective bits of the syndromes to obtain the corresponding vector r. Also note that the offline phase uses a fixed set of frame-numbers, since these are part of the initialization process. However, the TCH3 frames we use in the online phase do not necessarily need to have the same frame-numbers, it is only important that they have the same XOR-differences (and the corresponding frames were received without transmission errors).

4.3 Results

Now we present the results of actually carrying out the attack. In the offline phase we have generated approximately $350GB$ of LZ-compressed data. We have executed our attack against a set of 32 TCH3 frames with successive frame-numbers and matching XOR-differences. The online phase completed after 64% of the searchspace was tested, taking 43 minutes on an Intel Xeon E5540 processor. As a result, we have obtained the session key for one call, allowing decryption of the complete session. Since the speech codec used in GMR-1 is still being reverse-engineered by the OsmocomGMR project, we are currently not[9] able to actually reproduce the communication that took place.

5 Conclusion

We have described a ciphertext-only attack on the recently reverse-engineered A5-GMR-1 stream cipher, which is used to secure the Thuraya satellite commu-

[9] We note that there is still a remainder of "security by obscurity" in Thuraya (and GMR-1), although – quite ironically – not due to security related mechanisms.

nication system. Our attack and its implementation is specifically tailored for a real-world attack on Thuraya, but most likely applies to all communication systems based on the GMR-1 standard. We detail the hard- and software components we have used to capture and decode downlink speech data. We have executed the proposed attack with a non-optimized implementation on regular PC-hardware and only a handful of encrypted TCH3 frames (and some precomputed data). We have obtained the encryption key in *less than an hour* of computation, showing that our attack is indeed feasible. Our attack is easy to extend to the uplink and even more easy to parallelize. It can be further improved by using GPUs or even FPGAs to ultimately obtain real-time decryption capabilities.

We subsume our results with the explicit warning that solely *GMR-1 based systems should not be relied on* if strong privacy is required.

Acknowledgement

Understanding existing, closed-source communication infrastructures, their security aspects and implications is inherently important and only made possible by dedicated individuals. Therefore, we gratefully acknowledge that implementing and carrying out our attack would not have been possible without the availability of the OsmocomGMR [12] project.

We also thank Daehyun Strobel for fruitful discussions and his deep understanding of modulation methods.

References

[1] Steve Babbage. The History and Pre-History of ZUC. Technical report, 2011. Available from: `http://www.dacas.cn/zuc11/slides/session1/Steve%20Babbage.ppt`.

[2] Elad Barkan, Eli Biham, and Nathan Keller. Instant Ciphertext-Only Cryptanalysis of GSM encrypted communication. In *International Crytology Conference (CRYPTO)*, pages 600–616, 2003.

[3] Alex Biryukov, Adi Shamir, and David Wagner. Real Time Cryptanalysis of A5/1 on a PC. In *Fast Software Encryption (FSE)*, 2000.

[4] M. Briceno, I. Goldberg, and D. Wagner. A pedagogical implementation of the GSM A5/1 and A5/2 "voice privacy" encryption algorithms, 1999. Originally published at `http://www.scard.org`, mirror at `http://cryptome.org/gsm-a512.htm`.

[5] Jonathan Corgan. GNURadio, January 2012. Available from: `http://gnuradio.org/`.

[6] Benedikt Driessen, Ralf Hund, Carsten Willems, Christof Paar, and Thorsten Holz. Don't Trust Satellite Phones: A Security Analysis of Two Satphone Standards. In *IEEE Symposium on Security and Privacy*, 2012. To appear.

[7] ETSI. ETSI TS 101 376-3-9 V1.1.1 (2001-03); GEO-Mobile Radio Interface Specifications; Part 3: Network specifications; Sub-part 9: Security related Network Functions; GMR-1 03.020, 2001.

[8] ETSI. ETSI TS 101 376-5-3 V1.2.1 (2002-04); GEO-Mobile Radio Interface Specifications; Part 5: Radio interface physical layer specifications; Sub-part 3: Channel Coding; GMR-1 05.003, 2002.

[9] ETSI/SAGE. Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 and 128-EIA3. Document 1: 128-EEA3 and 128-EIA3 Specification. Version 1.4, 2010.

[10] ETSI/SAGE. Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 and 128-EIA3. Document 2: ZUC Specification. Version 1.4, 2010.

[11] Jovan Dj. Golic. Cryptanalysis of alleged A5 stream cipher. In *Proceedings of the 16th annual international conference on Theory and application of cryptographic techniques*, EUROCRYPT'97, pages 239–255. Springer-Verlag, 1997.

[12] Sylvain Munaut. OsmocomGMR, January 2012. Available from: `http://gmr.osmocom.org/`.

[13] Slobodan Petrovic and Amparo Fuster-Sabater. Cryptanalysis of the A5/2 Algorithm. Technical report, 2000. `http://eprint.iacr.org/`.

[14] TBS. The Satellite Encyclopedia, January 2012. Available from: `http://www.tbs-satellite.com/tse/online/sat_thuraya_1.html`.