CONNOTECH Experts-conseils inc.

DNS Resolver Software Change Planning

for Trust Anchor Key Management

Based on TAKREM

Thierry Moreau

Document Number C003595

2006/02/02

Document Revision History

C-Number	Date	Explanation
C003574	2006/01/09	Initial release
C003595	2006/02/02	Record sequence rules are now defined for the TAK-i and TAK-r files. Specific files are indicated for changes to the BIND 9.3.2 software. Various document updates.
C003595		Current version

Table of Contents

1. Introduction

TAKREM ([1]) is a recent development in the field of cryptographic key management support functions for digital signature assurance in the Internet network. The two fields of application are the X.509 self-signed root certificates ([2]) and the DNS security (DNSSEC) trust anchor keys ([3], [4]).

The Trust Anchor Key REnewal Method (TAKREM) turn into server management tools for the following functions:

1) a initial generation process for a number of public-private key pairs, and
2) a rollover message preparation process, starting from stored information from one of the above key pairs, including the installation of a private key in the protected computing environment used for generating digital signatures.

On the client side, the following software components are needed:

3) the initial configuration of a number of digest values for each initially generated public key, and
4) the rollover message validation process, including the possible installation of a public key as a trust anchor in the client-side configuration.

The present document focuses on client side of the DNSSEC application of the TAKREM procedure, i.e. DNS resolver software.

The present document is offered as a software implementation guide for the client side portion of the TAKREM procedures, for DNSSEC applications. That is, in the DNS application, the relying parties are end-systems using DNS resolver programs for enhancing the assurance about data retrieved from the DNS distributed database. This assurance ultimately rests on one or more *trust anchor keys* which may be rolled using the TAKREM procedure. The present document is intended for members of DNS resolver software development teams, in order to facilitate the deployment of sensible trust anchor key management facilities along with the DNSSEC protocol support.

A companion document [5] covers the server-side software tools for the DNS zone manager duties in relation with trust anchor key management.

The TAKREM procedure has implications in the trust anchor key life-cycle, which are also reflected in the resolver software.

- Signature Key Pair Generation

  The TAKREM key pair generation is done at once for a number of future trust anchor signature keys, and a digest array is distributed to the end-user systems, preferably as an intrinsic part of a DNS resolver software package. This is referred to as a TAK-i configuration file later in the present document.

- Dormant Phase

An individual trust anchor signature key stays in a dormant phase until the DNS zone manager starts the rollover operation for that key. The dormant phase has no impact on resolver software.

- Trust Anchor Key Rollover Operation

  The automation of trust anchor key rollover operation, while maintaining adequate security safeguards, is the main contribution of the TAKREM procedure and technology. It also represents the bulk of the complexity for DNS resolver software changes implied by the TAKREM support. Conceptually, the rollover operation provides some TAK-r information that is merged to the information from the TAK-i file, providing the new trust anchor key configuration. The recording of validated TAK-r information in a TAK-r file is a central function of the TAKREM-specific changes to the DNS resolver.

- Operational Phase

  For to support the operational phase of a trust anchor key, the TAKREM-specific software in the DNS resolver must provide up-to-date information about which signature keys may be trusted. A zone manager may whish to transition from a TAKREM management paradigm to a normal DNSSEC secure zone management paradigm, i.e. a parental DS secure delegation. This creates a requirement for parental DS flag information in the TAK-r file.

- Trust Anchor Key Retirement

  The TAKREM for DNSSEC application supports key retirement using a cryptoperiod start and end time in the SDDA RR, much like the RRSIG RR signature inception and expiration times.

The TAKREM application for DNSSEC is defined in two internet drafts ([3], [4]), respectively specifying the SDDA generic resource record format, and the TAKREM-specific use of this format. This separation in two draft documents isolates DNS protocol issues from TAKREM-specific process elements. The present document does not reflect the design independence between the two drafts, but the demonstration source code is perhaps closer to this dual specification organization.

The foremost DNS software package is the `bind` software ([6]). The DNS resolver logic for the DNSSEC support is provided in the current `bind` version 9.3.2. The present document refers to the `bind` software as an example of a DNS resolver software, and as a DNS authoritative nameserver for test environment setup (section 5.2).

