Analysis Suite go2signals

Manual

go2key

Product Version v26.1

November 25, 2025

PROCITEC®
HOUSE OF SIGNALS

Imprint

PROCITEC GmbH
Rastatter Straße 41
D-75179 Pforzheim
Germany

Phone: +49 7231 15561 0
Fax: +49 7231 15561 11
Email: service@procitec.com
Web: www.procitec.com

CEO:
Dipl.-Kaufmann / M. Sc. Stefan Haase

Registration Court:
HRB 504702 Amtsgericht Mannheim
Tax ID:
DE 203 881 534

Document ID:
PROCITEC-IMA-go2key_E-948ca82055

Contents

1. General

1.1. Welcome to go2key

The main purpose of go2key is the identification of an encryption *key* used to protect a radio transmission. Current version provides means for determination of an encryption *key* for DMR signals with ARC4 encrypted voice calls. Once the *key* is known, the decryption can be performed using standard workflow within any appropriate application within go2signals line, i.e. go2MONITOR or go2DECODE.

It includes the following features:

- determination of encryption *key* for DMR signals with ARC4 encrypted voice calls
- use of computational power of all CPU cores on a given machine
- interruption and resume of computation
- recall of recently recovered *key*s
- intuitive graphical user interface (GUI)
- powerful command line interface (CLI)

2. Basics

2.1. Software Start

To start go2key graphical user interface (GUI) click the desktop icon or use the operating system's start-menu. Using the command line interface requires, that the installation path of the go2key is known at the command line prompt. Either through appending it to the system's PATH variable or calling application with the full path of the go2key executable.

2.2. Overview

2.2.1. Using the graphical user interface

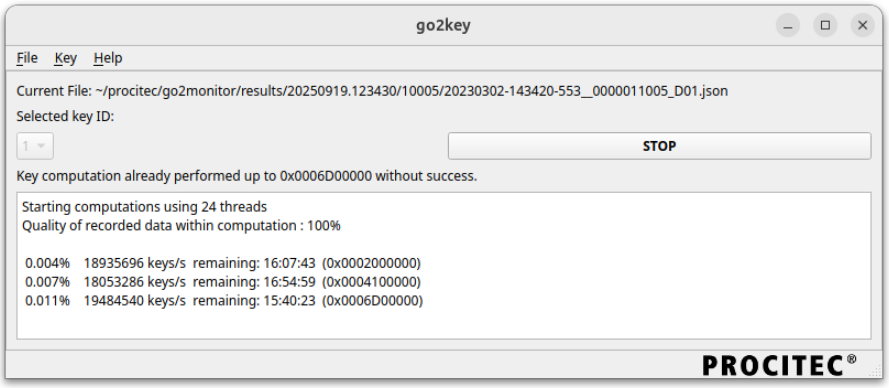


Figure 1: go2key with computation in progress

The encryption *key* recovery requires input data in form of a sample of the digital transmission, i.e. encrypted voice frames. These data frames can be extracted using go2MONITOR or go2DECODE and stored into a JSON (`*.json`) file. Please refer to chapter 3. Workflow on page 8 for details. Once a sample of encrypted voice frames (the JSON file) has been loaded, a list of *key-id*s used in the transmission appears in the drop down selection box. Computation is performed for selected *key-id* . If the amount and quality of the data sample is sufficient for *key* extraction then the *„Find key"* button starts computation.

The log window shows current progress of computation including tested *key* range and estimated remaining time. Some additional information is also presented in case data quality does not guarantee successful *key* recovery.

The computation can be paused at any time by *„Stop"* button. Pressing *„Find key"* again resumes data processing. The GUI application preserves the *key* extraction state for a given JSON file and selected *key-id* for a few recently executed computations. It means, that in case of intentional or accidental termination of the application, e.g. closing of the GUI window, system shutdown or power outage, the *key* extraction can be resumed from the last known state.

The number of threads used for computation (parallelization) is chosen automatically based on the capabilities of the CPU. On systems with Microsoft Windows the number of used threads is reduced by two, otherwise the user interface might get unresponsive.

2.2.2. Using the command line interface (CLI)

The command line interface to the go2key application is meant for expert users. It allows detailed parametrisation of input data and key ranges to be tested. The computation state, i.e. which key ranges have been already tested and which files have already been processed must be handled by the operator.

2.2.2.1. Basic workflow

- Extract voice data frames as described in chapter 3. Workflow on page 8
- List superframes and get overview of recorded data:

  `go2key list <path_to_json_file>`
- Select the *key-id* for *key* extraction
- Run the the application

  `go2key crack --key_id <selected_key_id> --auto_select <path_to_json_file>`
- Wait until key is found or no key is found after all keys have been tried.
- Decrypt voice: Place found *key* in decoder parameters of DMR decoder with appropriate *key-id*. Process the recording again with modified decoder parameters.

2.2.2.2. Command line options

crack runs the key extraction algorithm

list shows an overview of superframes in json file

benchmark runs the key extraction algorithm with random input data and shows the processing speed

2.2.2.3. Details

Options

```
>> go2key -h
usage: DMR go2key [-h] {list,benchmark,crack} ...

positional arguments:
  {list,benchmark,crack}
    list lists/shows superframes in record file
    benchmark benchmark cracking speed
    crack cracking of ARC4 encrypted voice

options:
  -h, --help show this help message and exit
```

Listing 2.1: Command line help to general options

Option 'crack'

```
>> go2key crack -h
   usage: DMR go2key crack [-h] [-i IDENT] [-v VALIDATION] [-a] -k KEY_ID [-s START_KEY] [-e END_KEY] [--
     num_threads NUM_THREADS] [--full] input

   positional arguments:
     input                 Record file with encrypted superframes.

   options:
     -h, --help            show this help message and exit
     -i IDENT, --ident IDENT
                           Superframe used for identification. Use command 'list' to see available frames.
     Not allowed if option '--auto_select' is
                           used. (default: None)
     -v VALIDATION, --validation VALIDATION
                           Superframes used for validation. It is allowed to use identification again. Not
     allowed if option '--auto_select' is used.
                           Allowed are 3 to 10 superframes. Recommendation: Use 6 or more for higher
     success probability. Format: Comma separated list.
                           Example: "0,2,3,8,11,44" (default: None)
     -a, --auto_select     Auto-select frames used for cracking. '--ident' and '--validation' are not
     allowed. (default: False)
     -k KEY_ID, --key_id KEY_ID
                           Crack this key id. (default: None)
     -s START_KEY, --start_key START_KEY
                           Start cracking with this key, has to be 0 or multiple of 256. (default: 0)
     -e END_KEY, --end_key END_KEY
                           Stop cracking with this key (exclusive), has to be multiple of 256 and > start
     key. (default: 1099511627776)
     --num_threads NUM_THREADS
                           Number of threads to use. If not given, the cpu count of the system is used (24)
     . (default: None)
     --full                Search full key space. Otherwise key cracking is aborted if the key is found. (
     default: False)
```

