sysjm

Description of the Program Family

CODING

Programs for Ciphering „in Place“

User Manual

Dipl.-Math. Jürgen Müller

Version 1.1 / Level: 01/12/2013

1 Description of the necessary Parameters of the Program CODING

1.1 Starting

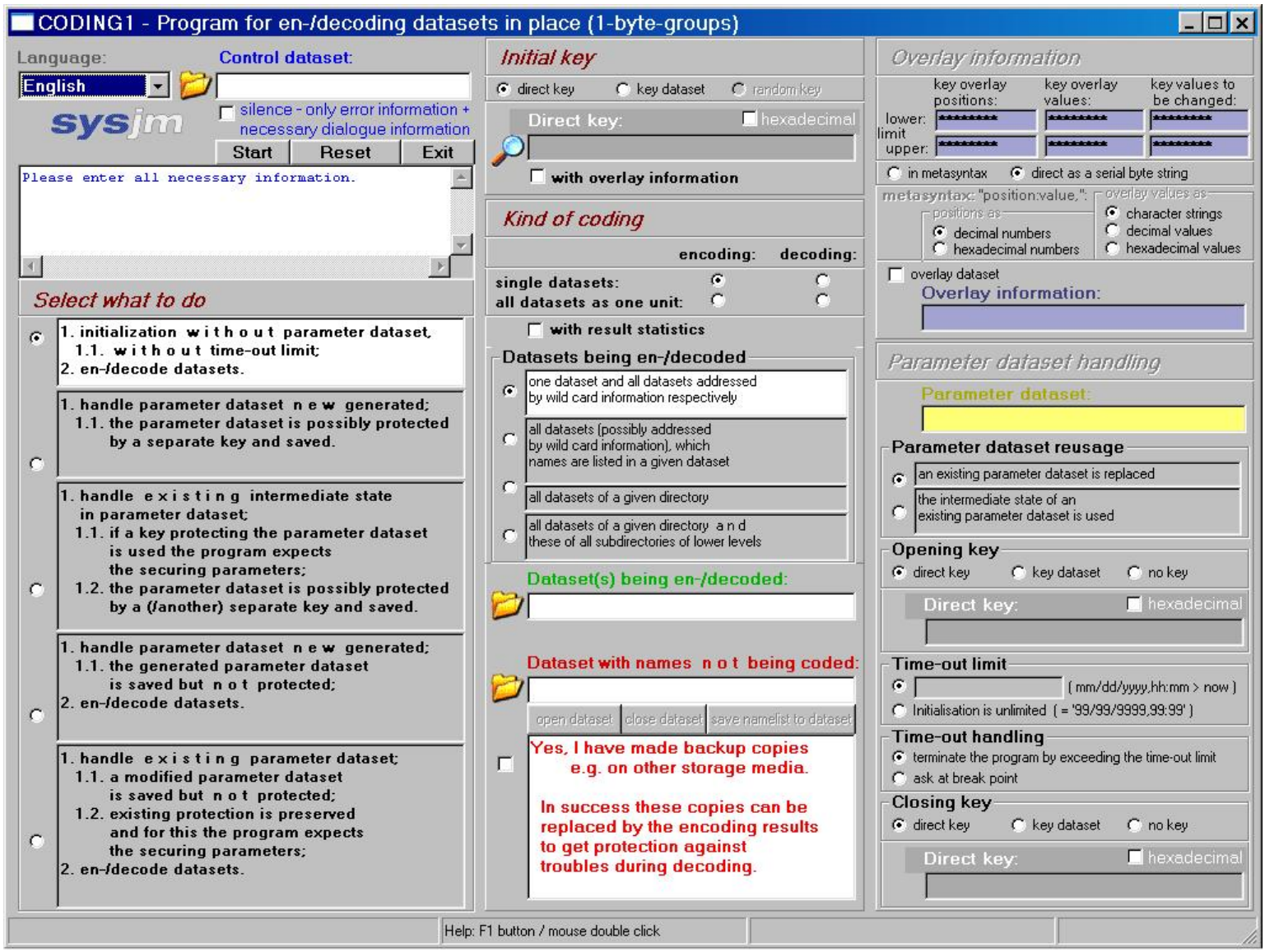
Bat: You can directly call one of the **DOS programs** (CODINGn.EXE, n=1,2,3) out of MS Windows (e.g. by double click), whereupon the program is started in a DOS box. Information applying to DOS programs below is marked „**Bat:**“ in front of it.

Dia: To start a **MS Windows program** version (applied information marked „**Dia:**“ in front of it) you have to copy the dataset CODING.EXE to the directory used to expand the Windows programs. Then you can call CODING.EXE to expand and store all necessary components.

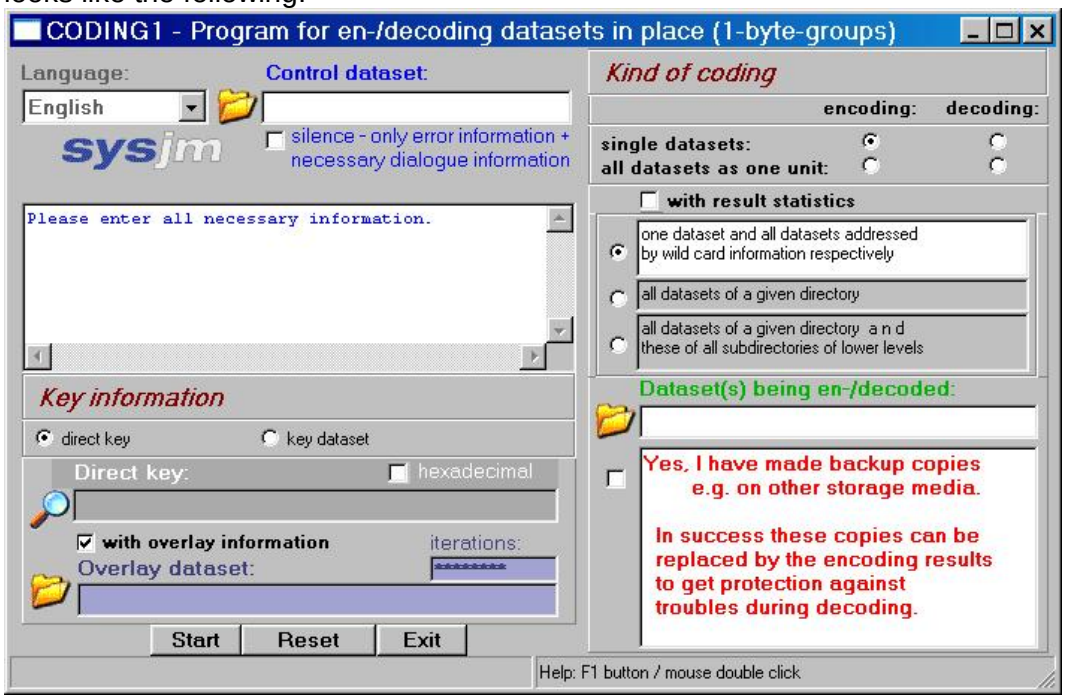
Please take into consideration that subdirectories HELPDBx (x=’D’ and x=’E’ at the moment) are constructed in this expanding directory covering help descriptions of each language.

After expanding CODING.EXE, you can start one of the corresponding MS Windows programs that means the CODING program in “simple” version (CODINGnS.EXE, n=1,2,3) or the CODING program in “enlarged” version (CODINGnE.EXE, n=1,2,3).

Having started the program, the main input mask – here the “enlarged” version of CODING1E (CODING2E/3E analogue) – is shown:



The main input mask of the “simple” version – here of CODING1S (CODING2S/3S analogue) – looks like the following:



1.2 Language

Bat: First you are asked the language the following dialog takes place (for example English and German):

```
Language:
  English    --> 0
  Deutsch    --> 1
```

Enter the digit corresponding the language you want to have.

Dia: Entering parameters, you can change the language at any time:

1.3 Identification

Bat: It follows the information of the identification page of CODING1 (CODING2/3 analogue):

```
01111010101101101110000000001010001011011001000011110000110010101100101001000100
1                                                                              0
0                                                                              1
1                                                                              1
1                                                                              0
0                                                                              0
0                                                                              1
1                                                                              0
0                                                                              1
1                                                                              1
1                                                                              0
0                                                                              0
0                                                                              0
0                                                                              0
1                                                                              1
0                                                                              1
0                                                                              0
1                                                                              1
0                                                                              1
0                                                                              0
1                                                                              1
0                                                                              1
1  * All rights reserved. The usage of this program is on own responsibility.  0
1                                                                              1
11110011111011011010011011111010010111011001000001101101101011001001111110000001 0
```

1.4 Restrictions

Bat:/Dia: Please notice that the existing version does not entitle to commercial use. If you need one of this programs for commercial usage, please turn to the author reachable under contact data at the end of this manual. Many thanks.

Please notice too that there is no absolute guarantee for the program in spite of highest quality check. If, in contrary to expectations, problems will be arise which provable are not caused by wrong services, the author will try hard to solve the problem immediately. Naturally your ideas to the program, the documentation, etc. are welcome every time.

Using the CODING method (see outline), there is **n o** possibility **a t a l l** to get a lost key so it is for the author (!). Please in case of doubt as a precaution use key or parameter datasets (see below) to avoid such problems. Many thanks to your comprehension.

Any **separate dataset is permitted to be bigger than 4 GB** to be processed correctly by the MS Windows versions of the CODING programs (the actual limit using MS Windows XP is 9,223,372,036,854,775,807 bytes).

„Simple“ and „enlarged“ version of each CODING dialogue program and the corresponding CODING batch program are complete compatible, but the programs CODING1, CODING2, and CODING3 are n o t compatible (!).

1.5 Common Things

Bat: After that you can finish the program during the input of program parameters every time by entering the value ‘9’ or without input of further characters by pressing the RETURN(¿) button (‘empty input’), so no dataset will be enciphered or deciphered.

Entering overlaying information (see below), also the values ‘0’ and ‘-1’ can do so in individual case.

You finish every input by pressing the RETURN(¿) button as usual.

Dia: Every time you can get help texts describing separate fields. By entering keyboard characters into a field, you can press the F1 button to get this help, by double click of the left mouse button you can get this help for all other fields.

Before using the „Start“ button, there is no action of enciphering or deciphering datasets or changing parameter datasets. In this phase and after working, the program can be leaved by pressing the “Exit” button at any time. Using the „Reset“ button, you can switch all fields to their default values at these phases.

Bat:/Dia: Using relative path information by entering dataset names, keep in mind that the starting directory is the directory the executable dataset or batch dataset (also called script or batch program) of the started program resides in.