2. Patent and Copyright Licensing

The TAKREM procedure is a patent pending technology ([7]). However, the following license announcement is to be read when GPL'ed software is at stake:

```
This software implements processes potentially covered by the
Canadian patent application number 2,511,366 and current and/or
future foreign counterparts, hereafter the TAKREM patent pending.

However, the patent holder CONNOTECH Experts-conseils inc.
granted a free, worldwide, and time-unlimited license to anyone
who so implement processes covered by the TAKREM patent pending
TO THE EXTENT THAT any such otherwise potentially infringing
activity is performed with free software licensed under the terms
of GNU General Public License Version 2, issued by the Free
Software Foundation, 51 Franklin St -- Fifth Floor, Boston, MA
02110, USA, irrespective of who is the copyright holder of such
free software.

For greater assurance, you may obtain a certified copy of an
"Unilateral and Conditional Patent License" in a dead of deposit
before Me Fabien L'Heureux, Notary for the Province of Quebec,
40, chemin Bates, room 126, Outremont, Quebec, Canada H2V 4T5,
quoting the file number 1418 (dated October 22nd, 2005) as a
reference.
```

The currently distributed software is licensed according to the GPL version 2.

Nothing in the present document should not be taken as legal advice. The issues of patent and copyright licensing should be addressed with the professional advice and relevant expertise.

3. Planning changes to a resolver software

3.1 Initial Configuration

The initial configuration of a DNS resolver should include the specification of two pathnames for the TAK-i and TAK-r files, which represent the complete configuration data for the TAKREM support in the DNS resolver. The subsection 4.3 describes the access requirements for these files, and the subsection 4.4 describes the file contents format.

The DNS resolver software portion that handles the configuration files should be augmented with a configuration element for to specify the two pathnames for the TAK-i and TAK-r files.

With the bind software, this may be implemented as an additional option statement:

```
[ dnssec-tak-files path_name path_name; ]
```

where file names for TAK-i and TAK-r files are indicated, respectively. At most a single occurrence of this option may be present in a resolver configuration. A corresponding documentation update is deserved, notably in the BIND 9 Administrator Reference Manual, chapter 6, BIND 9 Configuration Reference (the subsections on "options Statement Grammar", "options Statement Definition and Usage", and "trusted-keys Statement Grammar"). The source files that are updated for this option statement are bind-9.3.2/lib/isccfg/namedconf.c, bind-9.3.2/lib/dns/view.c, bind-9.3.2/lib/dns/include/dns/view.h, and bind-9.3.2/bin/named/server.c.

Note that the above specification is incomplete because a DNS resolver software might, in theory, support multiple SDDA-based DNSKEY authentication mechanisms. Whenever additional mechanisms are defined, a specifications revision should affix an indication of the authentication mechanism along the TAK-i and TAK-r pathnames, with TAKREM ([3]) being the default indication.

3.2 Addition of SDDA RR in Resolver Generic Query Logic

The SDDA RR is a DNS resource record which can be handled as a generic one in many portions of a DNS software transparently, including the general purpose query logic in a DNS resolver or name resolution utility. The SDDA RDATA portion has a specific structure and presentation format that can be supported in the general purpose query logic with minor software changes. It is perhaps wise to implement these changes as a first step towards TAKREM support in DNS resolver software.

In the case of the bind software, the addition of a new DNS RR type in the generic resolver processing requires the addition of two source files in a directory, with simple path name rules, respectively for C language declarations and C language function

definitions, respectively the file names bind-9.3.2\lib\dns\rdata\generic\sdda_65343.c and bind-9.3.2\lib\dns\rdata\generic\sdda_65343.h. The bind software configuration and build procedures will then reflect the new RR processing (specific RDATA structure and presentation format) throughout the resolver processing logic.

3.3 The TAKREM Processing Addendum

If a resolver software has been made DNSSEC-aware, the addition of TAKREM support should be considered with the demonstration source code as a starting point. This demonstration source code is like a TRKREM processing addendum, to be integrated in a DNS resolver software as one SDDA authentication mechanism for trust anchor key management. Various aspects of this development activity are covered in other subsections of the present document.

