CONNOTECH Experts-conseils inc.

Server Management Tools

for Trust Anchor Key Management

Based on TAKREM

Thierry Moreau

Document Number C004290

2010/01/31

Document Revision History

C-Number	Date	Explanation
C003515	2005/10/28	Initial release
C003529	2005/11/24	Added a spool size parameter and the capability to output multiple sets of components in the trust-anchor-foundry utility, minor change in title line header (not reflected in reference [24])
C003576	2006/01/06	The two DNS-related Internet Drafts ([4] [3]) were updated to revision -01. The three DNS utilities are now documented, with a diagram in the introduction. The document is released in spite of a lack of corresponding software distribution and an unfinished generic bar code read utility (lab-comm).
C003596	2006/02/02	Added the documentation for the takrem-barcode utility. Made a reference to GPL version 2 where the version was not specified. Various document updates.
C004290	2010/01/31	Somehow up-to-date with software revision for trust anchor foundry utility, including the backup functionality, section 4
C004290		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]). Major technical issues are mostly common to both X.509 and DNSSEC. The implementation details differ, however, and the present document is primarily focused on the DNSSEC application.

The present document is offered as a software implementation guide for the server side portion of the TAKREM procedures. As such, we avoid the repetition of detailed specifications provided elsewhere. In its first and current form, the present document is not a tutorial, i.e. neither for the TAKREM security scheme nor for the key management operations implied by it. The present document is a work-in-progress.

A trust anchor key is a public key for a given cryptosystem. The TAKREM procedures include the generation of key pairs for this cryptosystem, so it rests on established key generation strategies. In its current level of advancement, the supported cryptosystems are RSA and the PEKE cryptosystem. The PEKE public key format and the object identifier for this public key cryptography algorithm are defined in the annex A in [6].

Two main elements of the Trust Anchor Key REnewal Method (TAKREM) turn into server management software utilities:

1) a initial generation process for a number of public-private key pairs, implemented in the trust-anchor-foundry utility (including the dead storage output function as a bar code printout), 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 as two software utilities, respectively
   - o for dead storage input operation for retrieving a new trust anchor public key, implemented in the takrem-roll-pub utility, and
   - o for dead storage input operation for retrieving the corresponding private key, implemented in the takrem-roll-pri utility.

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

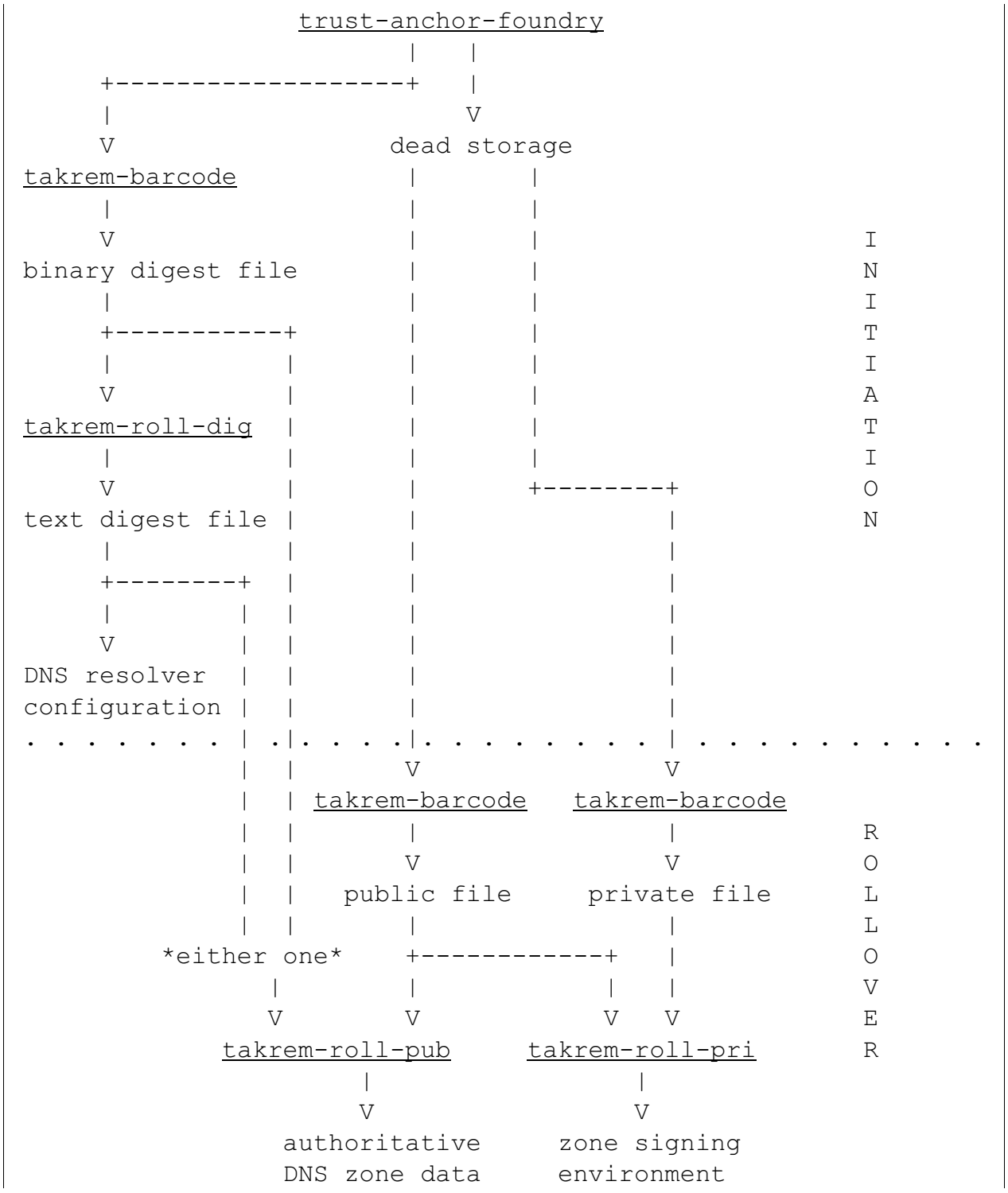
3) the initial configuration of a number of digest values for each initially generated public key, which is a secure configuration step not specified by the TAKREM design beyond a data format (a trust anchor key rollover piggybacks on an existing public key cryptography application scheme with a prior requirement for trust anchor key configuration storage), 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 software design issues for the implementation of TAKREM on the server side of a network use of public key cryptography (items 1 and 2 above). This is represented below in a diagrammatic form, where two additional utilities are introduced, respectively takrem-barcode for bar code reading and takrem-roll-dig for data format conversion applied to the TAKREM digest array.