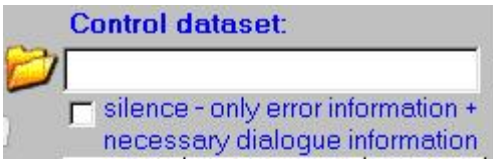
1.6 Output Dataset for Control Information

Bat: After the short display of the identification page, the starting information is written and the program asks you a dataset receiving control information:

```
En-/decoding program  C O D I N G 1  1.1  started.
  Started at: Th. 12/07/2006, 14:28:01.

Please enter the name of the dataset for control information,
 if only error and dialogue information shall be written to screen and
 to this dataset put the dataset name in round brackets ("(",")")
  (empty input terminates the program):
```

Bat:/Dia: Here you enter a dataset name with path information, if necessary, for a new or able to overwrite dataset which includes all relevant information (except of course security information !) of chosen parameters and of done program tasks.



Dia: To enter an existing dataset and its path information into the input field, you can click on the “folder” symbol on the left side, choose a dataset, and click on the “Open” button of the dataset choose dialogue. Afterwards you can still edit the name in the input field as you like.

Bat:/Dia: If (**Bat:**) this dataset name is covered in round brackets or (**Dia:**) the checkbox field is activated (**Bat:/Dia:**), only error and necessary dialogue information shall be passed on screen and transferred to this dataset. Normal information is written neither to screen nor to this control dataset.

The program works with the ANSI character set (MS Windows), it doesn’t work with the OEM character set (MS-DOS character set) which is taken into account reading this control dataset later.

1.7 Output of the Command Line Control Parameters

Bat: Now you can print out the detailed description of the control parameters in the command line calling the program into this output dataset for control information:

```
Shall a parameter description of this program be written
 to this dataset for control information ?
   0    - no, the program is to continue,
   1    - yes, after that the program is to continue,
   2    - yes, after that the program is to terminate.
Please enter the corresponding number:
```

This happens with the values '1' and '2', the value ’0’ skips this output. The values '0' and '1' continue the program.

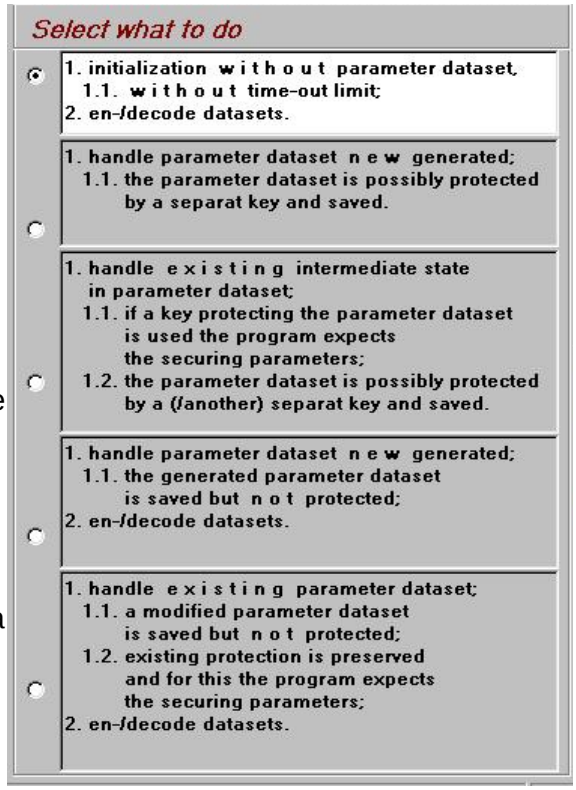
Please notice that the program only asks you this question, if you are asked the control dataset before. If the program gets the control dataset as a command line parameter, no such question is asked and no description of the command line control parameters is written.

1.8 Parameter Dataset Handling

Bat: If you continue the program, you are asked in which form the program should handle its job and therefore possibly limit the initialization process.:

```
W A R N I N G: The initialization of the coding parameters
               takes up several time(!).
Are you ready not to limit the initialization process
 and not to save intermediate results in a parameter dataset for coding ?
   0    - no, the initialization process shall be limited,
          parameter dataset handling is taken place,
   1    - yes, the initialization process remains  u n l i m i t e d,
          n o  protection of intermediate results takes place,
   9    - terminate the program.
Please enter the corresponding number:
```

Bat:/Dia: In principle the initialisation process especially for the program CODING3 can take a lot of time. If you work with big overlay datasets and/or a big iteration value (see below), there also may be a time problem for the programs CODING1 and CODING2. To reduce this time-consuming process, you can save the result of the initialisation process in a dataset – the parameter dataset – to use it at enciphering or deciphering time instead of the original key.



In extreme cases (especially program CODING3) its possible that the initialisation process takes up to several hours or even several days. Therefore you can save intermediate results of the initialisation process and continue with this initialisation process at later time using the intermediate results stored before. To do this, the initialization process has to be limited.

Dia: In the “simple” version of the program you have no possibility to handle parameter datasets to avoid higher complexity and details to be handled by the user.
In the “enlarged” version the qualities obtained by single questions in batch mode are fixed by selecting the task covering the actions to do. With that the parameter fields required to handle the actions are activated and such fields not required are deactivated. If necessary, this fixing can modified by other selection fields.

Bat: Entering the value ’0’, you tell the program to use a parameter dataset and possibly to limit the initialisation process. The value ’1’ is only useful if you want to handle datasets without using any parameter dataset.

1.8.1 Parameter Dataset

Bat: Choosing parameter dataset handling, the program asks you the name of the parameter dataset:

```
Please enter the name of the parameter dataset
  (empty input terminates the program):
```

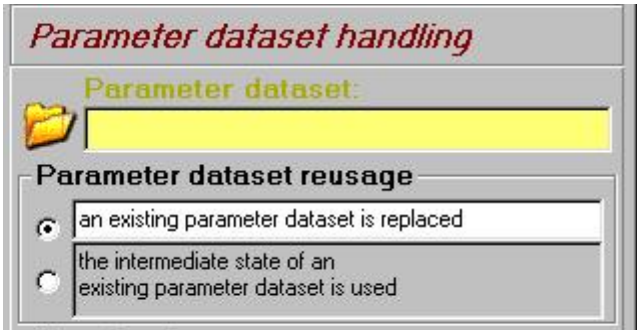
Bat:/Dia: Please notice that the size of a parameter dataset is calculated as:

*parameter_dataset_size := 512 + n * bytes of byte-group * 2 ** (8 * bytes of byte-group),*

with n=3 storing an intermediate result and n=2 storing a result of a parameter dataset, i.e. 2,048 / 393,728 / 150,995,456 (n=3) and 1,536 / 262,656 / 100,663,808 (n=2) bytes respectively using CODING1/2/3.

A parameter dataset used in the current program task is not en-/deciphered and that is why it is not destroyed (see about it 1.10. Exception Dataset below).

Dia: You can put a dataset name (with path name) into the input field of parameter datasets using the same method as described in 1.6. Output Dataset for Control Information above. The default extension with it is „par“. To help you especially by entering long dataset names, the program displays the place of the cursor or the range of places being marked at the left side of the status bar being at the bottom of the input mask. If the parameter dataset field isn’t selected i.e. hasn’t the focus, the field information is displayed as stars. If the parameter dataset field is selected, the contents of this field is only visible if the mouse pointer is located or moved in the central region of this field. If the mouse pointer slowly leaves the central region of this field, then the information is hidden (technical: the more suitable „OnMouseLeave“ event isn’t supported for such fields today using the relevant development system). Regardless of that marked field parts are always visible.



1.8.2 Using Parameter Dataset

Bat: In the following the program asks whether the named parameter dataset should be new generated or its intermediate state should be continued:

```
In which form the existing parameter dataset shall be used ?
   0    - the parameter dataset is  r e p l a c e d ,
   1    - the intermediate state of the this parameter dataset is used,
   9    - terminate the program.
Please enter the corresponding number:
```

1.8.3 Ciphering after Parameter Dataset Handling

Bat: If the program was able to open the parameter dataset successfully, you have to answer the question whether datasets have to be ciphered if initialisation of the parameter dataset is finished:

```
Shall coding of datasets take place after initialization ?
   0    - no, there is only the processing of the parameter dataset,
   1    - yes, after initialization coding of datasets takes place possibly,
   9    - terminate the program.
Please enter the corresponding number:
```

Dia: Using the “enlarged” version, these information are already fixed by selecting the task (see 1.8. above).

1.8.4 Time Limit

Bat: If the state of the parameter dataset indicates an intermediate state, you are asked a time limit the initialisation process is terminated automatically or manually or a new time-out limit is enquired:

```
Approximately until which time the process for initialization of the
 coding parameters shall be terminated and the intermediate results
 shall be written to the parameter dataset »parameter_dataset«
 to continue the process at later time ? (empty input terminates the program):
(in form of 'mm/dd/yyyy,hh:mm'; 99/99/9999,99:99 =unlimited):
```

Choosing „unlimited“, you are invited to confirm your selection because of safety reasons:

```
W A R N I N G: The initialization of the coding parameters
               takes up several time(!).
Are you ready not to limit the initialization process ?
   0    - no, the initialization process shall be limited,
   1    - yes, the initialization process remains  u n l i m i t e d,
   9    - terminate the program.
Please enter the corresponding number:
```

Dia: Using the “enlarged” version, you select „unlimited“ or you enter a time-out limit. Please notice that the program is warning you if the time-out limit is reached in less than 10 minutes till starting the program task by “Start” button.

1.8.5 Action reaching Time-out Limit

Bat: If you have inserted a real time-out limit, you are asked the action to do by exceeding this limit:

```
After exceeding the given time limit the program shall
   0    - terminated automatically,
   1    - be continued possibly with new time limit asked for.
   9    - Terminate the program now.
Please enter the corresponding number:
```

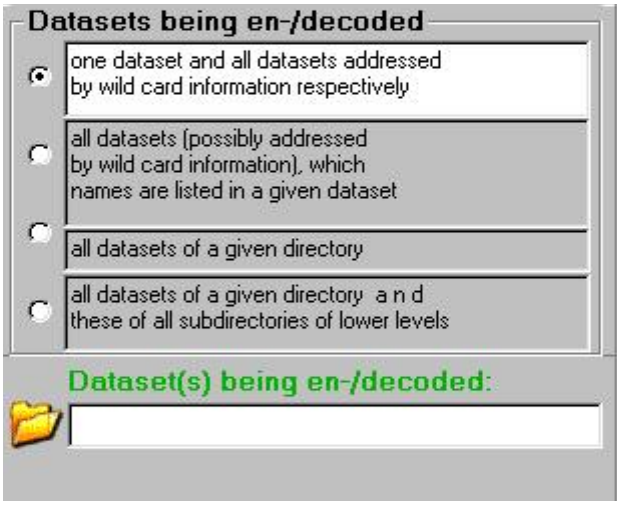
This closes the input of the parameter dataset handling information.