3.4 Software Environment Dependencies

3.4.1 C++ Source Code

The demonstration source code is C++ source code with little dependencies on external libraries or a coding style specific to a given DNS software package.

The bind source code is C source code. So a link with C++ modules may require recompilation of some of the bind executable with the sometimes stricter C++ language rules, or relaxing C++ language restriction with compiler control flags.

3.4.2 Multiple-Precision Arithmetic and Machine Language Dependencies

The discussion of software environment dependencies for server-side DNS tools in the reference [5] applies to the demonstration source code for the resolver side in the area of multiple-precision arithmetic and machine language dependencies, but only for the implementation of the MASH algorithm ([8]). A DNSSEC-aware resolver software already implements the RSA digital signature verification algorithm, typically through calls to a bignum and/or public key cryptography library, so it is perhaps reasonable to plan to implement the MASH algorithm support with the same library.

3.4.3 ASN.1 Idiosyncracies

The impact of ASN.1 idiosyncracies in the demonstration source code is limited to the recovery of proper input message format for MASH processing, which is a limited scope requirement that is implemented as manually coded software logic.

3.4.4 Logic Functions Spread among Resolver and TAKREM Addendum

The demonstration source code leaves signature validation and most of public key processing to the DNSSEC implementation within the resolver software, making more explicit the rollover-specific functionality, e.g. SDDA RR processing and trust anchor configuration handling. Note that this may have security implications if the programmatic interface is exposed to malicious tampering more than other interfaces between software components in the DNSSEC software implementation.

With this approach, one area of software logic duplication remains between the demonstration source code and the DNS resolver software, that is the parsing of text input according to DNS zone file syntax rules (see subsection 4.4 for a format description).

3.5 Modifications to DNSSEC Resolver Logic

The section 3 in reference [4] gives a description of the required modifications to a DNS resolver logic, essentially in the DNSSEC support. Such modifications are not provided in the distributed software. A detailed test plan should come first since the DNSSEC support logic is not trivial and dependent on resolver software implementation. In any event, the resolver logic modifications consist mainly of adding queries for SDDA RRset, validating the answers, and calling the demonstration source code API functions in appropriate circumstances.

4. The Demonstration Source Code and its API

4.1 API definition

The supplied demonstration source code offers a simple API for a DNS resolver software. This API uses the wire format for the relevant resource records, i.e. DNSKEY RR and SDDA RR. It is not specific to the TAKREM rollover scheme, although currently only TAKREM is known to comply to the SDDA model.

Demonstration source code API Summary:

Setup and cleanup functions:
- o sdda_api_config_validate
- o sdda_api_cleanup

Current trusted keys configuration queries:
- o sdda_api_known_name

- sdda_api_get_dnskey
- sdda_api_free_dnskey
- sdda_api_known_dnskey

DNSSEC protocol operations that may update the trusted keys configuration

- sdda_api_process_rr
- sdda_api_parental_ds

In terms of programming details, the API is based on a constant structure for each (currently only one) SDDA authentication mechanism. This structure holds an indication of the SDDA authentication mechanism, a set of function pointers, and a size indication for a context structure (of unknown contents as far as the application is concerned):

```
struct sdda_api_str {
   unsigned char auth_type;
   unsigned char *auth_type_extens;
   int auth_type_extens_len;
   int runtime_context_size;
   sdda_api_config_validate_t sdda_api_config_validate;
   sdda_api_cleanup_t         sdda_api_cleanup;
   sdda_api_known_name_t      sdda_api_known_name;
   sdda_api_get_dnskey_t      sdda_api_get_dnskey;
   sdda_api_free_dnskey_t     sdda_api_free_dnskey;
   sdda_api_known_dnskey_t    sdda_api_known_dnskey;
   sdda_api_process_rr_t      sdda_api_process_rr;
   sdda_api_parental_ds_t     sdda_api_parental_ds;
};
```