A companion document [5] covers the client-side software modifications for the DNS application of the TAKREM procedure.

Since trust anchor key management is related to DNS zone management with added security procedures in the case of the DNS root domain and gTLDs and ccTLDs, it is subject to issues that mainly policy issues. The present document sometimes refers to these issues, to the extent they interact with technical characteristics of the software tools.


trust-anchor-foundry
dead storage
takrem-barcode
binary digest file
takrem-roll-dig
text digest file
DNS resolver
configuration
INITIATION
takrem-barcode
takrem-barcode
public file
private file
ROLLOVER
either one
takrem-roll-pub
takrem-roll-pri
authoritative
DNS zone data
zone signing
environment

2. Design Requirements

2.1 TAKREM Software Components Organization

2.1.1 Overview

The initial generation process (trust-anchor-foundry utility) is currently implemented with the MASH-2 variant only and without support for advance issuance of security certificates. It is thus irrelevant whether the X.509 or the DNSSEC application is envisioned at this phase. An oversimplified view is that it generates a series of public key pairs and prints them as binary data bar code pages.

The other three utilities, takrem-roll-dig, takrem-roll-pub, and takrem-roll-pri, use the data from these bar code pages and recover respectively a TAKREM digest array in an application-oriented format, a private key, and a public key in the same type of format. It is when such application-oriented representation is obtained that the entity name is affixed to a public key, but it is an operational error to affix any name not initially advertised to the relying parties (this error is more likely if the binary representation of the TAKREM digest array is input to the takrem-roll-pub utility). Similarly, the software tools do little to prevent the use of a private signature key for a wrong entity name.

The takrem-roll-pri utility function is expected to be run in a highly protected computing environment. Ideally, it would be part of the high assurance firmware in a cryptographic device. In a more readily available hardware configuration, the highly protected computing environment can be a computer system without any live connection to a computer network.

2.1.2 DNSSEC Application

For the DNSSEC application, compatibility with the mainstream bind software is achieved by providing two utilities as a replacement for the key generation utility dnssec-keygen. This operation is done before a trust anchor key rollover. The two utilities are creating respectively the private key in the .private file (created by the takrem-roll-pri utility) and the public key in the .key file plus a TAKREM-specific SDDA RR in a .sdda file (created by the takrem-roll-pub utility).

The bind software version 9.3.2 is augmented with the SDDA RR support by the addition of two source files, sdda_65343.c and sdda_65343.h in the bind distribution directory lib/dns/rdata/generic. This is the normal mechanism to add a new resource record type in the bind software. The files sdda_65343.c and sdda_65343.h are compliant with the Internet

draft specifying the SDDA resource record ([4]).

A final change to the bind software for zone manager support of TAKREM for DNSSEC occurs in the zone signing utility (dnssec-signzone), where a DNSSEC signature key with the SEP bit set is used circumstantially (in contrast with signature keys with the SEP bit cleared). That is, unless the **`-k`** command-line option is present, the dnssec-signzone uses a signature key with the SEP bit set only for the DNSKEY RRset in the zone data to be signed. With the addition of the SDDA RRset, a signature key with the SEP bit set should also sign the SDDA RRset. This is implemented as a small logic change in the function **`signset`** in the file bind-9.3.2/bin/dnssec/dnssec-signzone.c. Note that the SEP cleared signature of an SDDA RRset may be manually removed from the signed zone file after dnssec-signzone utility is run.

2.1.3 Dead Storage of TAKREM Secret Key Material

2.1.3.1 Key Material in Multiple Components

The trust-anchor-foundry utility supports dual components, triple components, and a simple two out of three scheme, as specified in a utility command line option. The bar code read utility (takrem-barcode) take any number of components when recovering a binary file from bar code pages.

2.1.3.2 Key Material Backup Support

The practice of keeping backups of private signature keys is subject to debate when a high assurance application is at stake. In the case of DNSSEC private signature keys for DNS zones at the root or TLDs, such backups are mandated by the data escrow or mirror site obligations of zone managers to the ICANN overseeing body. The trust-anchor-foundry utility has been revised to allow the creation of independent sets of key components for the key material, so that one set can be assigned to data escrow custodians.

2.1.3.3 Dead Storage Format

The public part of the dead storage is exactly the input data to the MASH processing specified in the TAKREM procedures.

The private part of the dead storage is one of the prime factors, both for the RSA and the PEKE algorithms. This means that the takrem-roll-pri utility might need to do some arithmetic on the public key data in order to support private key formats with pre-computed fields (e.g. the private exponent in the RSA private key data representation).

2.1.4 Text Representation of TAKREM Initial Configuration

The text representation for the TAKREM initial configuration uses a simple output syntax inspired from the syntax for the DNS text representation of zone data (see section 5.1 of [7]): line-oriented output, DNS name occurs in first character position on a line, white space implies repetition of the previous name. There are two types of lines, respectively:

- o to introduce a TAKREM digest array with a zone name and a reference number, and
- o to convey a digest value in this digest array:

<name or blank> TAKREM <integer>
<name or blank> DIGEST <base 64 text>

This text representation is generated by the takrem-roll-dig utility (see section 6). Here is an example:

```
example.com. TAKREM 1000
             DIGEST zSJ5vpUVOsskM6MnHdVOweVTM5WI/4vhe3zHM0Gw8XY=
             DIGEST 26tHs/2UdM4nyH6uwYrC1m83XxTh91eDrSBSylYZ6ME=
             DIGEST Gqt/suJ6X2zCAxa42ZDAj+zKndUzW7WCwgQ1soj4bS0=
             DIGEST fzrf9aj2iAeFzem7f0GlVs8idYLWZC+x2VaegFLXWoY=
             DIGEST hQE+FsgjTDQ3KTFzlsi1s54Z26l9D/u6baSCNjTPhYA=
```

2.2 Unique Cryptographic Strength Requirements

One of the main rationale for trust anchor key rollover is the perceived threat of a powerful adversary breaking the signature algorithm over a period of time shorter than the cryptoperiod. Thus any cryptographic algorithm used within a rollover scheme should have large key sizes and well assured strength, most likely at the expense of bad performance characteristics. However, almost every well studied and standardized cryptographic algorithm attempts to provide a quantified level of security with the minimal performance impact.

Intrinsic to the TAKREM design is the selection of the MASH cryptographic algorithm for public key digest computation, which is an application of the above principles. For secret random number generation, this is an implementation issue discussed in the following sub-section.

2.2.1 Random Number Generation

The storage of secret data using the split-component technique is equivalent to one-time-pad or stream cipher encryption for relatively small messages. One-time-pad equivalence would occur if split-component storage uses a true random source (typically based on sampling of an unpredictable physical process). Stream cipher equivalence would occur if a standard stream cipher was seeded by a true random source, in which case one of the component would be determined by the random seed (we insert a specific processing step to avoid the direct observation of a contiguous keystream sequence, see characteristic D below).