The following information about the processed datasets, the exception dataset, the processing form, and the statistics are only asked if the question about ciphering data mentioned above is positively answered or if no parameter dataset handling is wanted.

1.9 Information to Datasets to process

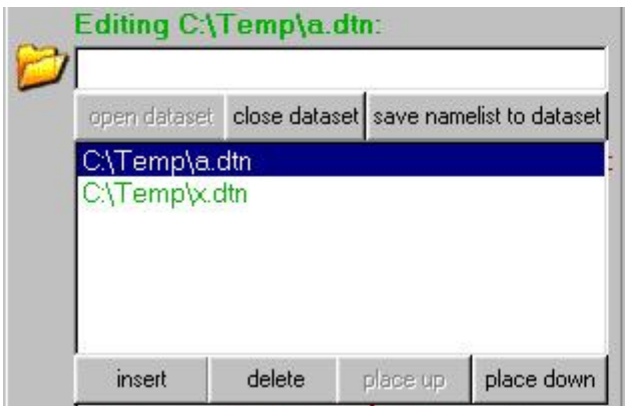
Bat: If not only a parameter dataset handling takes place, you are asked in which form you want to give the dataset(s) which shall be enciphered or deciphered:

```
Which datasets shall be processed?
   1  one dataset and all datasets addressed by wild cards respectively,
      which name is given in the following input
   2  all datasets (possibly addressed by wild card information),
      which names are listed in the dataset given in the following input
   3  all datasets of the directory given in the following input
   4  all datasets of the directory given in the following input
       a n d  these of all subdirectories of lower levels
   9  terminate the program.
Please enter the corresponding number:
```



Bat:/Dia: (**Bat:**) Entering the value ’1’ or (**Dia:**) choosing the first section (**Bat:/Dia:**), you can give the dataset or datasets processed in the following with path information, if necessary, in a direct way, where you can use the special character ‘?’ for any character and ’*****’ for any group of characters in a dataset name (”wild-card information“) as usual in MS-DOS to choose many datasets.

(**Bat:**) Entering the value ‘2’ or (**Dia:**) choosing the second section only possible in the “enlarged” version (**Bat:/Dia:**) tells the program, you want to give a dataset in the following which itself covers dataset information with one dataset item per line or one combination of dataset and directory information covered in "<",">" brackets per line to choose all matching datasets within a directory tree beginning with the given directory identifying datasets to process. I.e. you have prepared a dataset with dataset information of that datasets which shall be processed in the actual run. Please pay attention to the fact that **datasets addressed by many items shall be ciphered repeatedly !** If you don’t want to process these datasets in single mode, you can ignore this fact. But in any case the processing time will be increased.



Dia: Choosing a dataset covering dataset names in the “enlarged” version, you can edit the contents of this dataset by selecting the “open dataset” button. Doing this, the dataset contents is read into a list and you get the items mentioning one or more datasets repeatedly. You can exclude these entries individually. Further you have the possibility e.g. using the dataset choosing dialogue to enter single datasets to the dataset name field for that used as an editing field and modify it, before you can put it on the list of datasets (“insert” button). To insert an entry, it is put in place above the marked item. Selecting the “delete” button, you can remove a marked item and put it into the editing field. Possibly you can change it here and put it on the list again using the “insert” button. Every step putting an item on the list leads the program to check all list entries for overlapping the entry being inserted. To close work and pass the dataset list to the dataset opened, you have to select “save namelist to dataset” button followed by the “close dataset” button. If you would reject the dataset list, you can negate the question about saving the list after selecting the “close” button. The default extension with such datasets covering dataset names is “dtn”.

Bat:/Dia: In many cases you want to process all datasets of a directory or of a directory **a n d** all subdirectories of any level. This is possible by (**Bat:**) entering the value ‘3’ or (**Dia:**) choosing the third section (**Bat:/Dia:**) for datasets of exactly one directory and by (**Bat:**) entering the value ‘4’ or (**Dia:**) choosing the fourth section (**Bat:/Dia:**) for all datasets within a directory tree. In these cases the program in the following asks you the directory name, which can be completed by path information as known by MS-DOS.

Dia: By clicking on the “folder” symbol, you start the directory choose dialogue, then you can select a directory and you can edit this directory name in the editing field shown.

Bat: Depending on the input value you are asked as follows:
’1’:
```
Please enter the identification of the dataset(s), which shall be processed
  (empty input terminates the program):
```
’2’:
```
Please enter the name of the dataset, in which the names
 of all datasets to process are listed
  (empty input terminates the program):
```
’3’, ’4’:
```
Please enter the name of the directory,
 which contains the datasets to process
  (empty input terminates the program):
```

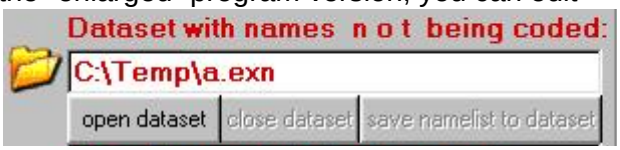
1.10 Exception Dataset

Bat: After that you are asked to enter the name of an exception dataset:

```
Please enter the name of the dataset, in which the names
 of all datasets  n o t  to be processed are listed
 (empty input = no exception dataset):
```

Bat:/Dia: This dataset comprises one dataset item per line (possibly with ”wild-card information“) or one combination of dataset and directory information covered in "<",">" brackets per line identifying datasets **n o t** being processed, even if these datasets are mentioned before as datasets to process. You can prepare a dataset with such dataset items before if necessary.

Dia: Choosing a dataset covering dataset names in the “enlarged” program version, you can edit the contents of this dataset or create this dataset by selecting the “open dataset” button as described by datasets to process (see 1.9. above). The default extension with such exception datasets covering dataset names is “exn”.



Bat:/Dia: The following datasets are **n o t enciphered/deciphered** by the current program task, even if they are not mentioned by the exception dataset to preserve the possibility of deciphering datasets enciphered before:

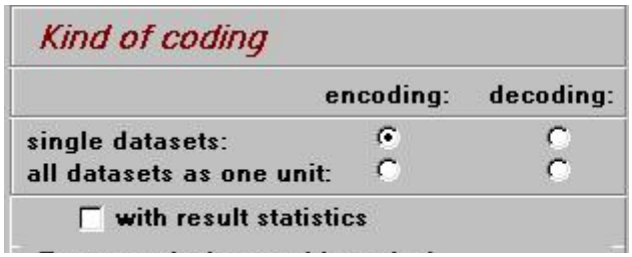
- the module dataset containing the called program just running,
- the dataset for control information,
- used datasets containing the names of the datasets to be processed, if they are given (e.g. the exception dataset too, if it is given here),
- used datasets containing the (opening or closing) key, if they are given,
- used datasets containing the overlay information, if they are given,
- used parameter datasets, if they are given.

Not using an exception dataset, the program assumes that all other datasets mentioned before as datasets to process have to be processed in fact.

1.11 Processing Form

Bat:/Dia: Because of that the datasets which shall be processed are fixed. But the question is, in which form this shall take place:

Dia:



Bat:

```
In which form the dataset(s) shall be processed?
   1  encoding of single dataset(s)
   2  decoding of single dataset(s)
   3  encoding of dataset(s) as  o n e  unit
   4  decoding of dataset(s) as  o n e  unit
   9  terminate the program.
Please enter the corresponding number:
```

Bat:/Dia: The terms “enciphering“/“encoding“ and “deciphering“/“decoding“ just go without saying (although these operations can be used in transposed form !). But it’s of safety interest what it is about the term “as one unit”.

First datasets enciphered as single datasets can be deciphered separate, i.e. not depending on other datasets. If you enciphere datasets as one unit, exactly these datasets have to be deciphered in same sequence to get the original contents. It’s not possible to deciphere such datasets separate (!).

The benefit of enciphering datasets as one unit is that the key parameters internally used by the program aren’t put back to the initial condition for every dataset. Rather all involved datasets are processed as one big dataset i.e. data blocks may be processed in a file-crossing way, so un-authorized persons have much more difficulties by making an attempt to deciphere such datasets.

Processing a directory first the datasets of the directory are processed in alphabetic order. Second the subdirectories of a directory are processed if required. These subdirectories are processed in alphabetic order too if all datasets should be processed as one unit. Otherwise the sequence of these subdirectories in internal tables of the operating system determines the sequence of the processing (for reasons of simplification). Not depending on that, a directory tree is excluded in accordance with a dataset-directory-combination entry of the exception dataset if it isn’t able to have datasets being processed in such a directory tree.

As disadvantage remains that directories enciphered as one unit may **n o t** be modified or have to transferred exactly in original condition before starting the deciphering process. You have the possibility to do this e.g. by creating a dataset with all datasets to be involved in enciphering sequence (see 1.6. dataset for control information above) and using this as a dataset covering datasets to be deciphered (see bellow). So the ciphering algorithm only works on the contents of each dataset, it doesn’t change any directory or similar information.

This fact may be used to get **further safety**. If you exchange the dataset on the first position for example by renaming, even using the correct key there is no possibility to get the original contents of the ciphered datasets without inverse renaming at all!

1.12 Statistics

(**Bat:**) Answering the following question positive or (**Dia:**) selecting the corresponding checkbox (see above) (**Bat:/Dia:**), you can get summarized statistics about the ciphering result at the end of a program task. Such statistics show you the quality of the definite ciphering result on enciphering, it’s uninterested by looking at the process of ciphering itself.

Bat:

```
Shall statistics of result data be made?
   0  no, result statistics shall not be made
   1  yes, result statistics shall be made
   9  terminate the program.
Please enter the corresponding number:
```

1.13 Key Information

Bat:/Dia:

The course of ciphering data is identical for all CODING programs:

- The user gives the program an initial key entered by the keyboard (or possibly by mouse moves using a parameter dataset) or using a key dataset.
- Possibly the user gives so-called overlay information and corresponding overlay parameters (see 2. below) as well.
- Using these initial data, the program calculates a set of parameters used to control the ciphering process and possibly stored in a parameter dataset.
- If a parameter dataset is used, you have the possibility to protect the usage of this parameter dataset against unauthorised persons using a separate key.

