Libseclog

Generated by Doxygen 1.8.2

Sun Jul 28 2013 10:25:26

Contents

Chapter 1

Libseclog documentation

Libseclog is a library for C language that make possible the generation of secure log entries using several secure logging schemes. Its main features are and design principles are:

- Set of supported logging system easily extensible
- Log entries generation procedure that follows the `Common Event Expression (CEE)` directives
- Simple and intuitive API

To install the library, follow the instructions in the page: Install Libseclog.

To build programs that use Libseclog, follow the instruction in the page: Use Libseclog.

To get information about the API, see the documentation in seclog.c file reference page.

1.1 List of supported logging schemes

- **SK** – Scheme for secure logging proposed by B. Schneier and J. Kelsey in *`Secure Audit Logs to Support Computer Forensics`*

1.2 Libseclog design

Chapter 2

Install Libseclog

2.1 Dependencies

The following dependencies need to be resolved:

- Autoconf
- Libtool
- OpenSSL (>=1.0.0)
- UUID
- Jansson (>=2.2.1)

2.1.1 Debian-like systems

To install the dependencies run the following commands:

```
apt-get update
apt-get install autoconf libtool libssl-dev uuid-dev libjansson-dev
```

2.2 Installation

```
tar zxvf libseclog-<VERSION>.tar.gz
cd libseclog
./autogen.sh
```

Before proceedings with the installation it's **strongly recommended** to take a look at the output provided by the command

```
./configure --help
```

then proceed with the installation

```
./configure
make
make install (as root)
```

2.2.1 Enable CEE logs

To support CEE-compliant log entries generation it's necessary to install the library **libumberlog**. The installation instruction are taken from the official libumberlog documentation.

```
git clone https://github.com/deirf/libumberlog.git
cd libumberlog
autoreconf -i
./configure
make
make install (as root)
```

Then, include the `-enable-ceelog` when run configure:

```
./configure --enable-ceelog
make
make install (as root)
```

2.2.2 Python bindings

To enable the generation of the Python bindings, use the option `-enable-python`.

```
./configure --enable-python
make
make install (as root)
```

To install the Python package, run

```
cd binding/python
pip install pySeclog-0.1.0_<BUILD>.tar.gz
```

Chapter 3

Use Libseclog

Libseclog supports `pkg-config`

```
pkg-config libseclog --cflags --libs
-I/usr/local/include  -L/usr/local/lib -lseclog
```

Chapter 4

Module Index

4.1 Modules

Here is a list of all modules:

Chapter 5

Namespace Index

5.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 6

Data Structure Index

6.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 7

File Index

7.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 8

Module Documentation

8.1 ERROR_CODES

Macros

- #define SECLOG_ERR_PARAM 100
- #define SECLOG_ERR_CTX 101
- #define SECLOG_ERR_UNKNOWN_SCHEME 102
- #define SECLOG_ERR_FILE_NOT_FOUND 103
- #define SECLOG_ERR_SYS 104
- #define SECLOG_ERR_OPENSSL 105
- #define SECLOG_ERR_UNKNOWN_MODE 106
- #define SECLOG_ERR_EVENT 107
- #define SECLOG_ERR_INVALID_MODE 108
- #define SECLOG_ERR_JSON 109
- #define SECLOG_ERR_SCHEME_NOT_IMPLEMENTED 110
- #define SECLOG_ERR_VRFY_CTX 111