Listing 2.2: Command line help, "crack" option

Option 'list'

```
>> go2key list -h
   usage: DMR go2key list [-h] [-k KEY_ID] input

   positional arguments:
     input                 Record file with encrypted superframes.

   options:
     -h, --help            show this help message and exit
     -k KEY_ID, --key_id KEY_ID
                           Only show superframes with this key id.
```

Listing 2.3: Command line help, "list" option

Option 'benchmark'

```
>>go2key benchmark -h
  usage: DMR go2key benchmark [-h] [--num_threads NUM_THREADS]

  options:
    -h, --help            show this help message and exit
    --num_threads NUM_THREADS
                          Number of threads to use. If not given, the cpu count of the system is used.
```

Listing 2.4: Command line help, "benchmark" option

2.2.2.4. Example execution

```
>> go2key list ~/procitec/analysis suite 25.1/mem_prod/20230302/20230302-143420-552__0000000000_D01.json
Record Summary
Superframes    : 53
   Key ID   1  : 53
Detailed Frame List
Index   KeyID   AlgID  Errors    Complete          IV                                  Time
-------------------------------------------------------------------------------------------------------
    0       1       1       0           T    7f5d51c9      2023-03-02 14:34:01.503490+00:00
    1       1       1       0           T    869044a9      2023-03-02 14:34:01.863490+00:00
    2       1       1       0           T    423ca71f      2023-03-02 14:34:02.223510+00:00
    3       1       1       1           T    6f0fb018      2023-03-02 14:34:02.583490+00:00
    4       1       1       0           T    d792144f      2023-03-02 14:34:02.943490+00:00
    5       1       1       0           T    dd029b30      2023-03-02 14:34:03.303490+00:00
    6       1       1       0           T    e9e5ab51      2023-03-02 14:34:03.663510+00:00
    7       1       1       0           T    95cbee3d      2023-03-02 14:34:04.023490+00:00
    8       1       1       0           T    04abbb5f      2023-03-02 14:34:04.383470+00:00
    9       1       1       0           T    3964c692      2023-03-02 14:34:04.743490+00:00
   10       1       1       0           T    bdcadc47      2023-03-02 14:34:05.103490+00:00
...
   50       1       1       0           T    eadb7aba      2023-03-02 14:34:19.503430+00:00
   51       1       1       0           T    fb83aabe      2023-03-02 14:34:19.863450+00:00
   52       1       1       0           T    bf9d34c2      2023-03-02 14:34:20.223410+00:00

>> go2key crack -k 53 -a ~/procitec/analysis suite 25.1/mem_prod/20230302/20230302-143420-552
    __0000000000_D01.json
Start to crack the key with identification-frame=0 and validation-frames=[0, 1, 2, 4, 5, 6, 7, 52]
Quality of recorded data within computation : 100%
 0.006%    31654484 keys/s  remaining: 9:38:52  (0x0004100000)
 0.012%    31624105 keys/s  remaining: 9:39:23  (0x0008100000)
 0.018%    31659621 keys/s  remaining: 9:38:42  (0x000C100000)
 0.025%    31641042 keys/s  remaining: 9:39:01  (0x0010100000)
 0.031%    31658694 keys/s  remaining: 9:38:39  (0x0014100000)
 0.035%    30871366 keys/s  remaining: 9:53:23  (0x0016C00000)
 0.041%    30848803 keys/s  remaining: 9:53:47  (0x001A700000)
 0.047%    30859882 keys/s  remaining: 9:53:32  (0x001E700000)

...

 0.432%    29292910 keys/s  remaining: 9:50:52  (0x011A900000)
 0.437%    29203317 keys/s  remaining: 9:50:45  (0x011E200000)
 0.443%    29540925 keys/s  remaining: 9:50:35  (0x0122000000)
-------------------------------------------------
KEY FOUND: 0x0123456789  (match)
-------------------------------------------------
Aborting. Wait for worker threads to join.


FINISHED
CHECKED 4876926976 KEYS
CONTIGUOUS RANGE 0x0000000000 TO 0x0122200000 (excluding)
FOUND 1 KEY
0x0123456789  (match)
```

Listing 2.5: Command line execution example

2.3. Key calculation algorithm

For a DMR voice transmission, the AMBE2+™ codec from DVSI is used.

- Voice is encoded in 20 ms long voice frames.
- A voice frame consists of 49 data bits and 23 bits of forward error correction.
- 18 voice frames are combined into one super frame.
- A super frame thus consists of 882 data bits and encodes 360 ms of voice.

This frame structure is used for encryption and is relevant for the parameterization of the key calculation algorithm.

ARC4 is a stream cipher and used to encrypt and decrypt the data bits. A 40-bit *key* is used for this, therefore there are a total of $2^{40} - 1 = 1\,099\,511\,627\,775$ *key*s (0 not allowed). If the correct *key* is known, anyone can decrypt a voice transmission.

To crack the *key*, a brute force approach is used here and all *key*s are tested.

- If an incorrect *key* is used, you get pseudo-random bits.
- If the correct *key* is used, valid data is obtained.

To distinguish the single correct *key* from the all $2^{40} - 2$ wrong keys, the content of the transmission is analyzed. For a unique recognition, the calculation algorithm needs several super frames (3-10). These do not have to be consecutive or from the same transmission, but of course the same *key* must have been used. Each *key* is assigned a *key-id* and is always sent unencrypted. Therefore, the authorized receiving device knows which *key* was used to encrypt the data. Transmissions from the same source with the same *key-id* generally suggest that the same 40 bit *key* was used.

To speed up the *key* calculation, the algorithm is divided into 2 sections, which are called here identification and validation.

- the identification step uses a single superframe (identification superframe)
- the validation step uses multiple superframes (validation superframes)

During identification, a very fast calculation method discards a large proportion of incorrect keys. The remaining wrong keys are excluded within the validation step. Only the correct *key* is recognized as correct by the validation. The super frame for identification can also be used for validation.