Prudent engineering is being applied to provide split-components in a stream cipher arrangement in a multi-stage design with the following characteristics:

A provable cryptographic properties,
B "artificially" long period (i.e. using a period lengthening method which do not extend the provableness),
C the Frgobit combining technique,
D a modified stream cipher mode for the split-component technique.

The basic idea implemented by the characteristics A-C above is that the stream cipher state (and, with some heuristic arguments, the stream cipher period) is "much larger" than the typical secret data size. We defer the rigorous presentation of the principles used in the combination of characteristics A-C in the software implementation.

Assuming the stream cipher state can (hypothetically) be recovered from the knowledge of a key stream segment of the same size as the stream cipher state, we whish to counter an attack from the knowledge of a single split-component value with the characteristic D above. The latter is procedurally independent from the stream cipher design (A-C). We use it to "equalize" each component in the split-component technique, i.e. there is neither key stream component nor any single ciphertext component. Thus a malicious party having a single component can not readily have a segment of the keystream output to start cryptanalysis.

Implementation-wise:

- characteristic A turns into a BBS (Blum-Blum-Shub) Pseudo-random number generator ([8], [9]), see source files mnumber_bbs.cpp and mnumber_bbs.h;

- characteristic B turns into a MTGFSR ([10]), see source files mtgfsr.cpp, and mtgfsr.h;

- characteristic C uses 20 independent PRNGs made of the above A-B combination, and uses the Frogbit algorithm ([11]) with a constant zero input, as an heuristic mixing method, see source files frogbit.cpp, frogbit.h, and super_prng.cpp;

- characteristic D uses a permutation of individual bits of the same rank among the two (or three) split components, part of the function **`print_key_material()`** in source file takrem_gener.cpp.

About the MTGFSR implementation, the current software distribution omits some self-validation algorithm (i.e. the Berlekamp-Massey algorithm) and the MTGFSR parameter search program.

2.3 Software Environment Dependencies

The currently distributed software is originating from CONNOTECH Experts-conseils inc., but there is no known good reason not to use other source code. Although it is not a purpose of the present document to assist selection of competing software tools, we are aware of two free software development projects from which a developer should find satisfactory solutions to the software environment dependencies:

- the GnuTLS project, The GNU Transport Layer Security Library ([12]), which uses a related ASN.1 library ([13]) and the Libgcrypt cryptographic library ([14]),

- the OpenSSL project ([15]) which is used as a cryptographic library for implementing DNSSEC security functions for the bind DNS software ([16]).

2.3.1 Multiple-Precision Arithmetic and Machine Language Dependencies

A "big number library" is a prerequisite for any software doing public key cryptography. A machine language dependency is inescapable if any decent performance is expected from the big number arithmetic operations.

The currently distributed software uses a large number library developed internally. See the high level source files mnumber.cpp, mnumber.h, montgom.cpp, mult_div.cpp, and mnum_dev.cpp.The need for assembler code optimizations is reflected in the source files diagonal.asm and trial96.asm, with support limited to the Intel x86 instruction set.

2.3.2 Secret Random Data Source Dependency

The pseudo-random number generation (PRNG) scheme described in section 2.2.1 above requires some secret true random data source for initial PRNG state initialization. This is no different from any cryptographic key generation application, except for the larger size of PRNG state. This is usually an hardware-dependent dependency.

The currently distributed software uses the Linux hardware environment sampling mechanism built-in the device handler `/dev/random`. See the preprocessor symbol `ENTROPY_FNAME` in the source file `super_prng.cpp`.

2.3.3 ASN.1 Idiosyncracies Dependency

The TAKREM use of ASN.1 format encoding and decoding is very limited. Since the DER encoding is used and few syntactic choices are included in the actual ASN.1 syntax, it is reasonable to plan a software implementation devoid of any full-blown ASN.1 format support library.

The currently distributed software uses some ASN.1 software developed internally. See the files source files `asn1defs.h`, `objid.h`, `asn1impl.cpp`, `asn1mnum.cpp`, and `objid.cpp`. The ASN.1 input parsing is assisted by low-level BER parsing (DER is basically a subset of BER) in the file `asn1pars.cpp`, while syntactic compliance with an ASN.1 syntax rule specification is implemented by manually coded program logic.

2.3.4 Bar Code Standardization Dependency

2.3.4.1 Formal Bar Code Specification

The specification in the present subsection defines a bar code format for storage of binary data in the "dead storage" format required by the TAKREM procedure. Besides the mandatory provisions, good practice of bar code printing techniques should be followed.

The bar code standard known as "Code 128 Code Set C" shall be used. The overall length of a bar code label shall not exceed 3.2 inches (81 mm) (implemented in the source file `code128.cpp`).

The bar code encoding and printout function shall take a byte string as input and proceed according to the three phase procedure described below to generate one or more bar code pages (implemented in the source file `code128packing.cpp`).

Phase I - Conversion into a Sequence of Integers

Logically, the byte string to be encoded is arranged as a bit string starting with the most significant bit in the first octet and ending with the least significant bit in the last octet. This byte string is cut in complete 33 bit blocks, each converted in an integer, in a most significant bit first notation. Once no more complete 33 bit block can be extracted from the string, it is extended with a zero bit and the minimum number of one bits (including zero) so that bit count is among 6, 13, 19, 26, or 33. This resulting bit string is converted in an integer and its original bit size is retained.

Phase II - Conversion into Code 128 Code Set C Symbols

The resulting sequence of integers produced in the preceding step is converted in a sequence of symbols in the base 99 encoding, i.e. digits in the range 0 to 98 inclusive. Each integer occupies from 1 to 5 digit places, respectively from original integer bit sizes of 6, 13, 19, 26, and 33. The sequence of Code 128 code set C symbols is obtained by adding one to each symbol in the base 99 encoding sequence (this avoids the space character ambiguity in the Code 128 checksum computation) .

Phase III - Packing into Code 128 Bar Code Labels

The sequence of Code 128 code set C symbols is packetized in a sequence of bar code labels, according to a defined disposition of labels on a page, and subject to the above label length limitation (e.g. this implies prefixing a "Start C" symbol, and affixing a calculated check digit and a "stop" symbol and allowing for quiet zones on each side of the label). Multiple labels appearing on a single page shall have a parallel orientation. The projection of labels along their length axis shall either not overlap at all or overlap exactly (i.e. every labels on a row will be aligned). The sequence of label reading operation shall be from left to right, top to bottom, as if reading text, when looking at labels horizontally. Some marking (e.g. a human readable title text) shall allow the determination of the top of the page. If multiple pages are required to encode the byte string to be encoded, they shall be numbered with a human readable indication.