4.1.1 Setup and Cleanup Functions

```
/* returns non-zero on error */
typedef int
(*sdda_api_config_validate_t)(struct sdda_api_ctx *sdda_ctx
                              ,const char *tak_i_filename
                              ,const char *tak_r_filename
                              );

/* returns non-zero on error */
typedef int
(*sdda_api_cleanup_t)(struct sdda_api_ctx *sdda_ctx);
```

4.1.2 Current Trusted Keys Configuration Queries

```
/*
   return value:
    -7 : "TAK-i with valid parental DS"
    -6 : "TAK-i with or without current trust"
    -5 : "no trust information"
    >0 : error
```

```
   in *dnskey_count (if dnskey_count not NULL):
    count of currently valid DNSKEY public keys (when return value
    is -6 or -7)
*/
typedef int
(*sdda_api_known_name_t)(struct sdda_api_ctx *sdda_ctx
                        ,unsigned long now
                        ,const char *zone_name
                        ,int *dnskey_count
                        );

/*
   return value:
    -5 : no trust information for this zone name
    -4 : no DSNKEY for this index value
     0 : the returned DNSKEY RR is known as a rolled key, current
    >0 : error
   in *algorithm (if algorithm not NULL):
    the algorithm type field in the returned DNSKEY RR
   in *dnskey_rdata_pubk (if dnskey_rdata_pubk not NULL):
    a pointer to the wire format for the public key field in the
    RDATA portion of the returned DNSKEY RR
   in *dnskey_rdata_pubk_len (if dnskey_rdata_pubk_len not NULL):
    the length of the returned public key
   in *cryptoper_start (if cryptoper_start not NULL):
    start of cryptoperiod for the returned DNSKEY RR
   in *cryptoper_end (if cryptoper_end not NULL):
    end of cryptoperiod for the returned DNSKEY RR

   usage rules:

     If the sdda_api_known_name function returns -6 or -7 for
     given current time and dns name parameters, it is safe to
     call sdda_api_get_dnskey with these parameter values and a
     dnskey_index parameter value from zero to the returned
     dnskey_count (from the sdda_api_known_name function call),
     provided no intervening call to sdda_api_process_rr changed
     the trust anchor key configuration for this dns name.

     After a call to this function with a non-null
     dnskey_rdata_pubk parameter and a zero return value, a call
     should be made to the sdda_api_free_dnskey with the returned
     pointer in the dnskey_rdata_pubk parameter.

   usage recommendation:
     The implementation performance is enhanced when successive
     calls to the sdda_api_get_dnskey function are for the same
     current time and dns name parameters, and increasing
     dnskey_index parameter values.
*/
typedef int
(*sdda_api_get_dnskey_t)(struct sdda_api_ctx *sdda_ctx
                        ,unsigned long now
                        ,const char *zone_name
                        ,int dnskey_index
```

```
                          ,int *algorithm
                          ,unsigned char **dnskey_rdata_pubk
                          ,int *dnskey_rdata_pubk_len
                          ,unsigned long *cryptoper_start
                          ,unsigned long *cryptoper_end
                          );

typedef void
(*sdda_api_free_dnskey_t)(struct sdda_api_ctx *sdda_ctx
                          ,unsigned char *dnskey_rr
                          );

/*
   return value:
    -5 : no trust information for this zone name
    -4 : this key is unknown as a previously rolled key
    -3 : this key is known as a rolled key, empty cryptoperiod
    -2 : this key is known as a rolled key, in the future
    -1 : this key is known as a rolled key, in the past
     0 : this key is known as a rolled key, current
    >0 : error
   in *cryptoper_start (if cryptoper_start not NULL):
    start of cryptoperiod for this DNSKEY (when return value is
    0, -1, -2, or -3)
   in *cryptoper_end (if cryptoper_end not NULL):
    end of cryptoperiod for this DNSKEY (when return value is 0,
    -1, -2, or -3)
*/
typedef int
(*sdda_api_known_dnskey_t)(struct sdda_api_ctx *sdda_ctx
                          ,unsigned long now
                          ,unsigned char *dnskey_rr
                          ,int dnskey_rr_len
                          ,unsigned long *cryptoper_start
                          ,unsigned long *cryptoper_end
                          );
```