Since bit errors can occur during radio transmissions, only those superframes should be used for *key* calculation which contain as few bit errors as possible. If the signal was received with poor quality, there may be too many bit errors in the data to calculate the correct *key*. In this case the algorithm returns no result. The number of bit errors is estimated based on the transmitted forward error correction bits. In addition, it is necessary that the superframes used for the *key* calculation contain different encryption initialization vectors.

In order to make a good selection of superframes from all received superframes, an automatism was implemented. However, it is also possible to specify the superframes manually. This specification must be a single identification superframe and between 3 and 10 superframes for validation. If possible, at least 8 validation superframes should be used. Since a superframe is transmitted every 360 ms, there should usually be enough data available.

It is very unlikely that the algorithm will return a false *key* (false positive probability) or fail to recognise the correct *key* (false negative probability). Both probabilities depend on the number of superframes used in the validation and were determined empirically with generated bit error-free test data. The false positive probability is smaller than 10^{-6} in all cases. The false negative probability is given in the following table depending on the number of validation superframes.

N	3	4-6	7-8	9-10
PN	<0.027	$<4.6*10^{-4}$	$<2.4*10^{-6}$	$<1.3*10^{-10}$

If, contrary to expectations, no *key* is found with error-free data, the search can be repeated with other super frames.

2.4. Keyboard Shortcuts (GUI)

Function	Shortcut
<Load file>	<Ctrl>+<O>
<Copy key>	<Ctrl>+<K>
<Save key>	<Ctrl>+<S>
<Manual>	<F1>
<Copy>	<Ctrl>+<C>
<Select all>	<Ctrl>+<A>
<Exit>	<Ctrl>+<Q>

Table 1: Keyboard Shortcuts

3. Workflow

In the following chapter we describe the complete workflow of using go2key with go2MONITOR and go2DECODE.

Note: the workflow for "DMR" and "DMR continuous" are identical but for simplicity only "DMR" is described here.

3.1. go2MONITOR

In case an emission has been detected and produced within go2MONITOR, the following steps are necessary for the production of a decrypted voice transmission.

- Start decoding the DMR signal to be decrypted in a go2MONITOR production channel

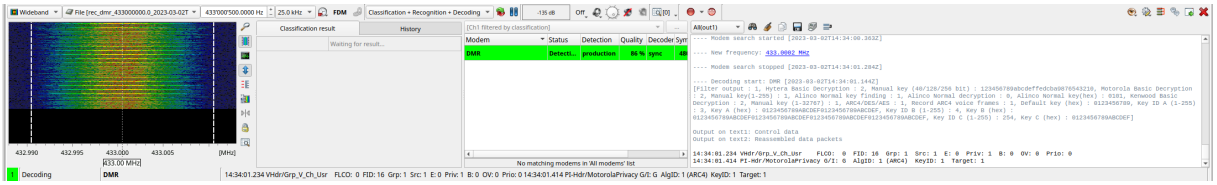


Figure 2: go2MONITOR production channel

- Right-click on the decoder to open the decoder parameters

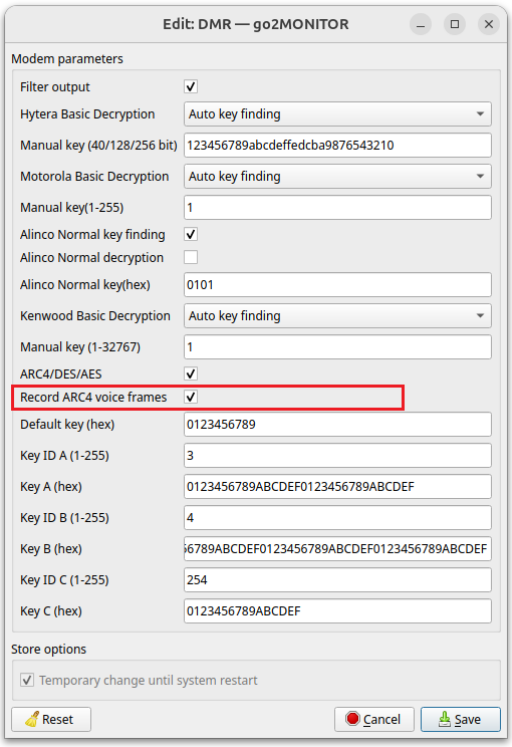


Figure 3: go2MONITOR decoder parameter editor with default DMR modem

and verify "Record ARC4 voice frames" is checked. This is the default setting.

- The signal has to be processed in order to extract voice frames and save them into a .json file. Files can be found in the results window by setting the filter to "Files"

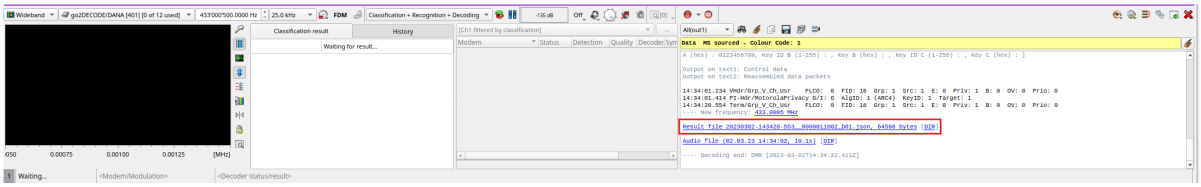


Figure 4: go2MONITOR production channel with voice frame recording results

- Open the file explorer by clicking on [DIR] and drag&drop the .json recording into go2key