2.3.4.2 Bar Code Application Details

The page layout consists of an A4 or North American letter format page in portrait orientation with up to 13 rows of 2 labels each. Label rows are positioned with space such that the folds in the paper do not cross a label when the page is folded in three sections. A title line is printed at the top, with page numbering information on the top right corner.

The bar codes are printed with a 7.5 mils "x" dimension (the width of the narrow bar or element), allowing up to 33 symbols in a single label while complying to the label length limitation. The label height is set to 1/2 inch (12.7 mm). A human readable indication of encoded code set C symbols is affixed under each label as two lines of decimal digits, where each symbol is to be read vertically.

These application characteristics and the formal specification in the preceding subsection allows the use of low-cost bar code reader equipment based the CCD technology. We experimented with the Opticon OPT-6125 bar code reader [17], to our satisfaction.

2.3.5 Black and White Graphics Output Format Dependency

The trust-anchor-foundry utility supports one of three different output formats. In every case, black and white graphics data is output at a 300 dot-per-inch resolution.

The trust-anchor-foundry utility outputs pages of bar code labels; these pages are not intended to be stored electronically. The tolerance in the bar code print quality being relatively small, any graphic format conversion can alter the print quality irremediably below minimum tolerances. Furthermore, a substandard print quality may be detected only when the bar codes are actually read, which might be decades after the trust-anchor-foundry utility run. For this reason, the trust-anchor-foundry utility directly supports a manufacturer-specific printer command language as the preferred graphics output format. The two bit map formats (i.e. the uncompressed .bmp format and the lossless .gif compressed graphic format) are supported essentially for demonstration purposes.

The format is chosen as a compile-time selection done in the source file abcd_config.h:

#define OUTPUT_FORMAT FORMAT_HP_PCL

selects the Hewlett-Packard Printer Command Language format as documented for the LaserJet series II Printer ([18]) (source code part of files bin2code128c.cpp, bin2code128c.h, and code128.cpp),

#define OUTPUT_FORMAT FORMAT_GIF

selects the .gif graphic file format (see source files giflzw.cpp and giflzw.h), and

#define OUTPUT_FORMAT FORMAT_BMP

selects the .bmp graphic file format (see source files bmp.cpp and bmp.h).

2.3.6 Cryptographic Algorithm Interoperability Dependencies

2.3.6.1 Trust Anchor Key Algorithm Selection

The functional aspects of key generation for RSA is modeled on the bind software version 9 ([16]) dnskey-keygen utility, but limited to RSA with SHA-1. The modulus key size range is from 512 to 4096 bits. An equivalent to the "large exponent" command option (-e) is used to select between a public exponent value of 3 or 65537. No attempt is made to support the meaningless dnskey-keygen option "-s strength."

The key generation for the PEKE cryptosystem ([6]) allows the selection of a public modulus size like the RSA modulus. The two PEKE operational parameters k and t are small integers specified directly as a utility command line option. The PEKE operational parameters S and C are larger integer values that can be specified either a) as an exact power of two, b) as a direct hexadecimal value, c) as a size indication applicable to a random value, d) as a size indication applicable to a random prime number.

The MASH cryptographic primitive has composite modulus generation requirements similar to the PEKE cryptosystem, i.e. a modulus N such that the $x^2 \ mod \ N$ primitive has the number-theoretic properties well studied since the Rabin-Williams cryptosystem proposal. The BBS pseudo-random number generator (see section 2.2.1 above) also needs this same type of composite modulus. The MASH small prime selection commands yet another parameter selection strategy (i.e. the three most significant bits must be one). Except for the possible benefits of source code reuse, the parameter generation requirements for MASH and BBS are not strict software dependencies.

The currently distributed software implements the classical prime number generation algorithm, i.e. the Miller-Rabin compositeness rejection test is used to select integers that are strongly believed to be prime. See the source file find_rnd_large_prim.cpp.

2.3.6.2 Trust Anchor Key Data Representation

The public key data representation details are mandated by the relevant public key application standard document (e.g. [19] for DNSSEC use of RSA public keys). The data representation for the private key counterpart is much less subject to standardized interoperability provisions. Nonetheless, there may be effective software dependencies in this area as well.

For DNSSEC, the bind software ([16]) appears to set an ad-hoc standard data representation for private keys using the semantic rules of the openssl software ([15]) . See for instance the source file openssl-0.9.8/crypto/rsa/rsa_gen.c for the meaning of RSA

private key fields.

3. Other Software Planning Factors

3.1 Patent and Copyright Licensing

The TAKREM procedure is a patent pending technology ([20]). 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.

3.2 Compliance with Government Issued Key Management Guidance

After 60 years of crypto-policy meandering, there is still a lot of confusion around government issued key management guidance. Moreover, trust anchor key management is a small (yet critical) corner of the IT security technology field where innovation is in demand. As a consequence, the lack of "established standard compliance" intrinsic to the server-side implementation of the TAKREM procedure should not impede a deployment effort that encompasses public sector applications.

Perhaps this view is supported by the following excerpt from reference [21], at section 10:

> "The types of key management components that are required for a specific cryptographic device and/or for suites of devices used by organizations should be standardized to the maximum possible extent, [...] A developer may choose to employ non-compliant key management as a result of security, accuracy, utility, or cost considerations. However, such developments should conform as closely as possible to established key management recommendations."

The same idea is repeated in the companion reference [22], at section 5:

> "Where the application of current key management products and services results in reduced security, accuracy, utility, or added cost to a cryptographic application, then an organization may initiate efforts to develop and implement other key management products and services types, variations, and, as necessary, production processes. However, such efforts should conform as closely as possible to established key management recommendations."

3.3 Applicability of Security Certification Principles

Before becoming matters for politicking between security vendors and client auditors, the quality principles used in the "Common Criteria" [23] make sense (as a productive intellectual exercise for avoiding security system design flaws). In this respect, a well-defined scope of security certification is an important starting point.

In the case of the TAKREM procedure, it should be argued at the outset of a certification program that the initial trust anchor key generation procedure implementation (i.e. the almost anecdotical use of the trust-anchor-foundry utility) has no "security functional requirements." The initial trust anchor key generation is merely a manufacturing step for printed pages holding numbers (key material). To some extent, the use of these pages might create "security assurance requirements," but going too far in this direction would devote excessive certification resources to a process step that is used only once in the lifetime of a security application, and to system functions that are completed *before* the key material is associated with any entity.