8.1.1 Detailed Description

8.1.2 Macro Definition Documentation

8.1.2.1 #define SECLOG_ERR_PARAM 100

Input parameter checking failure

8.1.2.2 #define SECLOG_ERR_CTX 101

Failure about libseclog context

8.1.2.3 #define SECLOG_ERR_UNKNOWN_SCHEME 102

Unknown secure logging scheme specified

8.1.2.4 #define SECLOG_ERR_FILE_NOT_FOUND 103

File not found

8.1.2.5 #define SECLOG_ERR_SYS 104

System error

8.1.2.6 #define SECLOG_ERR_OPENSSL 105

OpenSSL error

8.1.2.7 #define SECLOG_ERR_UNKNOWN_MODE 106

Unknown mode specified

8.1.2.8 #define SECLOG_ERR_EVENT 107

Failure about event

8.1.2.9 #define SECLOG_ERR_INVALID_MODE 108

Specified an invalid mode for the current function

8.1.2.10 #define SECLOG_ERR_JSON 109

JSON (Jansson) failure

8.1.2.11 #define SECLOG_ERR_SCHEME_NOT_IMPLEMENTED 110

Specified secure logging scheme not yet implemented

8.1.2.12 #define SECLOG_ERR_VRFY_CTX 111

Failure about verify context

8.2 RETURN_CODES

Macros

- #define SIGN_VERIFICATION_FAILURE 200
- #define M0_VERIFICATION_FAILURE 201
- #define M1_VERIFICATION_FAILURE 202
- #define AUTHKEY_NOT_PRESENT 203
- #define LOG_VERIFICATION_FAILURE 204
- #define CERT_VRFY_FAILURE 205
- #define ALREADY_VERIFIED 206
- #define ABNORMAL_CLOSE 207
- #define LOG_ENTRIES_LIMIT_REACHED 208

8.2.1 Detailed Description

8.2.2 Macro Definition Documentation

8.2.2.1 #define SIGN_VERIFICATION_FAILURE 200

Signature verification failure

8.2.2.2 #define M0_VERIFICATION_FAILURE 201

M0 message verification failure

8.2.2.3 #define M1_VERIFICATION_FAILURE 202

M1 message verification failure

8.2.2.4 #define AUTHKEY_NOT_PRESENT 203

Authentication key not present in the database

8.2.2.5 #define LOG_VERIFICATION_FAILURE 204

Log file verification failure

8.2.2.6 #define CERT_VRFY_FAILURE 205

X509 certificate verification failure

8.2.2.7 #define ALREADY_VERIFIED 206

Log entry already verified

8.2.2.8 #define ABNORMAL_CLOSE 207

Abnormal logging session closure

8.2.2.9 #define LOG_ENTRIES_LIMIT_REACHED 208

Reached maximum number of log entries per session

8.3 SCHEME_AND_MODE_FLAGS

Macros

- #define U 0x01
- #define T 0x02
- #define V 0x04
- #define SK 0x10
- #define MT 0x20

8.3.1 Detailed Description

8.3.2 Macro Definition Documentation

8.3.2.1 #define U 0x01

Select the U mode (client)

8.3.2.2 #define T 0x02

Select the T mode (server)

8.3.2.3 #define V 0x04

Select the V mode (auditor)

8.3.2.4 #define SK 0x10

Select the Schneier and Kelsey secure logging scheme

8.3.2.5 #define MT 0x20

Select the Ma and Tsudik secure logging scheme

Chapter 9

Namespace Documentation

9.1 pyseclog Namespace Reference

Namespaces

- namespace pyseclog

9.1.1 Detailed Description

```
Libseclog -- Library for Secure Logging

  Copyright (C) 2013 Politecnico di Torino, Italy
  TORSEC Group -- http://security.polito.it
  Author: Paolo Smiraglia <paolo.smiraglia@gmail.com>

This file is part of Libseclog.
Libseclog is free software: you can redistribute it and/or modify
it under the terms of the Lesser General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

Libseclog is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the Lesser General Public License
along with Libseclog.  If not, see <http://www.gnu.org/licenses/>.
```

9.2 pyseclog.pyseclog Namespace Reference

9.2.1 Detailed Description

```
Libseclog -- Library for Secure Logging

  Copyright (C) 2013 Politecnico di Torino, Italy
  TORSEC Group -- http://security.polito.it
  Author: Paolo Smiraglia <paolo.smiraglia@gmail.com>

This file is part of Libseclog.

Libseclog is free software: you can redistribute it and/or modify
it under the terms of the Lesser General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.
```

```
Libseclog is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the Lesser General Public License
along with Libseclog.  If not, see <http://www.gnu.org/licenses/>.
```

9.3 pyseclog_local Namespace Reference

9.3.1 Detailed Description

```
Libseclog -- Library for Secure Logging

  Copyright (C) 2013 Politecnico di Torino, Italy
  TORSEC Group -- http://security.polito.it
  Author: Paolo Smiraglia <paolo.smiraglia@gmail.com>

This file is part of Libseclog.
Libseclog is free software: you can redistribute it and/or modify
it under the terms of the Lesser General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

Libseclog is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the Lesser General Public License
along with Libseclog.  If not, see <http://www.gnu.org/licenses/>.
```