The protecting part of a parameter dataset is interpreted as a common data block and therefore, if necessary, is protected with analogue methods also used by ciphering data. If using parameter datasets, you are able to use any random key as initial key. In these cases the parameter dataset protection key takes the security function first associated with the initial key.

Technical that part of the parameter dataset possibly being ciphered has an internal structure, if enciphered, not being distinguishable from an analogue parameter dataset part not enciphered. In parameter datasets especially transformation tables are only present in form of their generation parameters, the transformation tables themselves have a much to concise structure. Therefore there is no possibility to decide whether a deciphering trial of a protected parameter dataset results in an original parameter dataset that means it is successful or not. Also parameter datasets “wrong” deciphered are usable. The programs **cannot** distinguish between “correct” and “wrong” deciphered parameter datasets.

ATTENTION: If you save parameter datasets comprising intermediate states, please take into account that these datasets contains information concerning the security, even if it doesn’t include the final ciphering parameters totally. Such datasets for example can hold your initial key not included in the final ciphering parameters and not being deduced form these final parameters (!). For your safety you should encipher such datasets with a “random” key **before deletion**.

The following information concerns all facts to deal with key information. The messages shown here are in accordance with the program CODING1. Replacing 256 or 512 characters with 131,072 or 262,144 characters using CODING2 and 50,331,648 or 100,663,296 characters using CODING3, you get the maximum characters being shown and accepted by these programs.

So it’s easy to see that for CODING2 and more necessary for CODING3 you have to use key datasets or overlay datasets in direct mode to get approximately the entire range of keys being possible.

If a key covers more than 256 /131,072 /50,331,648 key characters, the surplus characters are ignored. If on the other hand less than this number of key characters are given, then the given key characters are duplicated multiple and the result string is truncated to this number of characters for further use. Nevertheless you should not choose to less and homogeneous characters as key characters, because this, in extreme case (e.g. only hexadecimal null characters), could be result in no enciphering (!).

Not having a rare extreme case, the results of enciphering, also using short keys, have no characteristics with regard to the key length at all. Simply by searching the key a third person first will test a shorter key probably, before it takes a longer input key into account – and gives up then.

In principle all special characters excepting line end characters (mostly generated by RETURN(¿) button or <carriage return><line feed> or $0D_{hex}$ plus $0A_{hex}$ character) are interpreted as components of a key of ASCII characters. During the key input entering hexadecimal characters, line end characters may also be used to split key information to more than one line. (**Bat:**) Entering

ASCII characters by the keyboard, line end characters in general are ignored or interpreted as end of key information if combined with an empty line. (**Bat:/Dia:**) Line end characters in datasets comprising an ASCII key are interpreted as components of the key too.

1.13.1 Query to Parameter Dataset Security

Bat: If a parameter dataset is used, if the initialisation of the parameter dataset is finished, and if **no** cipher**ing** of data should take place, you are asked to protect this parameter dataset:

```
Shall the parameter dataset be protected by a separate key?
  '0'   - no, the parameter dataset shall not be protected by a key,
  'S'   - yes, the parameter dataset shall be protected by a separate key.
  '9'   - Terminate the program now.
Please enter the corresponding number or character:
```

If you wish a protection with a separate key, you are asked for information about the parameter dataset closing key.

Bat:/Dia: Using this parameter dataset, in future the program will ask you this key as parameter dataset opening key, unless you remove this protection key by answering the above question in the negative during a separate program task without ciphering data using the same parameter dataset.

The following information about the “initial key”, if not mentioned otherwise, also refers to “parameter dataset opening and closing key” analogous.

Dia: As discussed in 1.8. Parameter Dataset Handling using the “simple” version of the program, you can only work with initial key information, also a random key isn’t useful without parameter dataset.

In the “enlarged” version of the program selecting the action (see 1.8.) already fixes the key information needed in fact. So you can enter only this information.

1.13.2 Character Set

Bat:

```
Which character set you want to choose to enter the initial key
   1  ASCII characters (max. 256 characters)
   2  hexadecimal characters
      (max. 512 characters "0" to "9" and "A" to "F" respectively)
   9  terminate the program.
Please enter the corresponding number:
```

Bat:/Dia: In this dialog it’s about the character set to enter and read in key information. To get the possibility to enter all possible characters apart from the “normal” character set (ASCII character) which can be entered by the keyboard, you can choose another character set (hexadecimal character), so any two characters of which are combined to one key character by the program. With this character set key characters can be generated which can not or only laboriously generated in a direct way by keyboard buttons.

Dia: Using the checkbox field on the right above the key input field, you can decide if you want to enter or read key information direct or using a dataset in hexadecimal form or via ASCII characters. In dialogue entering a direct key you have the possibility to change the character set at any time i.e. from ASCII characters to hexadecimal characters and vice versa. With it ASCII characters automatically are expanded to its hexadecimal values or in opposite direction hexadecimal values are condensed to ASCII characters. The latter possibility only exists if there is no character value 00_{hex} in the key (internally interpreted as the end of a string).

Further entering a key, it's possible that leaving the input field by a mouse click problems can arise occasionally, which can be avoided if you finish a direct key by pushing the return button. In spite of appropriate efforts of the author it was impossible to remove all uneven patches up to now.

1.13.3 Input Mode

Bat:

```
In which way you want to enter the initial key?
   0  twice via keyboard  w i t h o u t  echoing the input on the screen
       (second time for controlling!)
   1  once via keyboard  w i t h  screen control
   2  via a dataset containing the key
   9  terminate the program.
Please enter the corresponding number:
```

Dia: In the selection area above the key input field you can decide to enter the key in a direct way (keyboard) or to read the key using a dataset. Working with the "enlarged" version and using a parameter dataset, you can additionally generate a random key by mouse moves or you can leave the parameter dataset unenciphered respectively.

1.13.4 Keyboard Input

Bat: If you want to enter the key characters by keyboard, you get the possibility with (mode='1'):

```
Please enter your initial key in hexadecimal/ASCII characters
 (end of key: press only return-button in line):
```

or with (mode='0'):

```
Please enter your initial key in hexadecimal/ASCII characters
 (end of key: press only return-button in line) once:
```

and with:

```
Please enter your initial key in hexadecimal/ASCII characters
 (end of key: press only return-button in line) twice (control input):
```

Dia: By entering a direct key, there first is a "magnifying glass" symbol on the left side of the key input field. This symbol shows that the corresponding key is not entered or is not acknowledged by repetition of the key. Any character of this field is represented by a star-character. There is no possibility to inspect key characters of this field in a direct way. To give you an orientation especially using long keys, the program displays the place of the cursor or the range of places being marked at the left side of the status bar as already described above.

If you have finished the input of your key (also see the remarks in 1.13.2. above), you can inform the program by pushing the return button or making a mouse click outside the key input field and outside the checkbox field for hexadecimal input. Then the program will ask you to repeat the key. If you answer in the negative, the key input is finished and the „magnifying glass" symbol is preserved as a memory. You can click this "magnifying glass" symbol at any time to carry through the confirmation of the key. If you answer in the affirmative, you get the possibility to confirm the key by repetition under field heading "Repeat key". Here too you can inform the program about the end of the input by pushing the return button or making a mouse click outside the key input field and outside the checkbox field for hexadecimal input.

If both inputs are identical, a "tick in green circle" symbol is shown indicating that the key input is finished successfully. Any additional change of the key is interpreted as a new input of the key and therefore has to be confirmed.

1.13.5 Input via Key Dataset

Bat: If you want to use a dataset comprising the key (mode='2') with:

```
In which way you want to enter the dataset name containing the initial key?
   0  twice via keyboard  w i t h o u t  echoing the input on the screen
       (second time for controlling!)
   1  once via keyboard  w i t h  screen control
   2  via a dataset containing the key
   9  terminate the program.
Please enter the corresponding number:
```

you are asked first the kind of input of the name of the dataset comprising the key.

At second with:

```
Please enter the name of the dataset containing the initial key
 (empty input terminates the program):
```

or with:

```
Please enter the name of the dataset containing the initial key
 the first time (empty input terminates the program):
```

and with:

```
Please enter the name of the dataset containing the initial key
 the second time (control input):
```

the dataset name is asked.

Dia: If you have chosen the item "key dataset" in the selection area for key input, the "folder" symbol is shown on the left side of the key input field. In analogy to the description of 1.6. Output Dataset for Control Information above you can insert a dataset name (with path information) in the key input field. Here you get the cursor position in the status bar too. If the key dataset field isn't selected i.e. hasn't the focus, the field information is displayed as stars. If the key dataset field is selected, the contents of this field only is visible if the mouse pointer is located or moved in the central region of this field. If the mouse pointer slowly leaves the central region of this field, then the information is hidden. Regardless of that marked field parts are always visible.

Bat:/Dia: Use key datasets if you have a complex or randomised key. A key dataset used by the current program task is n o t processed in this task (so not destroyed as well), even if this dataset is explicitly or implicitly addressed as a dataset to be processed (see 1.10. Exception Dataset above).

1.13.6 Random Key Input

Dia: In connection with the handling of parameter datasets you have the possibility to generate a random key by mouse moves in conjunction with pushing keyboard buttons and use this key as an initial key to generate the set of parameters for ciphering data in a parameter dataset. Clicking on the "mouse" symbol , you get a window with comments how to make it, before, after confirmation, you can begin with the key generation. During the key generation process the key input field shows ">> random key << generation ..." and a counter is set up in the status bar which shows you how many key elements are already generated. If all key elements are generated or if you cancel the key generation process before, you are informed by a separate dialogue window.

If the key generation process is successful, the "tick in green circle" symbol is shown. If you want to repeat this generation process, you only have to click on this "tick in green circle" symbol.

2 Description of Overlay Information of CODING

Bat:/Dia: The following information refers to the so-called “overlay information” especially being very significant for the programs CODING2 and CODING3. The values of the shown messages are values of CODING1 and have to be modified for CODING2 and CODING3 analogously.

“Overlay” means the possibility to modify a key manually or by datasets in a specific way resulting in a so-called derived key. Changes can be restricted to definite positions in the key, to particular key values, and to certain changing values inside the overlay information.

With that using a serial byte string, it is possible to use datasets, the overlay datasets, to overwrite values at “chance” positions in the present key by “chance” values derived by the overlay datasets.

Dia: Overlay information are only used if the corresponding checkbox field “with overlay information” below the initial key input field is switched on (see 1.13.1. above) or a parameter dataset (see 1.8 above) is used in the state of “overlay once again”.

2.1 Overlay Information

Bat:/Dia: In general every single overlay information is constructed by two values:

- the position value and
- the value which the overlay value is derived from.