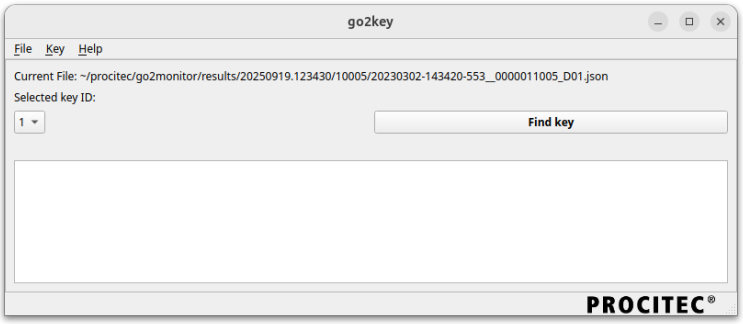


Figure 5: go2key with loaded voice frame recording

- Start *key* search with the *„Find key“* button

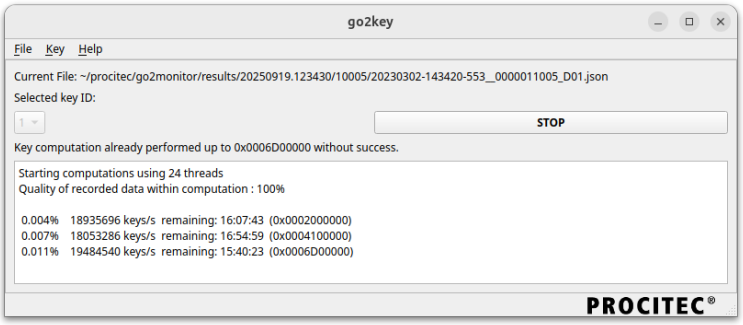


Figure 6: go2key with computation in progress

- Wait until go2key detects the correct key

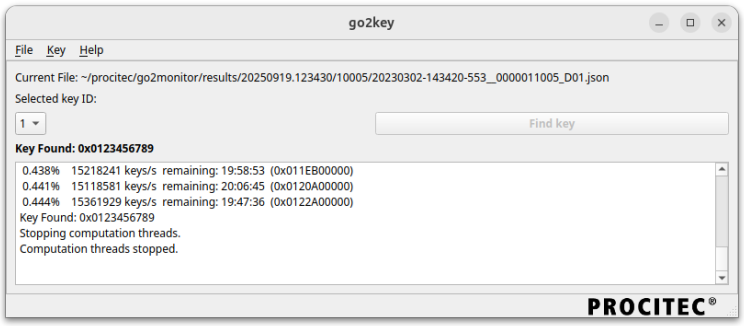


Figure 7: go2key with encryption key found

- The *key* **and** *key-id* need to be entered into decoder parameter form in go2MONITOR (note: omit the leading "0x" for the *key*).

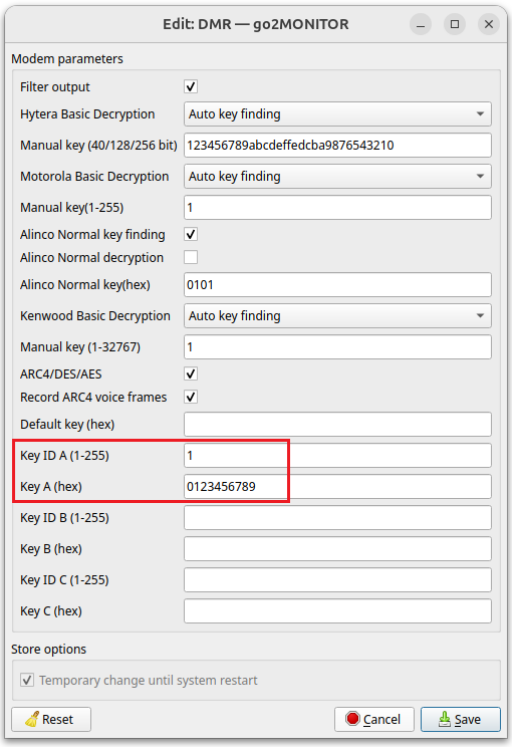


Figure 8: go2MONITOR decoder parameter editor with DMR, encryption key *and* key-id *are parametrized*

- Restart decoding, the signal is processed again, this time decrypting the voice data with the correct *key*

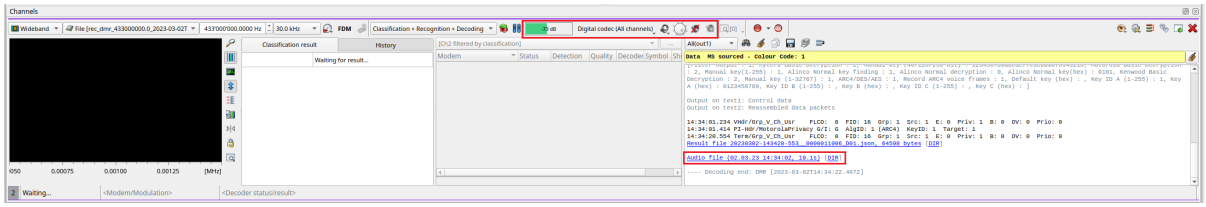


Figure 9: go2MONITOR production channel with audio results

The audio can be monitored during the processing through headphones or speakers. At the end of processing an audio file containing decrypted contents is reported in the decoder results window.

For details regarding modification of decoder parameters and handling of modified modem descriptions please refer to go2MONITOR user manual.

3.2. go2DECODE

In case an emission has been detected and produced within go2DECODE, the following steps are necessary for the production of decrypted voice transmission.

- Add DMR modem to the modem list.
- Open "Decod" tab for decoder parameters and make sure "Record ARC4 voice frames" checkbox is selected. This check box is selected by default.