9.4 setup Namespace Reference

9.4.1 Detailed Description

```
Libseclog -- Library for Secure Logging

  Copyright (C) 2013 Politecnico di Torino, Italy
  TORSEC Group -- http://security.polito.it
  Author: Paolo Smiraglia <paolo.smiraglia@gmail.com>

This file is part of Libseclog.
Libseclog is free software: you can redistribute it and/or modify
it under the terms of the Lesser General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

Libseclog is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the Lesser General Public License
along with Libseclog.  If not, see <http://www.gnu.org/licenses/>.
```

Chapter 10

Data Structure Documentation

10.1 __event_t Struct Reference

`#include <event.h>`

Data Fields

- char * desc
- unsigned int desc_len
- unsigned char type [EVENT_TYPE_LEN]
- char ** kws
- int n_kws

10.1.1 Detailed Description

Structure representing the Event object.

10.1.2 Field Documentation

10.1.2.1 char* __event_t::desc

Event description

10.1.2.2 unsigned int __event_t::desc_len

Event description length

10.1.2.3 unsigned char __event_t::type[EVENT_TYPE_LEN]

Event type

10.1.2.4 char** __event_t::kws

Array that will contain n_kws keywords

10.1.2.5 int __event_t::n_kws

Number of defined keywords

The documentation for this struct was generated from the following file:

- /home/paolo/ps/Personal/projects/libseclog/lib/event.h

10.2 __seclog_ctx_t Struct Reference

`#include <seclog.h>`

Data Fields

- int status
- unsigned long counter
- unsigned long max_counter
- char label [BUF_512+1]
- char session_id [UUID_LEN+1]
- char cert_path [MAX_PATH_LEN+1]
- char privkey_path [MAX_PATH_LEN+1]
- char s_cert_path [MAX_PATH_LEN+1]
- char ca_cert_path [MAX_PATH_LEN+1]
- char machine_id [MAX_MACHINE_ID_LEN+1]
- int seclog_scheme
- int seclog_mode
- sk_ctx_t * sk
- mt_ctx_t * mt

10.2.1 Detailed Description

Libseclog context structure

10.2.2 Field Documentation

10.2.2.1 int __seclog_ctx_t::status

Status of the context

10.2.2.2 unsigned long __seclog_ctx_t::counter

Number of the generated log entries in the current session

10.2.2.3 unsigned long __seclog_ctx_t::max_counter

Maximum number of log entries

10.2.2.4 char __seclog_ctx_t::label[BUF_512+1]

Session label

10.2.2.5 char __seclog_ctx_t::session_id[UUID_LEN+1]

Session ID (UUID)

10.2.2.6 char __seclog_ctx_t::cert_path[MAX_PATH_LEN+1]

Certificate file path

10.2.2.7 char __seclog_ctx_t::privkey_path[MAX_PATH_LEN+1]

Private key file path

10.2.2.8 char __seclog_ctx_t::s_cert_path[MAX_PATH_LEN+1]

Path of the trusted server certificate file

10.2.2.9 char __seclog_ctx_t::ca_cert_path[MAX_PATH_LEN+1]

Path of the Certification Authority certificate file

10.2.2.10 char __seclog_ctx_t::machine_id[MAX_MACHINE_ID_LEN+1]

Machine ID (fingerprint of the certificate)

10.2.2.11 int __seclog_ctx_t::seclog_scheme

Secure logging scheme identifier

10.2.2.12 int __seclog_ctx_t::seclog_mode

Secure logging scheme mode identifier

10.2.2.13 sk_ctx_t* __seclog_ctx_t::sk

Context for SK logging scheme

10.2.2.14 mt_ctx_t* __seclog_ctx_t::mt

Context for MT logging scheme

The documentation for this struct was generated from the following file:

- /home/paolo/ps/Personal/projects/libseclog/lib/seclog.h

10.3 __sk_ctx_t Struct Reference

`#include <sk.h>`

Data Fields

- unsigned char k [SESSION_KEY_LEN]
- unsigned char a [AUTH_KEY_LEN]
- unsigned char y_prev [SECLOG_DIGEST_LEN]
- unsigned char x0_digest [SECLOG_DIGEST_LEN]
- struct timeval * d
- struct timeval * d_plus
- unsigned int p
- char authkeys_file [MAX_PATH_LEN+1]
- seclog_ctx_t * parent