Using a serial byte string (“direct mode”), each information consists of character values formed by characters being behind one another inside any dataset or inside any input character string (value ’7’ below). Depending on the program (CODING1/2/3) the position and overlay values are derived each from 1 or 2 or 3 characters respectively.

Bat: In all other cases (not the direct mode) the information has to be like this:

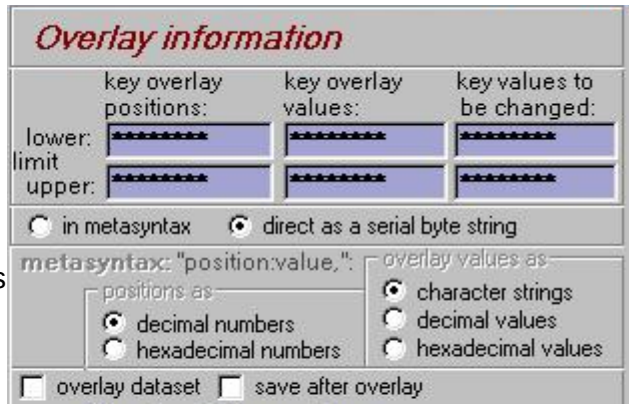
```
Shall the initial key be overlaid?
   0  no, no overlay of the initial key
   1  yes, in form of "<byteno-dec>:<charactr>,"
   2  yes, in form of "<byteno-dec>:<hexvalue>,"
   3  yes, in form of "<byteno-dec>:<decvalue>,"
   4  yes, in form of "<byteno-hex>:<charactr>,"
   5  yes, in form of "<byteno-hex>:<hexvalue>,"
   6  yes, in form of "<byteno-hex>:<decvalue>,"
   7  yes, in form of a serial byte string
   9  terminate the program.
Please enter the corresponding number:
```

Herein is

’byteno-dec/-hex’	a decimal or hexadecimal number of the referred key position in the interval of 1 to 256 /65,536 /16,777,216 or 1 to100_{hex} /10000_{hex} /1000000_{hex},
’charactr’	any 1 /2 /3 character(s),
’hexvalue’	a value in the interval of 0 to FF_{hex} /$FFFF_{hex}$ /$FFFFFF_{hex}$ consisting of hexadecimal half byte characters of 0 to 9 and A to F,
’decvalue’	a decimal value in the interval of 0 to 255 /65,535 /16,777,215.

Dia: Using the “simple” version of the program, you only have the possibility to use overlay dataset information in direct mode. Using the “enlarged” version, you can choose the form of the metasyntax or the direct mode, it’s also possible to input the overlay information in direct form.

Further the intervals of the key overlay positions, the key overlay values, and the key values to be changed can only be restricted by the user using the “enlarged” version, using the “simple” version these intervals cover “all values”.



For these interval fields it's valid out of safety reasons that if one of these fields isn't selected i.e. hasn't the focus, the field information is displayed as stars. If the field is selected, the contents of this field only is visible if the mouse pointer is located or moved in the central region of this field. If the mouse pointer slowly leaves the central region of this field, then the information is hidden. Regardless of that marked field parts are always visible.

Bat:/Dia: If a parameter dataset is used and only a parameter dataset handling takes place, you have the possibility to overlay the key of the parameter dataset once again in further program tasks. To do so, the parameter dataset is stored in state "overlay once again" and program terminates if the current overlay operation is done.

Dia: Using the "enlarged" version of the program, you can cause the program to do this by activating the checkbox field "save after overlay".
Bat: To find out what to do, the program ask you the following question:

```
Shall the initial key be overlaid once again in further program tasks?
   0  no, overlaying the key is finished after the current program task,
   1  yes, the initial key shall possibly be overlaid once again
      and the program terminates, if the current overlay operation is done.
   9  terminate the program.
Please enter the corresponding number:
```

Bat:/Dia: With that it's possible to carry out "any" complex overlay operations automatically.

2.2 Position Interval

Bat:

```
Please enter the lower and the upper limit of positions in the key,
 which may be changed by overlay information:
  lower limit of positions (1 to 256 (0=termination)):
```

and

```
  upper limit of positions (nnn to 256 (0=termination)):
```

Bat:/Dia: This information fixes the positions overlay information has to refer to to be potentially significant for changing the key. Only if the following overlay value interval information and the following key value interval information additionally come true, an overlay information really takes to overlay a key position by a value given by this overlay information.

Changes only take place at the positions of the position interval in principle.

2.3 Overlay Value Interval

Bat:

```
Please enter the lower and upper limit of values
 of overlay information the key may be changed by (in decimal form):
  lower limit of values (0 to 255 (-1=termination)):
```

and

```
  upper limit of values (nnn to 255 (-1=termination)):
```

Bat:/Dia: This information fixes the values of the overlay information potentially being relevant for changing the key. Only if the position interval information described before and the following key value interval information additionally come true, an overlay information really takes to overlay a key position by a value given by this overlay information.

2.4 Key Value Interval

Bat:

```
  Please enter the lower and upper limit of key values
   that may be changed by overlay information (in decimal form):
    lower limit of key values (0 to 255 (-1=termination)):
```

and

```
    upper limit of key values (nnn to 255 (-1=termination)):
```

Bat:/Dia: This information fixes the values of the key information potentially being relevant for changing. Only if the position interval information described before and the overlay value interval information described before additionally come true, an overlay information really takes to overlay a key position by a value given by this overlay information.

Doing this in general, it is possible that a position in the key can be overlaid repeatedly. After overlaying a position in fact, the overlay value is the new key value for the further overlay processing. The last overlay information applied to a position decides the value being the final key value at this position.
Carrying out overlay processing, an input value or a dataset value first is transformed by
CODING1:

overlay value := (input-/dataset-value + present key value ******* *113*
+ key value at previous position or at position 256 for position 1) modulo 256

CODING2:

overlay value := (input-/dataset-value + present key value ******* *30,781*
+ key value at previous position or at position 65,536 *for position 1) modulo* 65,536

CODING3:

overlay value := (input-/dataset-value + present key value ******* *100,000,000,019*
+key value at previous position or at position 16,777,216 *for position 1) modulo* 16,777,216

before the result-value is used as actual overlay value.
Position, overlay value, and key value interval information are of main interest to overlay datasets in direct mode. In direct mode **every** dataset can be set as an overlay dataset, program module datasets (*****.COM, *****.EXE) and other binary datasets too (**!**).

In general you have to observe that for all positions in the position interval **in connection with an overlay information varying in any kind and being (nearly) infinitely long** the following is valid:

As a result the key values of the key value interval are (almost) entirely transformed to overlay values of the overlay value interval.

If there are overlay values not belonging to the key value interval, the following **more restricted rule** is valid:

As a result all the key values are (almost) entirely transformed to overlay values of the overlay value interval which are not part of the key value interval (!).

The better an overlay information complies with these rules, all the more the named conclusions are valid. That must be taken into account to avoid trivial keys !

Given an initial key, an overlay information, and an overlay value interval, the **following is valid in general**:

Any key value interval covering this overlay value interval provides the same result key, if no key value being element of the key value interval but not being element of the overlay value interval is found in the initial key.

2.5 Input Mode

Bat:

```
  In which way you want to enter the overlay information?
     0  twice via keyboard  w i t h o u t  echoing the input on the screen
         (second time for controlling!)
     1  once via keyboard  w i t h  screen control
```

using overlay information of type 1 to 6:

```
     2  via a dataset containing the overlay information
```

using overlay information of type 7:

```
     2  via one dataset and all datasets addressed by wild cards respectively,
        which name is given in the following input
     3  via all datasets (possibly addressed by wild card information),
        which names are listed in the dataset given in the following input
     4  via all datasets of the directory given in the following input
     5  via all datasets of the directory given in the following input
         a n d  these of all subdirectories of lower levels
```

and:

```
     9  terminate the program.
  Please enter the corresponding number:
```

Dia: Using the “simple” version of the program, you only can use overlay dataset information, in the “enlarged” version you can enter such information in dialogue.

2.6 Keyboard Input

Bat: If you want to enter the overlay information by keyboard, you get the possibility depending on the type of information with:

’1’:

```
  Please enter the overlay information in form of "<byteno-dec>:<charactr>,"
```

’2’:

```
  Please enter the overlay information in form of "<byteno-dec>:<hexvalue>,"
```

’3’:

```
  Please enter the overlay information in form of "<byteno-dec>:<decvalue>,"
```

’4’:

```
  Please enter the overlay information in form of "<byteno-hex>:<charactr>,"
```

’5’:

```
  Please enter the overlay information in form of "<byteno-hex>:<hexvalue>,"
```

’6’:

```
  Please enter the overlay information in form of "<byteno-hex>:<decvalue>,"
```

’7’:

```
  Please enter the overlay information in form of a serial byte string
```

each with

```
   (end of overlay information: press only return-button in line):
```

or

```
   (end of overlay information: press only return-button in line) once:
```

and

```
   (end of overlay information: press only return-button in line) twice
                                                        (control input):
```

Dia: Using the “enlarged” version, you can enter the overlay information in direct form. If you enter an overlay information in direct mode, these information are handled as direct key information including repetition (see 1.13.4. above). If you enter overlay information in metasyntax, the following is valid. If this field isn’t selected i.e. hasn’t the focus, the field information is displayed as stars. If the field is selected, the contents of this field only is visible if the mouse pointer is located or moved in the central region of this field. If the mouse pointer slowly leaves the central region of this field, then the information is hidden. Regardless of that marked field parts are always visible.

Here the “magnifying glass“ symbol shows you that the syntax of the overlay information yet isn’t checked successfully with regard to the selected metasyntax.

2.7 Input via Overlay Datasets

Bat: If you want to use overlay datasets with:

```
In which way you want to enter the overlay dataset information?
   0  twice via keyboard  w i t h o u t  echoing the input on the screen
       (second time for controlling!)
   1  once via keyboard  w i t h  screen control
   9  terminate the program.
Please enter the corresponding number:
```

you are asked first the kind of input of the information of the overlay datasets. At second using overlay information of type 1 to 6:

```
Please enter the name of the dataset containing the overlay information
```

using overlay information of type 7, input mode ’2’:

```
Please enter the identification of the overlay dataset(s)
```

using overlay information of type 7, input mode ’3’:

```
Please enter the name of the dataset, in which the names
 of all overlay datasets are listed
```

using overlay information of type 7, input mode ’4’, ’5’:

```
Please enter the name of the directory,
 which contains the overlay datasets
```

all with:

```
   (empty input terminates the program):
```

or with:

```
   the first time (empty input terminates the program):
```

and:

```
   the second time (control input):
```

the dataset information is asked.