4.1.3 State-Updating Protocol Operations

These two operations are real-time, in the sense that their absolute time is irrelevant, while their order of occurrence is.

```
/*
   return value:
    -5 : SDDA not supported or no trust information for this zone
         name
    -4 : SDDA supported but does not target the supplied DNSKEY,
         or does not validate with the supplied DNSKEY
    -3 : SDDA validated, however the cryptoperiod is empty
    -2 : SDDA validated, however the cryptoperiod is in the future
    -1 : SDDA validated, however the cryptoperiod is in the past
     0 : SDDA validated, DNSKEY RR is now a current trust key
    >0 : error
```

```
   for return values in the range -3..0, the TAK-r file may be
   updated, which might involve a recording of the DNSKEY RR,
   recording of the SDDA RR, and the clearing of a parental DS
   indication

   assumptions:
     same owner names for the DNSKEY RR and SDDA RR
     a RRSIG RR signs (with this DNSKEY RR) the SDDA RRset
     enclosing the SDDA RR

   usage recommendation:
      within the SDDA RRset, every SDDA RR with a key tag matching
      the DNSKEY RR should be submitted to this function, a result
      in the range -3..0 being considered as a definitive outcome,
      with priority assigned to non-zero result within the range
      -3..0
*/
typedef int
(*sdda_api_process_rr_t)(struct sdda_api_ctx *sdda_ctx
                         ,unsigned long now
                         ,unsigned char *dnskey_rr
                         ,int dnskey_rr_len
                         ,unsigned char *sdda_rr
                         ,int sdda_rr_len
                         );

/*
   return value:
    -5 : no trust information for this zone name
     0 : parental ds cleared or set (perhaps redundant) according
         to parameter
    >0 : error
*/
typedef int
(*sdda_api_parental_ds_t)(struct sdda_api_ctx *sdda_ctx
                          ,const char *zone_name
                          ,int parental_ds_flag
                          );
```

4.2 the Demonstration Source Code Design

The intent of the demonstration source code is to provide a "qualified" reference source code for the TAKREM authentication mechanism, which operates behind a generic SDDA API in a DNSSEC-aware resolver software. The qualification refers to the lack of compliance to any specification for software coding rules, data types and other aspects of DNS software implementations.

The API functions for the TAKREM authentication mechanism are implemented in the source file takrem-sdda-auth.cpp, with declarations in the file takrem-resolv.h. The parsing of internal use files is implemented in the source files dns-rr-parse-taki.cpp, dns-rr-parse-takr.cpp, dns-rr-parse.cpp, dns-name-2text.cpp, and dns-rr-parse.h. A few other

include files are involved:

- the file sdda-api-defs.h declares the API functions, independent of the TAKREM authentication mechanism,
- the files dnssec-rr-defs.h and takrem-defs.h holds declarations based on relevant Internet RFCs and Internet draft documents.

The demonstration source code relies on some lower source code components:

- the large number arithmetic functions explained in above section 3.4.2,
- the MASH cryptographic algorithm implemented in the source files mash.cpp and mash.h, and
- a few calendar functions implemented in the files calendar.cpp and calendar.h.

These portion of the software implementation are not specific to the TAKREM authentication mechanism and may be replaced by functionally equivalent software.

4.3 Internal Use Files

The TAK-i file holds trust-anchor key initial information and and TAK-r holds trust anchor key rollover state information (e.g. a collection of SDDA RR received and accepted). These two files are used internally by the demonstration source code, and the TAK-i interoperability requirements (i.e. with the tools used by the various zone managers having a trust anchor key configured in the TAK-i file) are under control of DNS resolver developers and vendors. The proposed organization for the TAK-r file fits the TAKREM-specific file access requirements; its generalization to other SDDA-based authentication mechanisms is not necessarily proper.

The TAK-i file should be read-only and protected from malicious modifications. The TAK-r file is updated during rollover operation, like a log file. Preferably, the write operations should be restricted to the end of file, and the already written portion should be protected against malicious modifications. In many cases, the operating system file protection mechanisms might be the only mechanisms workable in practice, but a proprietary DNS resolver software implementation may use file contents obfuscation to attempt strengthening the integrity protection of trust anchor configuration.