4. The trust-anchor-foundry Utility

4.1 Software Utility Design

The main file is the source file bin2code128c.cpp, which is overloaded with irrelevant pixel font initialization and many page layout details. Once the command line decoding is complete, the files takrem_proc.cpp and takrem_gener.cpp do the actual key generation and bar code page contents preparation. The MASH cryptographic primitive implementation is in the source files mash.cpp and mash.h.

A separate electronic document shows a sample output from the trust-anchor-foundry utility ([24]), resulting from the following command line:

trust-anchor-foundry -aRSASHA1 -b2048 -e --output sample_%2.2d.bmp --mash-N 2048 --mash-p 384 --repl 3 --title "Sample trust-anchor-foundry output"

The pages are sorted according to the natural grouping of separate components for their distribution to the safe box custodians, e.g. in the case of 2 components: the public digests page first, and then the pages for the first component of each key pair, and then the pages for the second component of each key pair.

When a series of trust anchor key is generated with the trust-anchor-foundry utility, it is not associated with any entity. Instead, it is merely identified with a random hexadecimal string, eight character long. The title line on each page allows the matching of bar code pages needed to recover the key pairs. The title line contents is shown below with explanations:

```
      +---- key pair series identification
      |
      |    +--- rank of component set created for this key pair series
      |    |+-- custodian entity to which this page is assigned
      |    ||+- custodian entity backup instance indication, "0" or "1"
      |    |||
      |    |||   +- rank of key pair within this series
      |    |||   |
      |    |||   |   +-- either a "pub"lic portion or the "pri"vate counterpart
      |    |||   |   |
      |    |||   |   |   +---- the "A", "B", "C", ... pairing
      |    |||   |   |   |+--- component rank in the split components pairing
      |    |||   |   |   || +- number of components in this pairing
      |    |||   |   |   || |
2BC454F0.1a0.51.pub.B2/3      Title                        Page 1/...
```

The first page(s) hold the TAKREM digests of trust anchor key pairs, as shown below:

```
      +--------------- key pair series identification
      |     +---------- "digests" as an indication of the page contents
      |     |
2BC454F0.digests              Title                        Page 1/...
```

The supported arrangements are listed below:

```
Supported arrangements: 2/2b, 2/2, 2/3b, 2/3, 3/3b, 3/3, 2/4b,
2/4, 3/4b, 3/4, 4/4b, 4/4, 2/5b, 2/5, 3/5b, 3/5, 4/5b, 4/5, 5/5b,
5/5, 2/6b, 2/6, 3/6b, 3/6, 4/6, 5/6, 6/6b, 6/6, 2/7b, 2/7, 3/7,
4/7, 5/7, 6/7, 7/7, 2/8b, 2/8, 3/8, 4/8, 5/8, 6/8, 7/8, 8/8,
2/9b, 2/9, 3/9, 7/9, 8/9, 9/9, 2/10b, 2/10, 3/10, 8/10, 9/10,
10/10.
```

Parameter generation for MASH

The parameter generation for the MASH cryptographic primitive allows the selection of three sizes, respectively for the composite modulus number used in the MASH round function, the prime number used in the MASH final reduction function, and the salt field allowed by the TAKREM use of MASH. Moreover, a utility command line option allows a selection between MASH-1 and MASH-2.

Principles for varying the trust anchor key parameters in a key pair series

The trust-anchor-foundry is intended for the initial generation of a large number of key pairs, perhaps not all of the same size. The command line arguments that define cryptographic parameter sizes can be repeated, provided a "replication count" option occurs before any such repetition. A replication count applies to the cryptographic parameter sizes immediately preceding it, unless there is a single one occurring first in the command line arguments.

A replication count is given by the following parameter:

-r number
--replication number

The following parameters are not influenced by the replication count:

--exponent-large
-a algorithm

The following parameters can be varied in the replication sections:

-b keysize 512..4096

--peke-k number
--peke-t number
--peke-S [two[0..9]... | <number> | rnd[0..9]... | prime[0..9]...]
--peke-C [two[0..9]... | <number> | rnd[0..9]... | prime[0..9]...]

--mash-N number 512..4096
--mash-p number
--mash-salt number (bit size rounded up to an octet boundary)