During the input of the overlay information via keyboard, line end characters are ignored i.e. may be used to split overlay information to more than one line, or, combined with an empty line, are interpreted as end of overlay information. But using overlay datasets in direct mode, line end characters are part of the overlay information and not ignored.

Bat:/Dia: For the most part it recommends to use an overlay dataset if you have many overlay information. Any overlay dataset used by the current program task is n o t processed in this task (so not destroyed as well), even if this dataset is explicitly or implicitly addressed as a dataset to be processed (therefore see 1.10. Exception Dataset above).
Dia: Using the “enlarged” version, it’s possible to use overlay datasets in one of the metasyntax forms, using the “simple” version, all overlay datasets are in direct mode by definition.

If you choose “overlay dataset” activating the corresponding checkbox filed, the “folder” symbol is shown on the left side of the overlay information input field.
In analogy to the description of 1.6. Output Dataset for Control Information above you can insert a dataset name (with path information) in the input field. Here you get the cursor position in the status bar too. If the input field isn’t selected i.e. hasn’t the focus, the field information is displayed as stars. If the input field is selected, the contents of this field only is visible if the mouse pointer is located or moved in the central region of this field. If the mouse pointer slowly leaves the central region of this field, then the information is hidden. Regardless of that marked field parts are always visible.

2.8 Overlay Datasets in Direct Mode

Bat:/Dia: Overlay datasets in direct mode have a special position using key overlaying.

While it seems to be useful to give the overlay positions and their overlay values in a separate (symbolic) way using “small” keys in connection with CODING1, this action is completely unsuitable using “big” keys in connection with the other programs to generate a “random selection” key of the available key room of the respective program.

That is why the method of key overlaying is expanded especially with regard to CODING2 and CODING3.

First you can choose more than one overlay dataset (see 1.9 and 2.5 above) and all these datasets are operated as one **overlay unit** (see 1.11 above) **Dia:** (only “enlarged” version). To do so, you can insert a dataset name covering overlay datasets enclosed in round brackets (’(’, ’)’), a directory name enclosed in square brackets (’[’, ’]’) or enclosed in angle brackets (’{’, ’}’) in the overlay information input field. By clicking on the “folder” symbol, you can start the directory choose dialogue and select a directory too if the first character in the input field is one of the brackets ’[’ or ’{’.

Second in this process the overlay datasets are used to overlay the initial key not only once but multiple, what further more is called **iteration** of this overlay unit.

Without modification of the overlay unit each also this approach doesn’t supply much more “chance” because at least the concerned positions in the initial key mentioned by the overlay unit would be the same in every stage.

In fact the original overlay datasets are used to modify the key only once. At the second and following stages the overlay unit itself is **continually enciphered** (!) before overlaying the initial key again. For enciphering the overlay unit, parameters are used which are initialised themselves by the initial key.

There are two big advantages in this way:

- Not depending on the “chance nature” and therefore the quality of the overlay datasets, extreme chance overlay information is generated by using the ciphering method presented here on the overlay unit.
- The quality of overlaying doesn’t depend on the length of the overlay unit.

As shown by examinations of the author, after approx. (x = length of overlay unit in bytes)

1,410 / x (CODING1) **1,441,452 / x** (CODING2) **854,889,360 / x** (CODING3) **iterations** the derived key is not any longer differ from a “chance” key.

That is why the program asks you for the iteration value if using overlay datasets in direct mode:

Bat:

```
Please enter the value of the iterations the overlay information
 shall be changed by the key information overlaying the key each
  ('no iteration'=0 or 1 to 99999999 (-1=termination)):
```

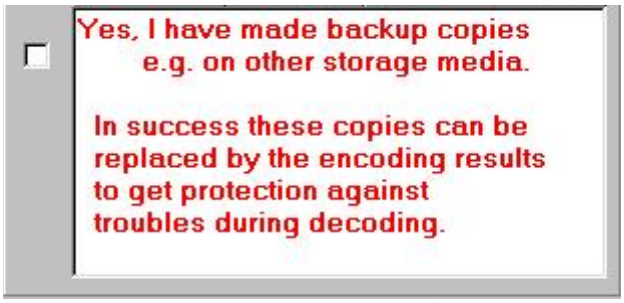
Dia: The contents of the iteration field acts as the overlay information input field in the case of entering an overlay dataset information (see above). The iteration field is only responsive or visible by handling overlay datasets in direct mode.

Bat:/Dia: Especially for overlay units being relative homogeneous or having less bytes than the numbers above, an iteration value bigger than zero shall be chosen. If there are no restrictions by using position, overlay value, and key value interval information, you shall choose the iteration value at least as described above to found round about 63.2% of all possible key values in the modified key (that's the value of a randomised distribution of the possible key values) assuming an initial key of "less" characters. Having a relative homogeneous overlay unit, you shall increase this iteration value by 1. If there are restrictions by using position, overlay value, and key value interval information, the calculated iteration value can be interpreted as a useful limit for a randomised variability.

You are free to choose a bigger iteration value (this has no affect to the given 63.2%). But it should be taken into consideration that big iteration values results in more time for key modification (here the overlay datasets are read n+1 times and internally ciphered n times).

3 Real Program Processing

Bat:/Dia: Because all ciphered datasets are directly processed (no copies are generated at all), a program interrupt in the ciphering phase can **irretrievably destroy (!)** one or more datasets, i.e. you **absolutely** have to make **backup copies** e.g. on other storage media (!). In success these copies can be replaced by the enciphering results to get protection against troubles during deciphering.



Dia: To rule out an unintentional mistake regarding this, **the program only starts if you have explicitly confirmed that you have made backup copies !**

If, for your opinion, you have entered and chosen all relevant information, you can start the real program task by using the “Start” button (see 1.5. Common Things). Before starting the program will again inform you about unconfirmed key and overlay information and will show you further warnings, so you have the possibility to put off the start of the program again. If important information is missing, the program stops the trial of starting in any case with a detailed error message. After correction of the errors, you can try to start the program again.

In this phase the program doesn’t check all consistencies. If overlay datasets and/or parameter datasets are handled, these datasets will be checked in detail at that moment.

3.1 Program Initialisation

Dia: If there are no difficulties find out by the dialogue system, the program begins its real task by displaying the identification side for short-term (see 1.3. above) before under program phase message ”Program is initializing ...“ the program shows the start message in the control output window.

If you have activated the checkbox field below the control information dataset field (see 1.6. above), then only “-- silence ! --“ is written into the control output window and a point in the next line shows that the actual program task has been ended successfully. If there is no error (!), no other messages are written to the control output window.

Bat:/Dia: If the program has to handle overlay information and shall report its activities (see 1.6. above), this is shown by the following message:

```
Begin of handling the overlay information: Th. 12/07/2006, 14:28:13.
```

by continuing its overlay information activities you can see:

```
Continuation of handling the overlay information: Th. 12/07/2006, 14:28:13.
```

and the end of this section is shown as the following:

```
End of handling the overlay information: Th. 12/07/2006, 14:28:13.
```

The initialisation section of the program reporting its activities is started with:

```
Begin of initialization: Th. 12/07/2006, 14:28:13.
```

The CODING programs report the activities in the initialisation and overlay information section in a different way. Because one table, one key, etc. covers 16,777,216 elements at 3 bytes in

CODING3 but only 256 bytes in CODING1, the activities corresponding to initialisation and overlay information are reported in a much more different way in CODING3 than in CODING1.

Therefore the programs CODING1 and CODING2 are only interruptible by time-out if they have closed an iteration step by the overlay handling, however the program CODING3 additionally proves the time-out limit and therefore can be interrupted during the initialisation of the ciphering parameters.

In case a parameter dataset handling takes place, the programs have information and enquiries for this handling in common.

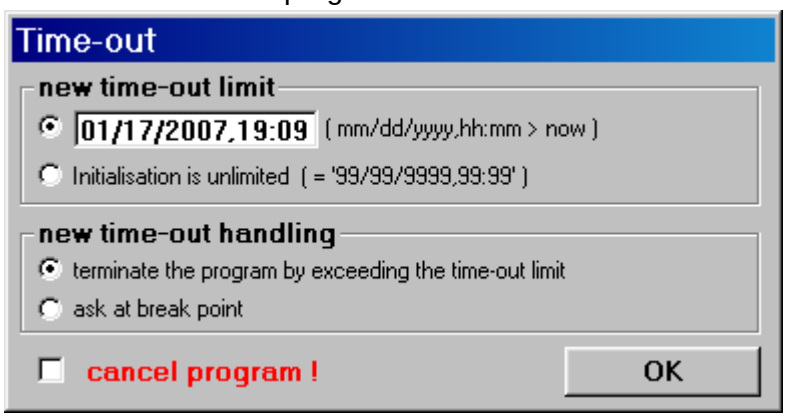
If, initialising a parameter dataset, the called time-out limit is exceeded and a new time-out limit shall be enquired, the program asks you as follows:

Bat:

```
Shall the program be continued with initialization ?
   0    - no, the program shall be terminated,
   1    - yes, the program shall continued with initialization.
Please enter the corresponding number:
```

After that the program asks for a new time-out limit (see 1.8.4. above) and the action reaching this limit (see 1.8.5. above) if you confirm the continuation of the program.

Dia: Besides the well known parameters of the time-out limit (see 1.8.4. above) and the action reaching this limit (see 1.8.5. above) the dialogue mask additionally comprises a checkbox field to cancel the program task early. Using the “OK” button, you either finish the actual program task or you continue the actual program task with the new parameters entered.



Bat:/Dia: If you use a parameter dataset and before the generation and final storing of the parameters for ciphering data, the quality of the initial key or the derived key respectively is shown in form of statistics (here: CODING1):

```
 contents of  +--------- character-frequencies in %: --------+ % binary   zero
 statistics:  min.:        max.:      mean:         variance:    ones:  groups:
-------------------------------------------------------------------------------
   Key:       0.00000000  1.56250000 (0.39062500)   0.38832041 49.36523437  93
```

For that a key is much more identical to a “random key” the better it corresponds to the following characteristics. That is

- the number of null groups that is the number of characters/elements not belonging to the key is close by the value 94 or 24,101 or 6,172,493 (CODING1/2/3) of all possible characters/elements,
- the percentage value of binary ones is close by 50%,
- the maximum character/element-frequency is close by the mean value,
- the variance is a small value.