4.4 Data Formats

The present section describes the file format for the TAK-i file, the TAK-r file, and test files intended to exercise the demonstration source code as a stand-alone software module prior to integration testing in a DNS resolver software. Thre present section is specific to the TAKREM authentication mechanism.

The file format uses a simple input syntax inspired from the syntax for the DNS text

representation of zone data (see section 5.1 of [9]): line-oriented input with continuation controlled by parentheses, comments introduced by a semicolon, DNS name occurs in first character position on a line, white space implies repetition of the previous name. The current demonstration source code does not support quoted strings, backslash escaped characters, and internationalized domain names. This can be seen either as a reasonable restriction assuming that zone names having trust anchors are likely to have simple names, or as a todo item in the software development planning.

Summary of input syntax

TAK-i file contents:

```
<name or blank>      TAKREM <integer>

<name or blank>      DIGEST <base 64 text>
```

The record sequence rules are 1) at least one TAKREM record, 2) each TAKREM record followed by at least one DIGEST record, and 3) any DIGESRT record having the same owner name as the preceding TAKREM record (e.g. blank name).

TAK-r file contents:

```
<name or blank> <ttl-class>  DNSKEY <integer> <integer> <integer or mnemonic>
                                    <base 64 text> <eol>

<name or blank> <ttl-class>  SDDA <integer> <integer or mnemonic> <integer> <integer>
                                    <integer or date-time> <integer or date-time>
                                    <base 64 text> <eol>

<name or blank>      PARENTALDS <integer>
```

The record sequence rules are 1) a DNSKEY record must be immediately followed by an SDDA record with the same owner name (e.g. blank name), and 2) any owner name must refer to a name present in the TAK-i file already parsed by the software module reading the TAK-r file.

In the above record syntax:

- <name or blank> is either <domain name> or blank space occurring at the beginning of the line,
- <ttl-class> is optional TTL value and optional DNS class mnemonic in any order,

and

- <eol> is end-of-line optionally provided by a semicolon introduced comment

Syntax for test files (informational only):

```
<name or blank> <ttl-class>  DNSKEY <integer> <integer> <integer or mnemonic>
                                    <base 64 text> <eol>

<name or blank> <ttl-class>  SDDA <integer> <integer or mnemonic> <integer> <integer>
                                    <integer or date-time> <integer or date-time>
                                    <base 64 text> <eol>

<name or blank> <ttl-class>  RRSIG  SDDA <integer or mnemonic> <integer> <integer>
                                    <integer or date-time> <integer or date-time> <integer>
                                    <domain name> <base 64 text> <eol>
```

5. Planning for test

5.1 The API test utility

5.1.1 Software Utility Design

The API test software is an interactive command-line utility for testing and experimentation with the demonstration source code.

The API test program uses a command parsing mechanism implemented in the files abcd_maintcmd.cpp and abcd_maintcmd.h. A primitive scripting mechanism is implemented in the file abcd_maintcmd_script.cpp. The command set is personalized in the files abcd_maintcmd_dispatch.h and abcd_maintcmd_dispatch.cpp, the latter containing the specific test commands. Two more commands are implemented in the file abcd_maintcmd_license.cpp. The files takrem-demo-test.cpp is the main program for the API test program.

The file dns-rr-parse-test.cpp contains test logic (function **`process_test_set_rr_s`**) similar to a DNS resolver software having in its current state a DNSKEY RRset, an SDDA RRset, and a few RRSIG RRs and looking for TAKREM rollover messages, if any, present in these RRs (actually, this logic applies to any SDDA authentication mechanism, it is not specific to TAKREM). In doing so, for RRSIG signature verification purposes, the API test program uses the SHA algorithm, implemented in th files sha.cpp, and sha.h. This rollover attempt operation is triggered by the command "DNS ROLL"

which expects a file name for a test file with a format specified in section 4.4. Such test files contain a DNSKEY RRset, an SDDA RRset and one or more RRSIG RRs targeting the SDDA RRset. Each of the test file is processed as if a DNS resolver would encounter the same set of DNS RRs (for the same and single zone name) in its operation, and the TAK-r file is updated accordingly. This is accomplished through exercising the SDDA API function **sdda_api_process_rr**.