4.2 Software Utility Command Line Syntax

```
trust-anchor-foundry -- A TAKREM cryptographic key generation utility with
                        output as code 128C barcode HP PCL printout

Copyright (C) 2005 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.

Synopsis

   trust-anchor-foundry [ <option> ...]
```

```
-t<title string>
--title <title string>
   Sets the title string to appear on top of each page.

-p[letter |a4 ]
--page [ letter | a4 ]
   Selects between letter size (8.5 by 11 inches) or the A4 format (210
   by 297 mm).

-o<file name>
--output  <file name>
   Specifies the output file name (default is standard output).

-s<spool page count>
--spool-pages  <spool page count>
   Specifies a spool page count so that printer spool object size can remain
   reasonable.

-a<algo>
--algorithm  <algo>
   Specifies the algorithm, among RSASHA1 RSAMD5 RSA PEKEV1.25 PEKEV0 PEKE.

-r<number>
--replication  <number>
   Specifies the replication count (can be repeated to vary the cryptographic
   parameter sizes).

-c<arrangement>[|,<arrangement>]...[|:<digest page copies>]
--components  <arrangement>[|,<arrangement>]...[|:<digest page copies>]
   Specifies one or more component arrangements, i.e. "<N>/<M>" or "<N>/<M>b"
   for <N> components out of <M> custodians (<N> and <M> are integers), and a
   trailing "b" indicates a backup set of components for each custodian; the
   number of digest page copies is a decimal value (default 1 copy of digests
   bar code pages).

--arrangements
   Lists the supported component arrangements.

-b<keysize>
--bitsize  <keysize>
   Specifies the public key modulus size.

-e
--exponent-large
   Specifies a larger public RSA exponent, i.e. 2^16+1 instead of 3.

--peke-k  <number>
   Specifies the PEKE paremeter k.

--peke-t  <number>
   Specifies the PEKE paremeter t.

--peke-S  [ two[0..9]... | <hex number> | rnd[0..9]... | prime[0..9]... ]
   Specifies the PEKE paremeter S, either as an exact power of two, a large
```

```
   number, a random number of the specified bit size, or a random prime of the
   specified bit size.

--peke-C  [ two[0..9]... | <hex number> | rnd[0..9]... | prime[0..9]... ]
   Specifies the PEKE paremeter C, either as an exact power of two, a large
   number, a random number of the specified bit size, or a random prime of the
   specified bit size.

--mash-N  <keysize>
   Specifies the modulus size for the MASH function.

--mash-p  <number>
   Specifies the prime size for the MASH function.

--mash-salt  <number>
   Specifies the salt size (bit size rounded up to an octet boundary) for the
MASH function input.

-h
--help
   Displays this help message.

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

4.3 Software Build-Time Options

Every software built-time option described here is controlled by preprocessor symbols found in the source file abcd_config.h.

The graphics output format selection is a software built-time option described in section 2.3.5.

For the applications where the PEKE cryptosystem support meets the requirements, **#define**'ing the preprocessor symbol HAS_RSA_SUPPORT to zero removes support for the RSA cryptosystem. In the unlikely event that the PEKE cryptosystem support is not necessary, the preprocessor symbol HAS_PEKE_SUPPORT can be set to zero.

The BBS pseudo-random generator requires a large composite modulus that is generated by the trust-anchor-foundry utility when the preprocessor symbol HAS_BBS_DESIGN_SUPPORT is set to non-zero. Twenty such composite numbers are output by the utility program so they can be cut-and-pasted as static initialization data in the source file super_prng.cpp. A purist who challenges this parameter self-reconstructing approach may disable the BBS generator by setting the preprocessor symbol BBS_BIG_DESIGN_NOT_SMALL to zero.

5. The takrem-barcode Bar Code Reading Utility

5.1 Software Utility Design

The takrem-barcode utility supports bar code reading and component merging for TAKREM purposes, creating local files of binary data. This allows the creation of three types of files, i.e.

1) the public digest array file, passed to the takrem-roll-dig utility and perhaps the takrem-roll-pub utility (hereafter TAKREM-D),
2) the public portion of a single key pair, passed to the takrem-roll-pub utility and the takrem-roll-pri utility (hereafter TAKREM-PUB), and
3) the private counterpart of a public key, passed to the takrem-roll-pri utility (hereafter TAKREM-PRI).

Generally, the reading of a bar code label provides an input data element in an interactive computer session. The operator controls the interactive session, thus the bar code input is interpreted in the context created by the operator. Typically, a bar code label holds no control information. This is the case for TAKREM bar code labels which convey neither label rank data within the series of labels on the page, nor the type of data among the three options above (TAKREM-D, TAKREM-PUB, or TAKREM-PRI), and nor the component rank in the case of multiple component storage. Note that input order does not matter for the multiple components of a given data of a given type.

The takrem-barcode utility is a command-line tool that reads interactive commands from the user and accepts bar code label input from a serial port, assuming a properly initialized Opticon OPT-6125 bar code reader is connected to the serial port. This interactivity is implemented in the source files lab_comm_select.cpp, lab_comm_console.cpp, lab_comm_serial.cpp, lab_comm_select.h, and lab_comm_serial.h. The generic user command parsing mechanism is implemented in the files abcd_maintcmd.cpp and abcd_maintcmd.h, while personalization of the command set is implemented in the files abcd_maintcmd_dispatch.cpp and abcd_maintcmd_dispatch.h. The specific commands are implemented in the files abcd_maintcmd_com.cpp and abcd_maintcmd_license.cpp. The bar code decoding logic is implemented in the files code128decode.cpp and code128decode.h.

The Opticon OPT-6125 bar code reader reader equipment must be initialized by scanning the following sequence of bar code labels. This equipment initialization is to be done once and for all, i.e. the equipment's memory retains its setting upon losing power. Refer to manufacturer's documentation ([17]) for further explanation on the meaning of these sequences.

ZZ Start/end program menu -[[-

U2	RS232 factory defaults	-V3-	
ZZ	Start/end program menu	-[[-	
ZZ	Start/end program menu	-[[-	
S0	Single read	-T1-	
A6	Code 128 only	-B7-	
ZZ	Start/end program menu	-[[-	
ZZ	Start/end program menu	-[[-	
O9	Code 128	-P:-	
0E	E	-1F-	
0F	F	-1G-	
0G	G	-1H-	
0H	H	-1I-	
ZZ	Start/end program menu	-[[-	

ZZ	Start/end program menu	-[[-	
M9	Code 128	-N:-	
0J	J	-1K-	
0K	K	-1L-	
0L	L	-1M-	
0M	M	-1N-	
ZZ	Start/end program menu	-[[-	
ZZ	Start/end program menu	-[[-	
Y5	5 seconds	-Z6-	
X2	Three times redundant	-Y3-	
ZZ	Start/end program menu	-[[-	

The takrem-barcode utility has interactive commands to indicate the start and end of a series of labels (respectively CODE128 START and CODE128 END).

At the end of such a series of labels (upon the command CODE128 END), the decoded bar code series is assigned to one of the options above (TAKREM-D, TAKREM-PUB, or TAKREM-PRI), and to a component rank in the case of multiple component storage (applicable to TAKREM-PUB, or TAKREM-PRI). When the single or last component is read, a

corresponding binary file is written. The data validation is deferred to the utilities takrem-roll-dig, takrem-roll-pub, and/or takrem-roll-pri. The bar code to file process is controlled by the takrem-barcode utility interactive commands CODE128 TAKREM-D, CODE128 TAKREM-PUB, and CODE128 TAKREM-PRI (see below for a more detailed explanation).

The state maintained for the above process of reading TAKREM label series by the takrem-barcode utility can be reported with the interactive command CODE128 TAKREM.

5.2 Software Utility Command Line Syntax

The program startup help text is shown below:

```
./takrem-barcode -- A TAKREM utility for scanning bar code pages.

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-barcode [|-h|--help|-w|--warranty]

Options

      -h
      --help
           Displays this help message.

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

The interactive command help is shown below:

```
Command:    "COM SHOW"
Description:      Display the current serial communications port
configuration.

Command:    "COM PORT <string>"
Description:      Set the serial communications port name.
Command parameters:
       o    A quoted string of length in the range 1 to 50 inclusive.

Command:    "COM SPEED <unsigned>"
Description:      Set the baud rate for the serial communications port.
Command parameters:
```

- An unsigned integer in the range 1200 to 115200 inclusive.

Command: "COM RTS <unsigned>"
Description: Set the serial communications port RTS configuration (0 or 1).
Command parameters:
- An unsigned integer in the range 0 to 1 inclusive.

Command: "COM DTR <unsigned>"
Description: Set the serial communications port DTR configuration (0 or 1).
Command parameters:
- An unsigned integer in the range 0 to 1 inclusive.

Command: "COM CONNECT"
Description: Connect the serial communications port.

Command: "COM DISC"
Description: Disconnect the serial communications port.

Command: "CODE128 START"
Description: Start a bar code label scan series.

Command: "CODE128 END"
Description: End a bar code label scan series.

Command: "CODE128 TAKREM-D <string>"
Description: Indicate to expect TAKREM digest information (upon CODE128 END), with output file name.
Command parameters:
- A quoted string of length in the range 1 to 80 inclusive.