Bat: If the initialisation process of a parameter dataset is closed and the program shall ask for the next action, you will find the following question if datasets shall be ciphered:

```
Shall the program be continued with coding datasets ?
   0    - no, the program shall be terminated,
   1    - yes, the program shall continued with coding datasets.
Please enter the corresponding number:
```

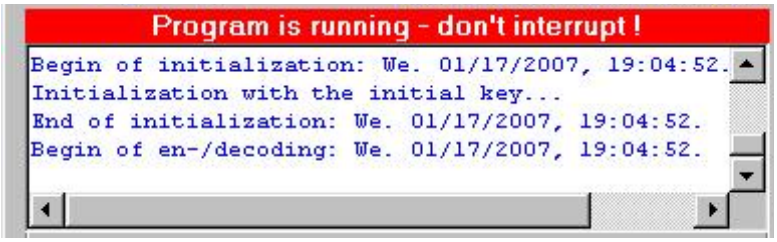
Bat:/Dia: If the program shall report its activities, the end of initialisation is shown as follows:

```
End of initialization: Th. 12/07/2006, 14:28:13.
```

3.2 Program Ciphering

Closing the initialisation – also a complete initialised parameter dataset needs an initialisation for activating the transformation table(s) (see 1.13. above) – and if there is no plain handling of a parameter dataset, the program starts with ciphering the data (i.e.):

```
Begin of en-/decoding: Th. 12/07/2006, 14:28:13.
```



With that the program is **n o t** interruptible any more. Because all ciphered datasets are directly processed (no copies are generated at all), a program interrupt in this phase can **irretrievably destroy (!)** one or more datasets, i.e. you **absolutely** have to make **backup copies** e.g. on other storage media (!). In success these copies can be replaced by the enciphering results to get protection against troubles during deciphering.

If the program shall report its activities (see 1.6. above), the names of all processed datasets and directories are written to the screen and to the control dataset during the processing, so every step can be controlled by you actual and later.

If you request a set of statistics of all datasets (see 1.12. above), these results are written at the end of the ciphering process (here: CODING1):

```
 contents of  +--------- character-frequencies in %: --------+ % binary   zero
 statistics:  min.:        max.:      mean:         variance:    ones:  groups:
-------------------------------------------------------------------------------
all datasets: 0.38749079  0.39417626 (0.39062500)   0.00130741 49.99381021   0
```

The program task ends by writing the number of characters, datasets, and directories which have been processed and the end message of the ciphering section (i.e.):

```
   212,430,600 bytes in  54 dataset(s) in  9 directory(ies) processed.
End of process: Th. 12/07/2006, 14:28:35.
```

With the terminate message

```
En-/decoding program  C O D I N G 1  1.1  terminated.
```

the program (**Bat:**) stops (**Dia:**) leaves the task (**Bat:/Dia:**).

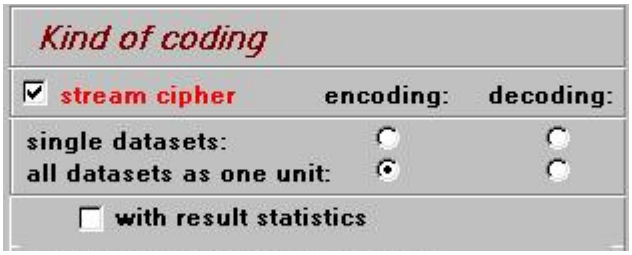
Dia: If a program task is finished, the operation buttons (see 1.5. above) are displayed and activated again. Now you can leave the application by using the “Exit” button, you can use the “Reset” button to clear input fields including the control output window or to reset values to the default. If not doing so, messages of the last program task(s) are preserved during the processing of further tasks, but identical datasets are overwritten or modified again (!).

If you move the mouse pointer in the control output window and this window covers more than 2 message lines, the **control output window is faded up** to the right lower corner of the main input mask so you can read the messages much better. As soon as the mouse pointer leaves the control output window, this window is scaled down to its initial size.

4 Extensions of the Standard Program

4.1 Stream Cipher

Bat:/Dia: Associated with a professional usage, the program in “enlarged” version described until now can be expanded fundamentally by the possibility of **stream ciphering**. This means the possibility to store an achieved parameter state which can be reactivated and continued to ciphere further datasets. In connection with that a set of parameters can be used potentially **to encipher as well as to decipher data**. **Dia:**



Bat: After handling the processing form (see 1.11) the following question is asked additionally:

```
Shall stream cipher processing be used?
   0  no, stream cipher processing shall  n o t  be used,
   1  yes, stream cipher processing shall be used.
Please enter the corresponding number:
```

or:

```
Shall stream cipher processing be used?
   0  no, stream cipher processing shall  n o t  be used,
   1  yes, stream cipher processing shall be used
      using a "normal" parameter dataset,
   2  yes, stream cipher processing shall be used
      using a component exchange parameter dataset,
   3  yes, stream cipher processing shall be used
      using both a "normal" parameter dataset
      and a component exchange parameter dataset.
Please enter the corresponding number:
```

Bat:/Dia: It’s true a “normal” parameter dataset used to store the state of stream ciphering isn’t a usual dataset but in contrary to the statement in 1.8.1 Parameter Dataset it is changed. This is because a recent parameter state is stored again in it at end. That is why you should make a copy of such a parameter dataset before processing to be able to fall back on a protected processing state. This copy can be deleted or can be overwritten by the new generated parameter dataset after successful processing.

Furthermore you have to take into account that you have to use **different and therefore own parameter datasets** both for enciphering and deciphering a data unit.

Stream ciphering should be used

- first if you wish **higher safety** and
- second if you want **data transmission** between two or more persons/offices involved using an uncertain and/or unsafe (public) connection (i.e. via **internet**).

Using stream ciphering the exchange of keys may be limited to a minimum. In contrary to the standard program which generates components at the beginning and in the following if a cycle of a pseudo random number process is finished (i.e. pseudo random numbers would be repeated), you can control component exchanges in the expanded program (tightened up ciphering) more exactly (see 5.2 Algorithm Parameters below).

In general the current internal operation key is used as an initial key to carry out a component exchange. Using the resulting components for ciphering, you have no possibility to generate the initial key in reverse processing (see “Examination of the Safety of the Algorithm in brief” too).

So if a set of components is compromised, all data ciphered before the last component exchange is not compromised but further data is.

To preserve the protection against compromising data ciphered after the next component exchange (naturally data ciphered with the compromised set of components is compromised too), it's possible – if program works in highest cipher state – to "enciphere" the initial key by the components of a second parameter dataset, the **component exchange parameter dataset**, before a component exchange is carried out using this ciphered initial key.

To brake this barrier by an unauthorized third party (without any direct access to the component exchange parameter dataset), first an attacker has to compromise "plenty" of sets of components of the "normal" parameter dataset to have only basically a chance to get the set of components of the second parameter dataset.

Ciphering **single datasets** (see 1.11 Processing Form above) using stream ciphering, a component exchange is made at the beginning of **every** dataset.

If only the state of parameters of the component exchange parameter dataset is saved at end and all datasets are ciphered as one **unit**, a component exchange is made at the beginning of the processing of every unit. This is an absolute necessity to get adequate recovery points processing more than one unit (the intermediate state of the standard parameters can not be saved because there is no "normal" parameter dataset or this dataset may not be saved !).

If both a "normal" parameter dataset and a component exchange parameter dataset is used to save the processing state of stream ciphering and all datasets are ciphered as one **unit**, a component exchange is made at the beginning of the processing of the first unit. This is shown by the **stream cipher usage value** of the parameter dataset which indicates whether a parameter dataset is modified after initialisation or still in initial condition. This introduced component exchange is made to guarantee the dependency already of the first used set of components on the component exchange parameter dataset (afterwards it's given automatically).

4.2 Component Exchange Parameter Dataset

Bat:/Dia: First a component exchange parameter dataset is a "normal" parameter dataset with the addition that this parameter dataset is only used to enciphere the operation key in this program task.

Dia:

Bat: Therefore the program (in highest cipher state) asks you first if you want to use a component exchange parameter dataset:

```
Shall a component exchange parameter dataset be used?
  '0'   - no, a component exchange parameter dataset shall not be used,
  'C'   - yes, a component exchange parameter dataset shall be used.
  '9'   - Terminate the program now.
Please enter the corresponding number or character:
```

If this question is answered affirmative, the program will ask you the dataset name:

```
Please enter the name of the component exchange parameter dataset
  (empty input terminates the program):
```

Bat:/Dia: As a "normal" parameter dataset a component exchange parameter dataset used in the current program task is n o t en-/deciphered as a normal dataset but it is changed by storing the final state of the ciphering parameters. That is why you should make a copy of a component exchange parameter dataset before processing to be able to fall back on a protected processing state. This copy can be deleted or can be overwritten by the new generated component exchange parameter dataset after successful processing.

As a “normal” parameter dataset a component exchange parameter dataset can be saved by a key used again to protect the final parameter state of this dataset. If doing so, you will be asked to enter the corresponding parameters or you can enter these information in the mask fields shown before (as described in 1.13 Key Information especially in the sections 1.13.1 to 1.13.4).

Dia: To be able to leave a “normal” parameter dataset unchanged (the stream cipher state is not saved in it) you can activate the corresponding checkbox filed below the name filed of the component exchange parameter dataset.

4.3 Stream Cipher Usage Value

Bat:/Dia: You can explicitly change – invert – the stream cipher usage value mentioned before too.

Dia:

Bat: In the section described in 1.13.1 Query to Parameter Dataset Security the program – if it works in highest cipher state – asks for:

```
Shall the parameter dataset be protected by a separate key?
  '0'   - no, the parameter dataset shall not be protected by a key,
  'S'   - yes, the parameter dataset shall be protected by a separate key,
  'R'   - yes, the parameter dataset shall be protected by a separate key
          and its stream cipher usage value has to be inverted.
  '9'   - Terminate the program now.
Please enter the corresponding number or character:
```

Bat:/Dia: You can change the stream cipher usage value only by changing the protection key of the parameter dataset.

4.4 Compromising Key

Bat:/Dia: As described in 1.13 Key Information, there is no possibility to decide whether a deciphering trial of a protected parameter dataset results in an original parameter dataset that means it’s successful or not. This is internally guaranteed in this way that all information being “treacherous” in any form are transformed in a representation in which **every** value occurs as a regular valid value. Only if this is done, the so modified information is stored in a parameter dataset.

Even if the algorithm realized in the program is traced regarding this, you have **under no circumstances** the possibility to distinguish between a “correct” and “wrong” protection key with regard to a parameter dataset !

This opens up the chance to use parameter datasets involved with stream ciphering in two (or more) different ways. If an involved person/office isn’t “master of the situation” any more, it has the possibility to inform all others about this “compromising” state in such a way that it doesn’t use the parameter dataset in the present way but changes to a set of parameters by using a **compromising key** previously defined for enciphering the parameter dataset. Being online this send-site transition can also be arranged by a keyboard shortcut for example.