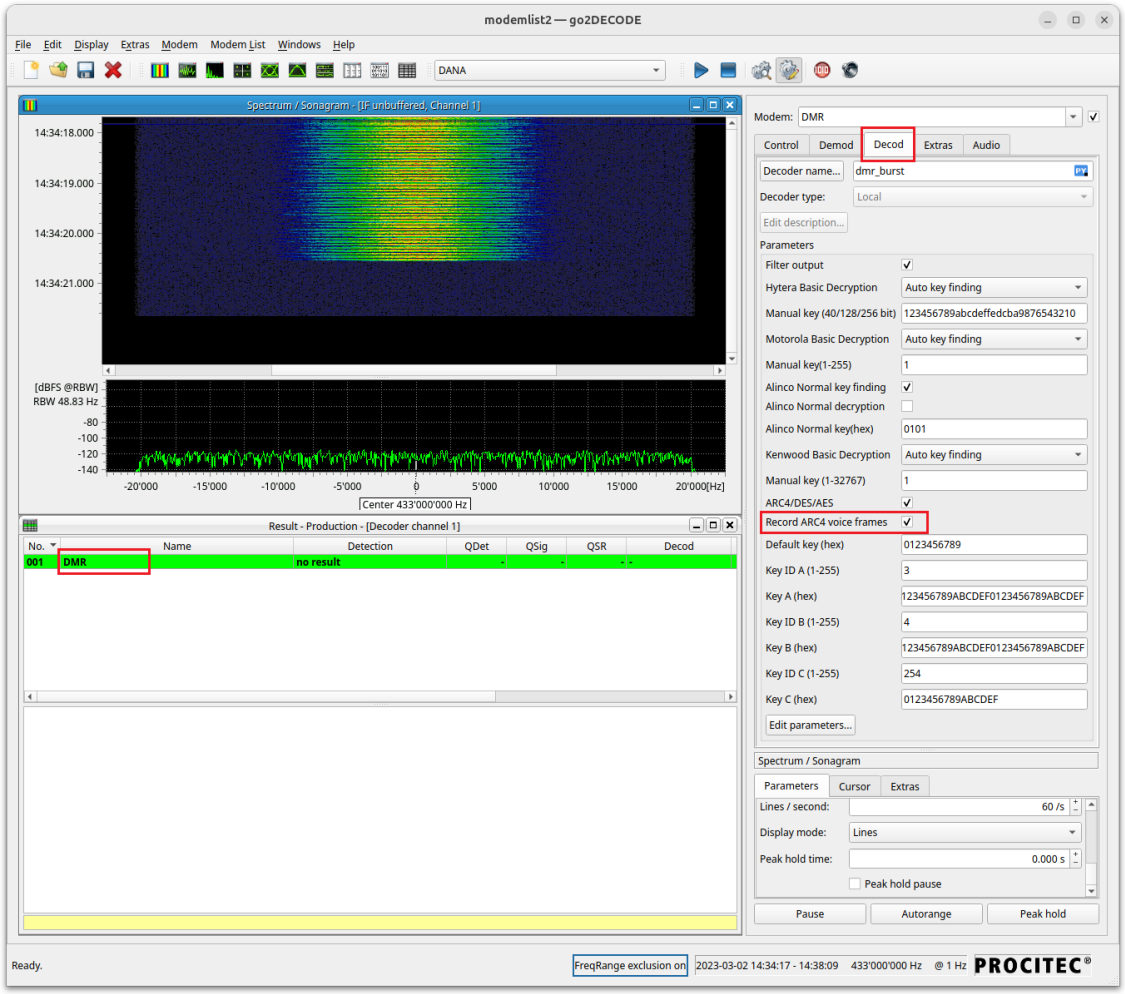


Figure 10: go2DECODE decoder parameters

- Process emission in order to create .json file with extracted voice frames.

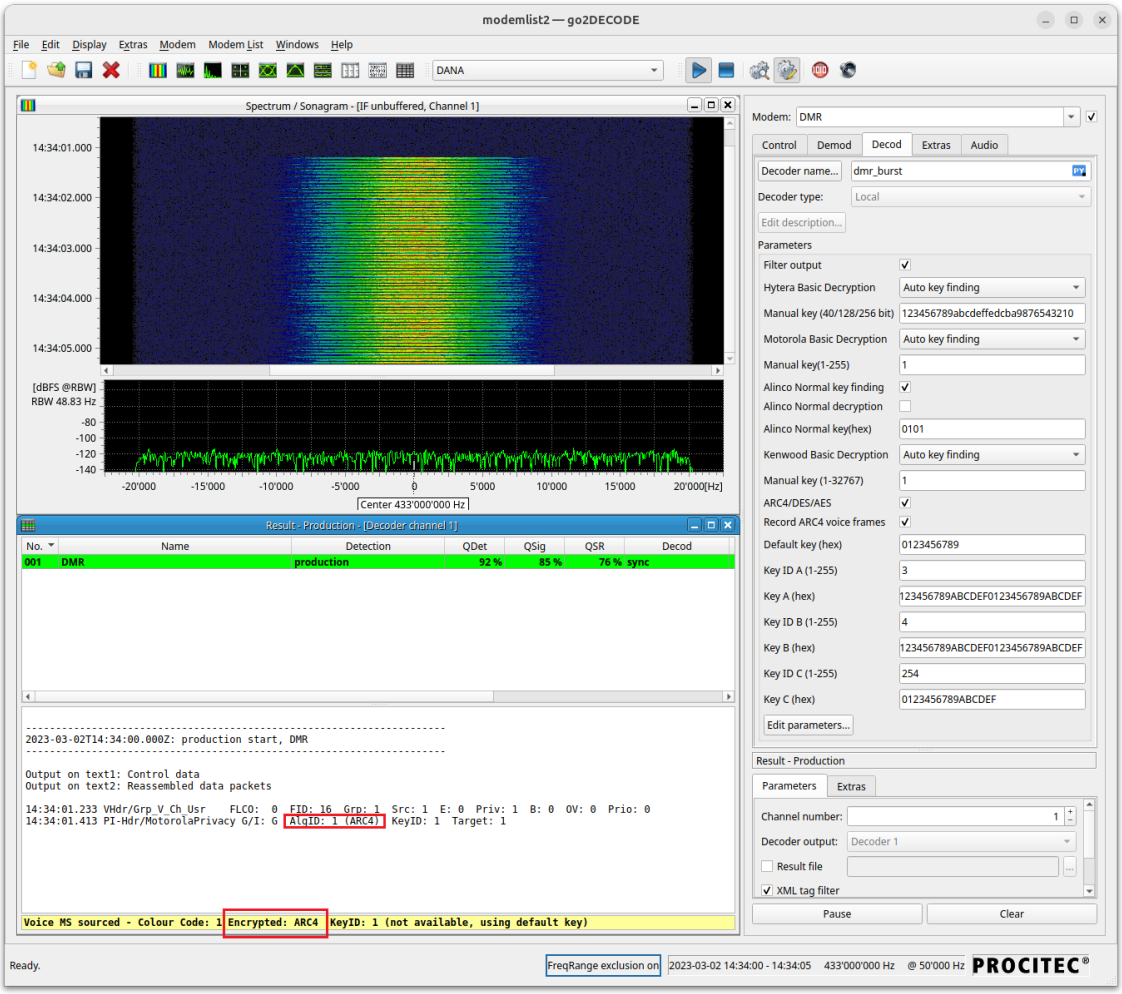


Figure 11: go2DECODE DMR signal production

- Find the corresponding voice frames extraction in a .json file.
  There are two options available:

  1. Deactivate "XML tag filter" in "Result display" parameters and find the path to the .json result file in the result raw XML, see Figure 12