Specifically, the DNS resolver processing rule is as follows: among the above collection of DNS RRs set, we process each individual SDDA RR within the SDDA RRset; if an SDDA RR validly authenticates a DNSKEY RR within the DNSKEY RRset and the SDDA RRset is validly signed by this DNSKEY public key in one of the RRSIG RR, the TAK-r information is recorded in the TAK-r file.

The other SDDA API functions are exercised by the API test software, with the exception of **sdda_api_known_dnskey**. Upon startup, the API test software reads a TAK-i file and a current TAK-r file specified as program startup arguments. This exercises the SDDA API function **sdda_api_config_validate**, while the function **sdda_api_cleanup** is exercised at normal program exit ("EXIT" command). The SDDA trust state query functions **sdda_api_known_name** and **sdda_api_get_dnskey** are exercised respectively with the commands "TRUST" and "TRUST KEYINDEX" (this last command also exercise the API function **sdda_api_free_dnskey**).

5.1.2 Software Utility Command Line Syntax

The program startup help text is shown below:

```
./takrem-demo-resolv -- A test and demonstration utility for TAKREM resolver
procedures.

Copyright (C) 2006 CONNOTECH Experts-conseils inc. This program
is free software; you may redistribute it under the terms of the
GNU General Public License. This program has absolutely no
warranty. Use the --warranty option for details.

More help is available with the interactive command "help".

Synopsis:

./takrem-demo-resolv [|-h|--help|-w|--warranty] <tak-i file> <tak-r file>

<tak-i file>
       The file name for the TAKREM initialization file.

<tak-r file>
       The file name for the TAKREM cumulative rollover file.
```

```
Options

      -h
      --help
           Displays this help message.

      -w
      --warranty
           Displays license and lack of warranty information.
```

The interactive command help is shown below:

```
Command:    "SCRIPT <string>"
Description:
            Read a "script file" (stored command sequence) into this program's
            command history buffer.
Command parameters:
       o    A quoted string of length in the range 1 to 250 inclusive.

Command:    "SCRIPT OFF"
Description:
            Turn off this program's special handling of command history buffer
            when a script file has been read.

Command:    "SCRIPT SAVE <string>"
Description:
            Save the current history buffer in a file e.g. to be edited and
            then used as a "script file" (stored command sequence).
Command parameters:
       o    A quoted string of length in the range 1 to 250 inclusive.

Command:    "GPL ABCD-W"
Description:
            Display the warranty disclaimer for this software.

Command:    "GPL ABCD-C"
Description:
            Display the copyright information (GNU GPL license) for this
            software.

Command:    "NOW"
Description:
            Tells to take the local computer clock as current time indication.

Command:    "NOW <string>"
Description:
            Take the date-time string (e.g. "20051231115959") as current time
            indication.
Command parameters:
       o    A quoted string of length in the range 14 to 14 inclusive.

Command:    "DNS DS <string> <integer>"
Description:
            Indicate the presence (1) or absence (0) of a parental DS for the
```

```
            indicated DNS zone.
Command parameters:
       o    A quoted string of length in the range 1 to 255 inclusive.
       o    An signed integer in the range 0 to 1 inclusive.

Command:    "DNS ROLL <string>"
Description:
            Trigger a TAKREM rollover attempt from the collection of DNS RRs
            in the indicated file name.
Command parameters:
       o    A quoted string of length in the range 1 to 255 inclusive.

Command:    "TRUST <string>"
Description:
            Query the current trust configuration about this DNS zone.
Command parameters:
       o    A quoted string of length in the range 1 to 255 inclusive.

Command:    "TRUST KEYINDEX <string> <integer>"
Description:
            Retrieve a current DNSKEY RR within this DNS zone, by index value.
Command parameters:
       o    A quoted string of length in the range 1 to 255 inclusive.
       o    An signed integer in the range 0 to 999999 inclusive.

Command:    "HELP"
Description:
            List of maintenance commands, with parameters and short
            description.

Command:    "EXIT"
Description:
            Exit the program.
```

5.2 Authoritative Zone Configurations for Testing

[[... work in progress ...]]