Then the recipients detect that the normal set of parameters of the transmitter fails and have the possibility to change to the compromising set of parameters by enciphering with the normal protection key and deciphering with the compromising key and try it again. If this action is successful, the recipients for their part shall adapt their contents to the compromising state of the transmitter.

4.5 Disturbance of Data Transmission

For a transmitter a situation can appear in which after deciphering no useful result is yielded from a certain data block up. Because the development of the parameters is running in absolute parallel manner regarding the involved systems, there is only the possibility – besides compromising the transmitter – that

- there is a transmission error i.e. one or more data blocks are damaged,
- a wrong protection key has been used,
- a third party adds or deletes or destroys data blocks of the block stream to sabotage the communication.

Because the ciphering method progress is **not** coupled to block contents, it's **potentially possible to resynchronise** the ciphering parameters to the block stream in the first and third case. That's feasible by continuing the work if transmission errors appear or by ignoring data blocks or by using the ciphering parameters virtual on unavailable data blocks.

But the given program is only an offline version at the moment (no direct/online communication) so these conversions are reserved to a corresponding online version of the program.

4.6 Further Potential Application Possibilities

The existing algorithm ciphers the data blocks independently of each other. You can choose (nearly) any block length you like, even variable block length, using the given program (see 5.2 Algorithm Parameters).

The chaining of data blocks analogue the CBC mode (Cipher Block Chaining) of the DES process isn't very useful in the opinion of the author, because only one error in data stream not limited to data transmission forces the termination and therefore requires recovery. This opens up compromising possibilities which are unnecessary because the safety of the present algorithm seems to be sufficient enough even without block chaining. By the way error correcting codes (e.g. Huffman-Code) should be used to detect and if necessary correct transmission errors.

With the CFB mode (Cipher Feedback) of the DES process for example you are able to generate **pseudo random numbers**. As described in "Convergence Characteristic of the Central Operator XOR", it's not the aim of the given method to generate all possible states of a block. The given algorithm directs to block states with uniform distributions of bits, bytes and elements (byte-groups). How far these block contents can be changed to get "useable" pseudo random numbers by suitable modifications, it is reserved to future examinations.

The OFB mode (Output Feedback) of the DES process is independent of a block contents and is used for **authentication** among other things. Also the given algorithm can be used for authentications in (nearly) any length whatever. You only choose a fixed block length and a fixed block contents and so you have "a lot of" different authentication methods based on the given algorithm. As evaluated in "Examination of the Safety of the Algorithm in Brief", there is no possibility to draw conclusions to the ciphering parameters or even to the initial key.

In addition and on the analogy of that just said before, you can implement "any" **hash method** with (nearly) any result length using the given method. By authentication as well as by hash methods there is – in contrary to pseudo random numbers – no particular interest that all states of results are generated. In the process the previous contents of the (single) block is simply connected to the operation key (i.e. per 'xor') before the next block is treated. Treating the last block expanded to the hash length (equal block length) in any way gives the result which is the hash cipher of the data.

5 System and Algorithm Parameters of the Program

The described programs and components can be suited to the various requirements controlled by internal parameters.

5.1 System Parameters

This includes all parameters concerning computer, operation system, and operating mode adaptations.

5.1.1 Gruppe

=1,2,3 - count of bytes per byte-group (element) (fixed for CODING1/2/3)

5.1.2 Modulart

=0 - main program
=1 - subprogram for GUI
=2 - subprogram in DLL

5.1.3 DateiSystem

=0 - Windows (also >4 GB)
=1 - Unix without file logic >2GB
=2 - Unix with file logic >2GB
- - - - host file logic is not yet integrated

5.1.4 ZeichenSatz

=0 - ASCII
=1 - EBCDIC (i.e. Host-Computer)

If transferring enciphered files from ASCII to EBCDIC or vice versa, no byte conversion may be taken place ! In case of doubt the files can be expanded before transferring and can be unexpanded after transferring. After deciphering a transferred file on the destination system, an explicit transformation to the destination system character set has to be additionally done for files comprising "legible" characters.

5.1.5 BSigRtL

Double/Integer value significance of the mantissa/integer bytes increasing (the exponent following)
=0,2 - from left to right (e.g. PC CPU)
=1,3 - from right to left (e.g. mainframe CPU)

Output (and input) of the program is the byte sequence which is constructed or may be constructed
=0,1 - by =0
=2,3 - by =3 (“correct byte sequence”)

CODING2/3: Because almost all operations are carried out by elements and not by bytes, for =1/=2 all input data as keys and file data are first switched from “correct byte sequence” to “correct element valency” to be switched back at output time. Byte operations in connection with the remainder of an element can only be carried out in the state of “correct byte sequence” and therefore are excepted out of this as well as parameter datasets.

If (using identical CODING algorithm parameters, other system parameters variable) all computers work with =0/=1 o r with =2/=3 (other combinations are n o t possible), then files can be enciphered on a computer of one kind and can be deciphered on a computer of the other kind. Also parameter files can be generated, completed, and/or used on computers of one or the other kind using the same value combinations.

5.1.6 Verpflichtung

=0 - without user commitment
=1 - with user commitment and agreement

5.2 Algorithm Parameters

Using the following parameters, the algorithm can be adapted to a big range of requests which concerns technical as well as security points.

5.2.1 **varBlk** (CODING1/2)

=0 - fixed block length
=1 - variable block length with max. block length = 2 * min. block length

5.2.2 BlockElem

=n - fixed/minimal block length (of data) in elements (byte-groups);
- it m u s t be >1 and <2**(16*Gruppe) and
- it m u s t be >=8/4/4 (CODING1/2/3), if variable block length

=512,256,16384 - current value for CODING1/2/3

5.2.3 BlkKey

For fixed block length (CODING1/2: 'varBlk'=0):
=0 - operation key length = block length
=1 - block length >= operation key length
=2 - operation key length >= 2 * block length
=3 - block length < operation key length < 2 * block length

CODING1/2: For variable block length ('varBlk'=1):
=1 - min. block length >= operation key length
=2 - operation key length >= 2 * max. block length
=3 - min. block length < operation key length < 2 * max. block length

CODING3:
=4 - 2 + 'BlkKey2'=2 + variable block length
with max. block length = 2 * min. block length
(bigger max. block length aren't very useful at the moment and that is why it is not realized)

5.2.4 **BlkKey2** (CODING3)

=0 - no relevance (’BlkKey’<>2 and <>4)
=1 - operation key length in elements < 3 * block length in elements
=2 - operation key length in elements >= 3 * block length in elements

5.2.5 Safety

=0 - **default ciphering**:
- no repetition of linear substitution and no shifting modifying data blocks and the parameter dataset,
- static repetition of linear substitution changing operation keys on exchange of components,
- exchange of the components only after finishing a random number cycle,
- static linear modification of the operation key after an average number of 'MeanBlksD' blocks modifying data and of 'MeanBlksO' blocks modifying overlay data (operation key modification for e v e r y block if 'MeanBlks_'=0),
- no usage of a separate component exchange parameter dataset,
- no stream ciphering.

=1 - **tightened up ciphering**
- variable repetition of linear substitution and shifting modifying data blocks and the parameter dataset,

- variable repetition of linear substitution changing operation keys on exchange of components,
- exchange of the components after an average number of 'MeanBlksD' blocks modifying data and of 'MeanBlksO' blocks modifying overlay data (exchange of the components for e v e r y block if 'MeanBlks_'=0),
- static linear modification of the operation key for any other block if operation key is exhausted,
- no usage of a separate component exchange parameter dataset,
- possibility of stream ciphering in accordance with using a parameter file and saving the last cipher state in the parameter dataset.

=2 - **highest cipher state**:
- tightened up ciphering (see =1),
- exchange of the components possibly using a separate component exchange parameter dataset and saving the last component exchange state in the component exchange parameter dataset,
- exchange of the components of this component exchange parameter dataset after an average number of 'MeanBlksX' operation key blocks (exchange of these components for e v e r y operation key modification if 'MeanBlksX'=0),
- static linear modification of the operation key of the component exchange parameter dataset for any other operation key modification.

=3 - **highest cipher state**:
- the static linear modification of the operation key is interpreted as an "exchange of the components" too.

5.2.6 stLoop

=n - static repetition of linear substitution >0 CODING3: and < 1000
=2 - current value

5.2.7 stLoopC

=n - static repetition of linear substitution >0 modifying the operation key of the component exchange parameter dataset
=2 - current value

5.2.8 addLoop

=n - additional value of the repetition value of linear substitution ciphering data blocks; this value increases the repetition value 'Loop' derived from the block length in elements as follows:
Loop := ($\log_2$(block elements)+3)/4 + value out of [0,($\log_2$(block elements)+3)/4]
=0 - current value

5.2.9 opLoop

=n - basic and max. additional value of the repetition value of linear substitution changing operation keys on exchange of components,
- >= $\log_2$(key elements)/4 is useful, CODING3: < 500

=2,4,6 - current values for CODING1/2/3

5.2.10 PDLoop

=n - basic and max. additional value of the repetition value of linear substitution changing a parameter block
- > $\log_2$(2* key elements)/4 is useful

=3,5,7 - current values for CODING1/2/3

5.2.11 EndShift

=0 - no data block shifting at end
=1 - with data block shifting at end

5.2.12 ClearCore

=0 - no explicit erasing of main storage areas used by the program at end
=1 - main storage areas used by the program are explicitly erased at end

Each of the following values identifies the bits of the repetition value i.e. the bits of the 2. random number word / 1. random number element of the original and the current number which are to be compared to one another. 2 to the power of this bit count is the value which identifies the average number of blocks to be modified by
- an identical operation key ('Safety'=0) or
- identical components ('Safety'=1,2,3)

i.e. using 'Safety'=0 it m u s t be:
MeanBlks_ := 2**q -1 with q >= 0 a n d
with q < 8*Gruppe - ln(block elements / key elements) / ln(2) <= 16*Gruppe

5.2.13 MeanBlksX

=n - modifying operation key blocks
=63,8388607,4294967295 - current values for CODING1/2/3

5.2.14 MeanBlksD

=n - modifying data blocks
- <=2**6 –1 / <=2**23 –1 / <=2**32 –1
=63,8388607,4294967295 - current values for CODING1/2/3

5.2.15 MeanBlksO

=n - modifying overlay data blocks
- <=2**6 –1 / <=2**23 –1 / <=2**32 –1
=63,8388607,4294967295 - current values for CODING1/2/3

6 Contact Data

Email-address:
sysjm@t-online.de, subject "Coding"