10.3.1 Detailed Description

Schneier and Kelsey context structure

10.3.2 Field Documentation

10.3.2.1 unsigned char __sk_ctx_t::k[SESSION_KEY_LEN]

Session key

10.3.2.2 unsigned char __sk_ctx_t::a[AUTH_KEY_LEN]

Authentication key

10.3.2.3 unsigned char __sk_ctx_t::y_prev[SECLOG_DIGEST_LEN]

Previous hash chain element

10.3.2.4 unsigned char __sk_ctx_t::x0_digest[SECLOG_DIGEST_LEN]

Digest of X0

10.3.2.5 struct timeval* __sk_ctx_t::d

Timestamp at initialisation

10.3.2.6 struct timeval* __sk_ctx_t::d_plus

Timeout for the initialisation

10.3.2.7 unsigned int __sk_ctx_t::p

Protocol step

10.3.2.8 char __sk_ctx_t::authkeys_file[MAX_PATH_LEN+1]

File where the authentication keys will be stored

10.3.2.9 seclog_ctx_t* __sk_ctx_t::parent

Pointer to the parent seclog_ctx_t struct

The documentation for this struct was generated from the following file:

- /home/paolo/ps/Personal/projects/libseclog/lib/sk/sk.h

10.4 __sk_vrfy_ctx_t Struct Reference

`#include <sk.h>`

Data Fields

- unsigned char a [AUTH_KEY_LEN]
- unsigned char y_prev [SECLOG_DIGEST_LEN]
- vrfy_ctx_t * parent

10.4.1 Detailed Description

Schneier and Kelsey verification context structure

10.4.2 Field Documentation

10.4.2.1 unsigned char __sk_vrfy_ctx_t::a[AUTH_KEY_LEN]

Authentication key

10.4.2.2 unsigned char __sk_vrfy_ctx_t::y_prev[SECLOG_DIGEST_LEN]

Previous hash chain element

10.4.2.3 vrfy_ctx_t* __sk_vrfy_ctx_t::parent

Pointer to the parent vrfy_ctx_t struct

The documentation for this struct was generated from the following file:

- /home/paolo/ps/Personal/projects/libseclog/lib/sk/sk.h

10.5 __vrfy_ctx_t Struct Reference

`#include <seclog.h>`

Data Fields

- int status
- char session_id [UUID_LEN+1]
- char machine_id [MAX_MACHINE_ID_LEN+1]
- unsigned long counter

- int seclog_scheme
- sk_vrfy_ctx_t * sk

10.5.1 Detailed Description

Verification context structure

10.5.2 Field Documentation

10.5.2.1 int __vrfy_ctx_t::status

Status of the context

10.5.2.2 char __vrfy_ctx_t::session_id[UUID_LEN+1]

Session ID (UUID)

10.5.2.3 char __vrfy_ctx_t::machine_id[MAX_MACHINE_ID_LEN+1]

Machine ID (fingerprint of the certificate)

10.5.2.4 unsigned long __vrfy_ctx_t::counter

Last verified log entry

10.5.2.5 int __vrfy_ctx_t::seclog_scheme

Secure logging scheme identifier

10.5.2.6 sk_vrfy_ctx_t* __vrfy_ctx_t::sk

Verify context for SK logging scheme

The documentation for this struct was generated from the following file:

- /home/paolo/ps/Personal/projects/libseclog/lib/seclog.h

10.6 seclog-server.seclogServer Class Reference

10.6.1 Detailed Description

The documentation for this class was generated from the following file:

- /home/paolo/ps/Personal/projects/libseclog/binding/python/pySeclog/bin/seclog-server.py