Command: "CODE128 TAKREM-D"
Description: Indicate to expect TAKREM digest information (upon CODE128 END).

Command: "CODE128 TAKREM-PUB <unsigned> <string>"
Description: Indicate to expect TAKREM public information (upon CODE128 END), with output file name and component count.
Command parameters:
- An unsigned integer in the range 1 to 4 inclusive.
- A quoted string of length in the range 1 to 80 inclusive.

Command: "CODE128 TAKREM-PUB"
Description: Indicate to expect TAKREM public information (upon CODE128 END).

Command: "CODE128 TAKREM-PRI <unsigned> <string>"
Description: Indicate to expect TAKREM private information (upon CODE128 END), with output file name and component count.
Command parameters:
- An unsigned integer in the range 1 to 4 inclusive.
- A quoted string of length in the range 1 to 80 inclusive.

```
Command:    "CODE128 TAKREM-PRI"
Description:      Indicate to expect TAKREM private information (upon CODE128
END).

Command:    "CODE128 TAKREM"
Description:      Display TAKREM barcode input status.

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

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

6. The takrem-roll-dig Utility

6.1 Software Utility Design

The takrem-roll-dig utility reads a binary file recovered from bar code pages, and creates a TAKREM-specific file format for digest array. The current version of the takrem-roll-dig utility is specific to the DNSSEC application. There are two added values in the conversion process:

a) some context information is affixed to the digest array, especially the domain name to which the trust anchor key is assigned, and

b) the output file format allows the merging of many digest arrays (e.g. for different DNS zones) in a single file using standard text editing tools.

6.2 Software Utility Command Line Syntax

```
takrem-roll-dig -- A TAKREM utility for DNS Trust Anchor Key rollover,
                   digest array text file preparation.

Copyright (C) 2005 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.

The takrem-roll-dig utility converts a raw TAKREM digest array
file into the textual representation used for initial trust
anchor key distribution in the DNSSEC application of the TAKREM
procedure. A raw TAKREM digest file is normally obtained from the
barcode printout created by the trust-anchor-foundry utility.
```

```
Since the output of the takrem-roll-dig conversion utility
contains all the information from the input file in a convenient
format, there is little incentive to run this utility more than
once after a given run of the trust-anchor-foundry utility. It is
expected that a number of files created by the takrem-roll-dig
utility, e.g. for a set of DNS islands of trust, will be
concatenated into a single initial trust anchor key distribution
file or configuration element.

Synopsis:

takrem-roll-dig [options...] <raw digest file>

<raw digest file>
     File name for the takrem-roll-dig utility input.

Options

-z<name>
--zone=<name>
     Specifies the zone name for the TAKREM digest array, as an
     absolute DNS domain name. The default, example.com. is of
     little practical use, but the TAKREM digest file in the
     textual representation can be later edited.

-r<number>
--reference=<number>
     Specifies the TAKREM reference number allowing an optional
     search optimization in the trust anchor key rollover
     procedure in the DNS resolver software. Unless the reference
     numbers are tracked by a zone manager in the event of
     multiple digest arrays, the default processing should be
     used. The default is a simple hash of the takrem-roll-dig
     utility input data.

-o<file>
--output=<file>
     Specifies the utility output file name. By default, the
     utility output is directed to the standard output.

-h
--help
     Displays this help message.

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

7. The takrem-roll-pub Utility

7.1 Software Utility Design

The takrem-roll-pub utility reads two files, and creates text formatted configuration lines for DNS zone configuration. These lines are respectively for a DNSKEY RR, an SDDA RR authenticating this DNSKEY contents, and an optional DS RR targeting this DNSKEY RR.

7.2 Software Utility Command Line Syntax

```
takrem-roll-pub -- A TAKREM utility for DNS Trust Anchor Key rollover
                   public key preparation for rollover operation.

Copyright (C) 2005 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.

The takrem-roll-pub utility prepares the DNS textual
representation of DNS resource records associated with a trust
anchor key rollover operation for a given DNS zone. The utility
output includes a DNSKEY record, an SDDA record, and a DS record.
The SDDA record is specific to the TAKREM rollover procedure, and
the DS record is the normal DNSSEC secure parental delegation
which may be used e.g. in the transition from an "island of
trust" to a normal secured zone. It is expected that the
takrem-roll-pub utility is merged to the zone data (and parental
zone data in the case of the DS record) using text file editing
tools.

Synopsis:

takrem-roll-pub [options...] <digest data file> <public takrem file>

<digest data file>
     This specifies an input file name. The takrem-roll-pub
     utility uses the TAKREM digest array input data for to
     partially validate the key rollover as the DNS resolver
     software will do. The digest data file may be either a raw
     digest file for a single zone (created by the
     trust-anchor-foundry utility) or a textual representation
     for one or more zones (created by the takrem-roll-dig
     conversion utility). With the textual alternative, the zone
     name and reference number values are supplied with the data
     file contents, and need not be provided as utility command-
     line arguments.

<public takrem file>
     This specifies the input file name for the TAKREM public
     information pertaining to a new key to be rolled in,
```

```
     including the public key value and the MASH algorithm
     parameters. Prior to the rollover operation, and since it
     was created by the trust-anchor-foundry utility, this public
     information should have been kept in some secret dead
     storage arrangement. It is thus normally recovered from
     multiple bar code page printouts created by the
     trust-anchor-foundry utility when the TAKREM digest array
     was also created.

Options

-z<name>
--zone=<name>
     Specifies the zone name for the TAKREM digest array, as an
     absolute DNS domain name. If no value is found in the digest
     data file, the default, example.com. is of little practical
     use. (The utility output file can be later edited to change
     the zone name, but that invalidates the DS record.)

-r<number> or -r or -r-
--reference=<number> or --reference or --no-reference
     Specifies the TAKREM reference number allowing an optional
     search optimization in the trust anchor key rollover
     procedure in the DNS resolver software. Preferably, the
     value from the digest data file should be relied upon, in
     which case the presence of a reference number in the SDDA
     record is requested by the -r or --reference option without
     a number, and its absence is requested by the -r- or
     --no-reference option. The default is a reference number in
     the SDDA record if and only if one is found in the digest
     data file.

-t<number>
--ttl=<number>
     Specifies the time-to-live (TTL) value to be used in
     constructing the DNSKEY, SDDA, and DS records. The parameter
     value is a duration expressed in seconds. The default value
     is one day (86400).

-s<time>
--start=<time>
     Specifies the cryptoperiod start date and time as an
     absolute value (e.g. 20051230115900) or as a relative time
     from now, expressed in seconds as a signed number. The
     default is one hour in the past (-3600). The cryptoperiod
     start time is reflected in the SDDA record like a signature
     inception time in an RRSIG record.

-e<time>
--end=<time>
     Specifies the cryptoperiod start date and time as an
     absolute value (e.g. 20060730115900) or as a time relative
     from now expressed in days as a signed number. The default
     is about 7 months in the future (+213). The cryptoperiod end
```

```
     time is reflected in the SDDA record like a signature
     expiration time in an RRSIG record.

--mash-1 or --mash-2
     Specifies whether the SDDA record is constructed with the
     MASH-1 or MASH-2 algorithm variant. The default is MASH-2.

--ds-rr or --no-ds-rr
     Specifies whether a DS resource record is created in the
     output file for the DNSKEY resource record. The default is
     no such creation unless an option --sha-1 or --sha-256
     is present.

--sha-1 or --sha-256
     Specifies whether the DS record is constructed with the
     SHA-1 or SHA-256 digest algorithm. The default is SHA-256.

-o<file>
--output=<file>
     Specifies the utility output file name. By default, the
     utility output is directed to the standard output.

-h
--help
     Displays this help message.

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

8. The takrem-roll-pri Utility.

8.1 Software Utiltiy Design

The takrem-roll-pri utility reads two binary files, and creates bind representation of an RSA private key for the dnssec-signzone utility.

If the function performed by the bind software utility is to be performed, either totally or in part, in a dedicated digital signature device, the bar code reading function (currently in the takrem-barcode utility) and the private key import function (currently in the takrem-roll-pri utility) would need to be integrated in the dedicated signature device. This motivated the use of a serial interface for the bar code reader input device (e.g. Opticon OPT-6125), instead of a USB interface that would be more ubiquitous among the personal computer equipment available to experimenters with the present set of TAKREM server management tools.

8.2 Software Utility Command Line Syntax