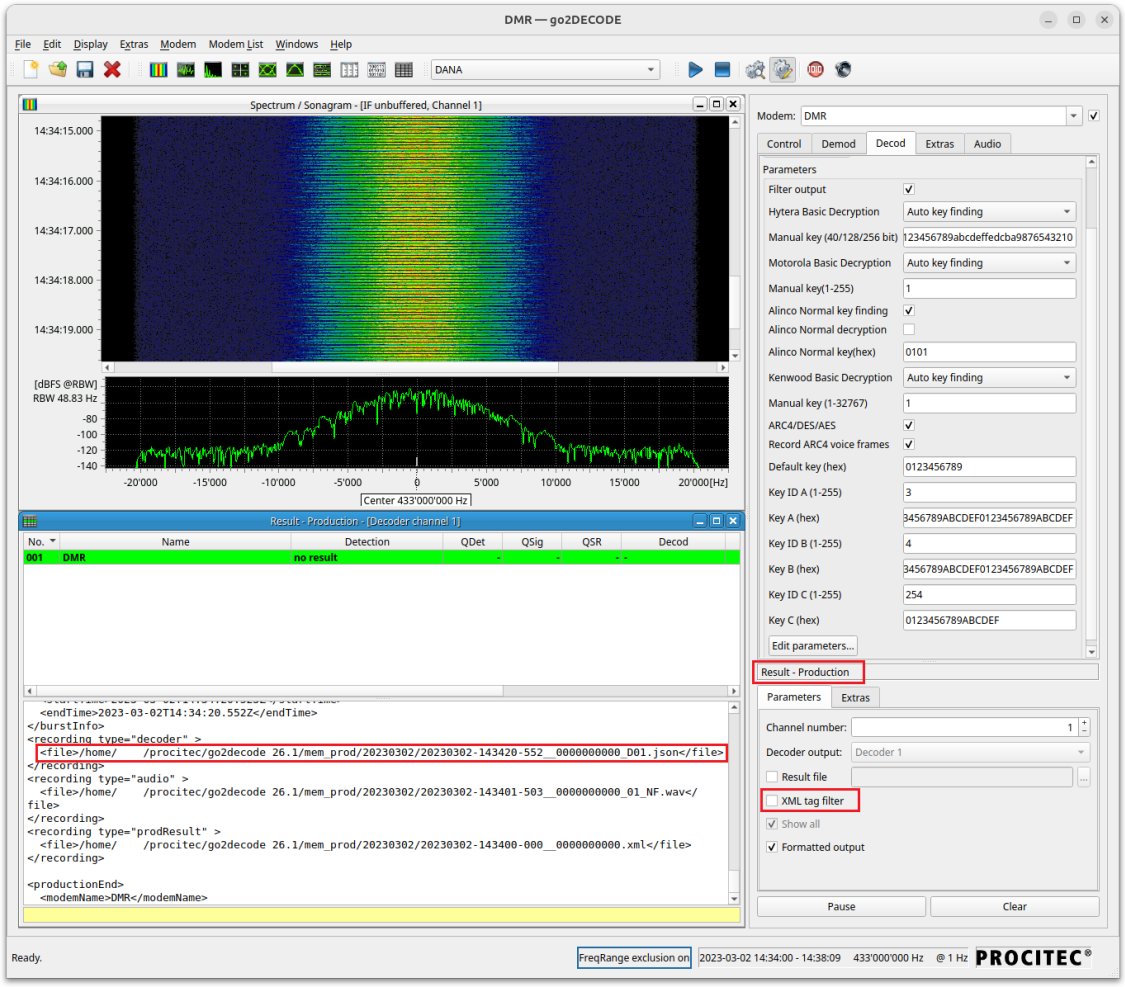


Figure 12: go2DECODE, access produced files

2. Open PMO from go2DECODE "Extras" menu. In PMO select date and "Binary Results" tab, see Figure 13.

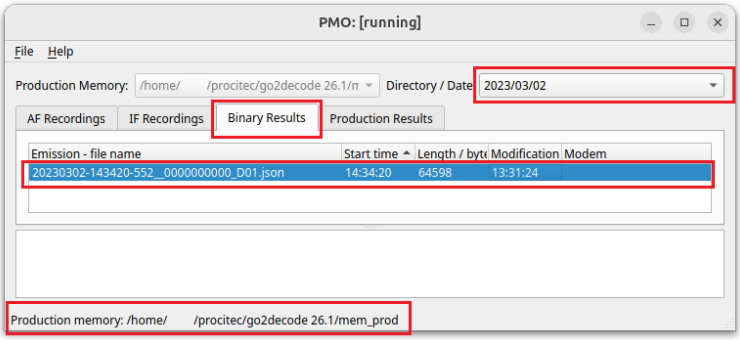


Figure 13: go2DECODE, use PMO to identify the .json file

- Open the .json file in go2key.

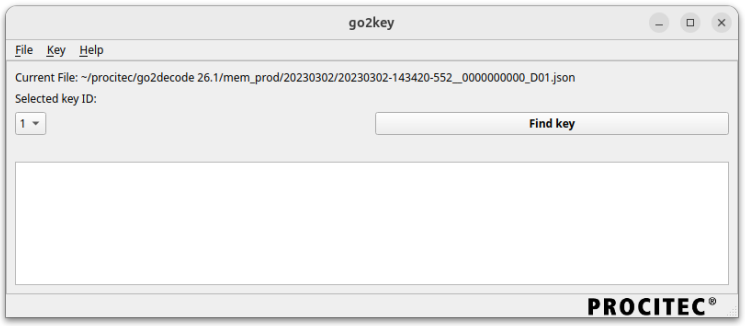


Figure 14: go2key with loaded voice frame recording

- Start *key* search with the *„Find key“* button.