In order to test a DNS resolver operation under diversified conditions, it is necessary to set up authoritative zone nameservers, on a local or private network, or even on a single machine with various loopback IP addresses. The bind software ([6]) with the server management tools described in reference [5] appears as a convenient test environment solution (even if the resolver side is unrelated to bind). The server-side demands for SDDA and TAKREM software testing are much lower than on the resolver side because 1) once a zone is signed and authenticated with SDDA records, the nameservice function is only modified by the addition of queries and replies for SDDA RRsets, and 2) the system count in deployment on the server side is orders of magnitude less than the resolver system count.

In designing test plans to cover the possible operational conditions, the gregarious property of DNSSEC digital signatures becomes a simplifying assumption (at least to a security specialist used to independent chains of crypto-based assurance): an RRSIG RR signature authenticates a whole set of RRs of the same type for the same owner name, e.g. a KSK authenticating a particular DNSKEY signature key at one point in time also authenticates every other DNSKEY occurring in the enclosing DNSKEY RRset (this applies also at the parental side of a secure delegation for DS records).

The use of caching nameservers for testing purposes might be envisioned, e.g. for the expected operational condition where a DNSKEY RRset is cached, but the SDDA RRset authenticating one of the DNSKEY RR would not be cached. However, such a test is believed to validate the chaching nameserver implementation, and not the DNS resolver.

[[... work in progress ...]]

5.3 Test plan contents

[[... work in progress ...]]

- Testing the demonstration source code robustness in input files processing (TAK-i and TAK-r).

- Testing the demonstration source code functionality.

- Testing the DNS resolver (augmented with the demonstration source code) functionality:
  - a currently valid trust anchor is used
  - an expired trust anchor is not used, SDDA RRset queried before giving up
  - a successful SDDA authentication is recorded in TAK-r file
  - a failed SDDA authentication attempt is followed by a parental DS query
  - such a parental DS query, if successful, is recorded in TAK-r file
  - a zone with TAK-i information and a parental DS flag is queried first for the parental DS
  - when unsuccessful, such a parental DS query is followed by an SDDA query
  - above queries are not performed if “local policy” trust anchor (if implemented in DNS resolver) exists for a zone
  - interaction of the above when the chain to the root zone has more than one zone listed in the TAK-i file
  - tests when the zone is the root zone (i.e. no parental DS)

[[... work in progress ...]]

6. References

[1] Thierry Moreau, "A Note About Trust Anchor Key Distribution", CONNOTECH Experts-conseils inc., Document Number C003444, 2005/07/05, http://www.connotech.com/takrem.pdf

[2] Thierry Moreau, "Trust Anchor Key Renewal Method Applied to X.509 Self-signed Certificates (TAKREM-X.509)", Internet Draft (work-in-progress), draft-moreau-pkix-takrem-01.txt, September, 2005

[3] Thierry Moreau, "The Trust Anchor Key Renewal Method Applied to DNS Security (TAKREM-DNSSEC)", Internet Draft (work-in-progress), draft-moreau-dnsext-takrem-dns-00.txt, October, 2005

[4] Thierry Moreau, "The SEP DNSKEY Direct Authenticator DNS Resource Record (SDDA-RR)", Internet Draft (work-in-progress), draft-moreau-dnsext-sdda-rr-00.txt, October, 2005

[5] Thierry Moreau, "Server Management Tools for Trust Anchor Key Management Based on TAKREM", CONNOTECH Experts-conseils inc., Document Number C003595, 2006/02/02

[6] http://www.isc.org/index.pl?/sw/bind/, ISC (Internet Systems Consortium) BIND (Berkeley Internet Name Domain)

[7] Thierry Moreau, "Trust Anchor Key Cryptogram and Cryptoperiod Management Method", Canadian patent application number 2,511,366, filed on June 30, 2005, http://patents1.ic.gc.ca/details?patent_number=2511366&language=EN

[8] International standard document ISO/IEC 10118-4:1998, "Information technology - Security techniques - Hash-functions - Part 4: Hash-functions using modular arithmetic"

[9] P. Mockapetris, "Domain Names - Implementation and Specification", RFC1035, November 1987