```
takrem-roll-pri -- A TAKREM utility for DNS Trust Anchor Key rollover
                   private key recovery for rollover operation.

Copyright (C) 2005 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.

The takrem-roll-pri utility prepares the textual representation
of a DNSSEC private key, in a format compatible with the bind
software [[... reference RFC2313, RFC2437, RFC3447 ...]].
Normally, the private key is the counterpart of a new public key
introduced with a trust anchor key rollover operation for a given
DNS zone. The input files to the takrem-roll-pri utility are
normally recovered from multiple bar code page printouts created
by the trust-anchor-foundry utility when a TAKREM digest array
was also created. Prior to the run of the takrem-roll-pri utility
and since they were created by the trust-anchor-foundry utility,
these input files should have been kept in some secret dead
storage arrangement.

Synopsis:

takrem-roll-pri [options...] <public takrem file> <private takrem file>

<public takrem file>
     This specifies the input file name for the TAKREM public
     information pertaining to the public counterpart of the
     private key to be prepared, including the public key
     parameters and the MASH algorithm parameters (only the
     public key parameters are relevant to the takrem-roll-pri
     utility).

<private takrem file>
     This specifies the input file name for the private key
     information.

Options

-o<file>
--output=<file>
     Specifies the utility output file name. By default, the
     utility output is directed to the standard output.

-h
--help
     Displays this help message.

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

9. 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-01.txt, December, 2005

[4] Thierry Moreau, "The SEP DNSKEY Direct Authenticator DNS Resource Record (SDDA-RR)", Internet Draft (work-in-progress), draft-moreau-dnsext-sdda-rr-01.txt, December, 2005

[5] Thierry Moreau, "DNS Resolver Software Change Planning for Trust Anchor Key Management Based on TAKREM", CONNOTECH Experts-conseils inc., Document Number C003595, 2006/02/02

[6] Thierry Moreau, "PEKE, Probabilistic Encryption Key Exchange, 10 Years Later, Including the PEKEv1.25 Specifications", CONNOTECH Experts-conseils inc. Document Number C003449, 2005/10/03, available at http://eprint.iacr.org/2005/183

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

[8] Blum, L., Blum, Manuel, and Shub, M., "A Simple Unpredictable Pseudo-random Number Generator", SIAM Journal of Computing, vol. 15, no. 2, May 1986, pp 364-383

[9] Thierry Moreau,"A practical 'perfect' pseudo-random number generator", CONNOTECH Experts-conseils, Inc., February 27, 1996,

[10] Thierry Moreau, "The 'Multiplexed and Twisted' GFSR, a Flexible Scheme for the Creation of Pseudo-random Generators", CONNOTECH Experts-conseils Inc., April 2000 (revised April 2005 and October 2005), http://www.connotech.com/mtgfsr_04.pdf

[11] Thierry Moreau, "The Frogbit cipher, a data integrity algorithm", CONNOTECH

Experts-conseils Inc., January 1997 (revised April 2005), http://www.connotech.com/ecrypt/frogbit.pdf

[12] http://josefsson.org/gnutls/, The GNU Transport Layer Security Library [Overview]

[13] http://josefsson.org/gnutls/releases/libtasn1/, Index of /gnutls/releases/libtasn1

[14] http://directory.fsf.org/libgcrypt.html, Libgcrypt - Cryptographic library

[15] http://www.openssl.org/, OpenSSL: The Open Source toolkit for SSL/TLS

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

[17] Opticon, "User's Manual, OPT-6125 Series Handheld CCD Scanner" Manual number 25-ULGPMU01-01, April 2002

[18] Hewlett-Packard, "LaserJet Series II Printer Technical Reference Manual", document number 3440-90905, May 1987

[19] R. Arends, R. Austein, M. Larson, D. Massey, S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005

[20] 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

[21] Elaine Barker, William Barker, William Burr, William Polk, and Miles Smid, "Recommendation for Key Management - Part 1: General", NIST Special Publication 800-57 part 1, August, 2005

[22] Elaine Barker, William Barker, William Burr, William Polk, and Miles Smid, "Recommendation for Key Management - Part 2: Best Practices for Key Management Organization", NIST Special Publication 800-57 part 2, 2005

[23] Common Criteria Project Sponsoring Organizations, i.e. Canada: Communications Security Establishment, France: Service Central de la Sécurité des Systèmes d'Information, Germany: Bundesamt für Sicherheit in der Informationstechnik, Netherlands: Netherlands National Communications Security Agency, United Kingdom: Communications-Electronics Security Group, and US: National Institute of Standards and Technology and National Security Agency, "Common Criteria for Information

Technology Security Evaluation", part 1: "Introduction and general model", part 2: "Security functional requirements", and part 3: "Security Assurance Requirements", Version 2.2, January 2004.

[24] untitled document, http://www.connotech.com/takrem_tools/sample_pages_01.pdf.gz