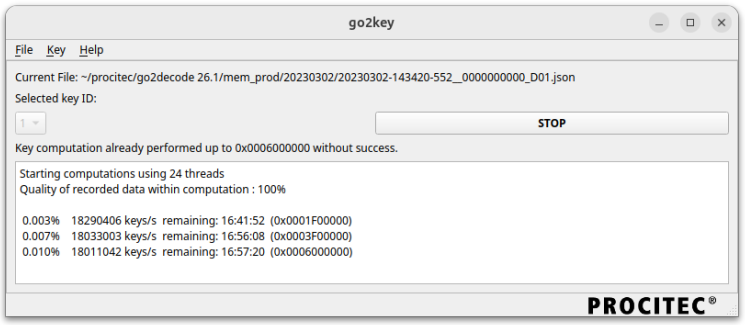


Figure 15: go2key with computation in progress

- Wait until go2key detects the correct *key*

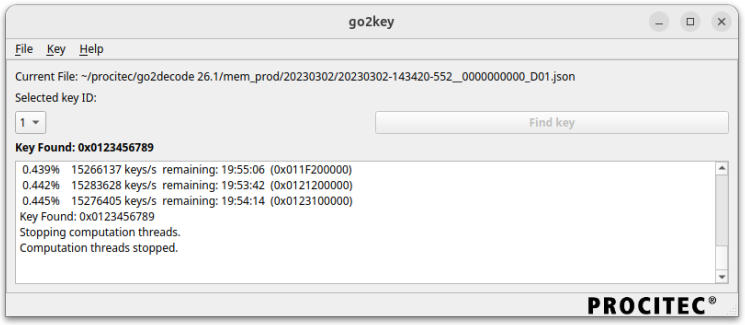


Figure 16: go2key with encryption key *found*

- The *key* **and** *key-id* need to be entered into decoder parameter form in go2DECODE (note: omit the leading "0x" for the *key*).

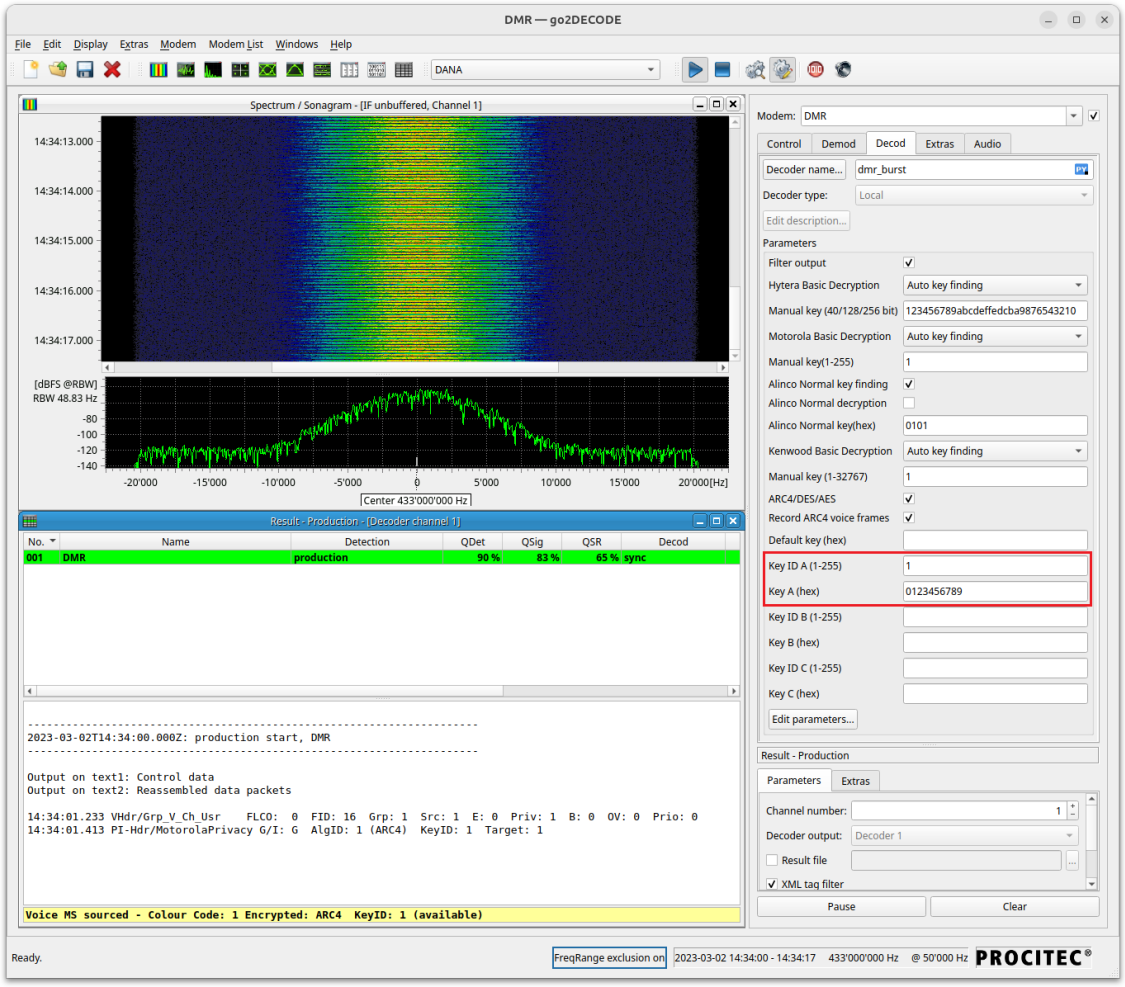


Figure 17: go2DECODE, decoder parameters for DMR with parametrized encryption key *and* key-id

- Restart decoding, the signal is processed again, this time decrypting the voice data with the correct *key*. For live playback please turn on "Audio out" from "Extras" menu or toolbar and set "Mode" to "Digital"

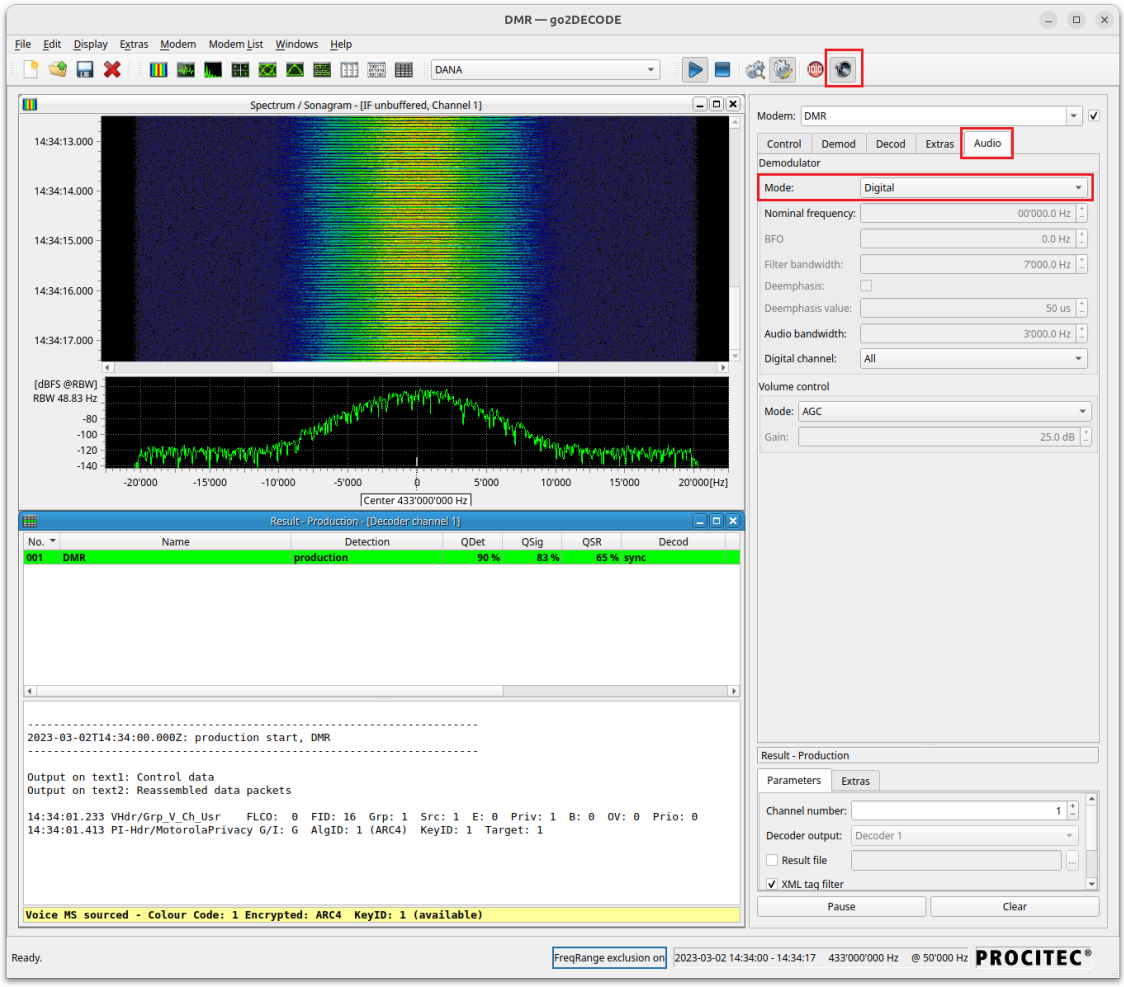


Figure 18: go2DECODE, audio playback and DMR modem with parametrized encryption key *and* key-id

- The decrypted audio recording can be found as a .wav file using PMO (see Figure 19) or by peeking into raw XML result stream (see Figure 12).

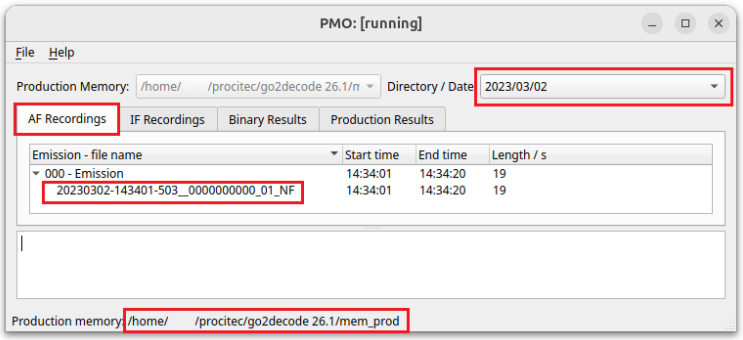


Figure 19: go2DECODE, use PMO to identify and play the .wav file

For details regarding modification of decoder parameters and handling of modified modem descriptions please refer to go2DECODE user manual.

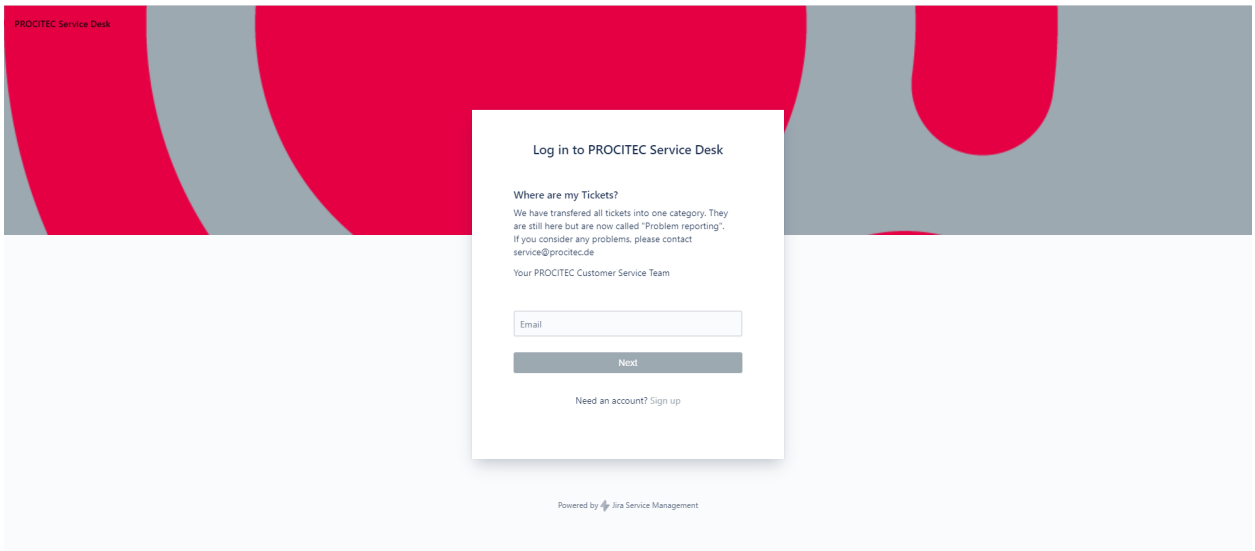
A. Support

Requests and suggestions?

All requests or suggestions regarding our go2signals product-range are very much appreciated; we would be delighted to hear from you.

Any questions? We are happy to assist you!

If you have any further questions, please do not hesitate to contact our Support Team for rapid assistance – just raise a service request at: **http://servicedesk.procitec.com**.



PROCITEC GmbH
Rastatter Straße 41
D-75179 Pforzheim
Phone: +49 7231 15561 0
Web: www.procitec.com
Email: service@procitec.com

List of Figures

List of Tables