Chapter 11

File Documentation

11.1 /home/paolo/ps/Personal/projects/libseclog/lib/common.h File Reference

Macros

- #define CTX_INIT 1
- #define CTX_NOT_INIT !CTX_INIT
- #define SECLOG_SUCCESS 0
- #define SECLOG_ERR_PARAM 100
- #define SECLOG_ERR_CTX 101
- #define SECLOG_ERR_UNKNOWN_SCHEME 102
- #define SECLOG_ERR_FILE_NOT_FOUND 103
- #define SECLOG_ERR_SYS 104
- #define SECLOG_ERR_OPENSSL 105
- #define SECLOG_ERR_UNKNOWN_MODE 106
- #define SECLOG_ERR_EVENT 107
- #define SECLOG_ERR_INVALID_MODE 108
- #define SECLOG_ERR_JSON 109
- #define SECLOG_ERR_SCHEME_NOT_IMPLEMENTED 110
- #define SECLOG_ERR_VRFY_CTX 111
- #define SIGN_VERIFICATION_FAILURE 200
- #define M0_VERIFICATION_FAILURE 201
- #define M1_VERIFICATION_FAILURE 202
- #define AUTHKEY_NOT_PRESENT 203
- #define LOG_VERIFICATION_FAILURE 204
- #define CERT_VRFY_FAILURE 205
- #define ALREADY_VERIFIED 206
- #define ABNORMAL_CLOSE 207
- #define LOG_ENTRIES_LIMIT_REACHED 208
- #define U 0x01
- #define T 0x02
- #define V 0x04
- #define SK 0x10
- #define MT 0x20

Typedefs

- typedef struct __seclog_ctx_t seclog_ctx_t
- typedef struct __vrfy_ctx_t vrfy_ctx_t

11.1.1 Detailed Description

Author

Paolo Smiraglia paolo.smiraglia@gmail.com

11.1.2 Macro Definition Documentation

11.1.2.1 #define CTX_INIT 1

Context initialised

11.1.2.2 #define CTX_NOT_INIT !CTX_INIT

Context not initialised

11.1.2.3 #define SECLOG_SUCCESS 0

Success

11.1.3 Typedef Documentation

11.1.3.1 typedef struct __seclog_ctx_t seclog_ctx_t

Typedef for the struct __seclog_ctx_t

11.1.3.2 typedef struct __vrfy_ctx_t vrfy_ctx_t

Typedef for the struct __vrfy_ctx_t

11.2 /home/paolo/ps/Personal/projects/libseclog/lib/event.c File Reference

```
#include "common.h"
#include "event.h"
#include "helper.h"
#include <stdlib.h>
#include <stdarg.h>
#include <string.h>
```

Functions

- event_t * event_new (const char *e, int nkeys,...)
- int event_free (event_t *e)

11.2.1 Detailed Description

Author

Paolo Smiraglia paolo.smiraglia@gmail.com

11.2.2 Function Documentation

11.2.2.1 event_t* event_new (const char * e, int nkeys, ...)

This function allocates the memory for a new event_t struct.

Parameters

in	*e*	Event description.
in	*nkeys*	Number of specified keywords.
in	...	Variable list of keywords.

Returns

In case of success the function returns a valid pointer to a event_t struct, otherwise NULL.

11.2.2.2 int event_free (event_t * e)

This function free the memory allocated by the function event_new().

Parameters

in	*e*	Valid pointer to event_t struct.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES.

11.3 /home/paolo/ps/Personal/projects/libseclog/lib/event.h File Reference

Data Structures

- struct __event_t

Typedefs

- typedef struct __event_t event_t

Functions

- event_t * event_new (const char *e, int nkeys,...)
- int event_free (event_t *e)

11.3.1 Detailed Description

Author

Paolo Smiraglia paolo.smiraglia@gmail.com

11.3.2 Typedef Documentation

11.3.2.1 typedef struct __event_t event_t

Structure representing the Event object.

11.3.3 Function Documentation

11.3.3.1 event_t* event_new (const char * e, int nkeys, ...)

This function allocates the memory for a new event_t struct.

Parameters

in	*e*	Event description.
in	*nkeys*	Number of specified keywords.
in	...	Variable list of keywords.

Returns

In case of success the function returns a valid pointer to a event_t struct, otherwise NULL.

11.3.3.2 int event_free (event_t * e)

This function free the memory allocated by the function event_new().

Parameters

in	*e*	Valid pointer to event_t struct.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES.

11.4 /home/paolo/ps/Personal/projects/libseclog/lib/seclog.c File Reference

```
#include "seclog.h"
#include "helper.h"
#include "sk/sk.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <uuid/uuid.h>
```

Functions

- seclog_ctx_t * seclog_new_ctx (void)
- int seclog_init (seclog_ctx_t *ctx, int flags, const char *cert_path, const char *privkey_path, const char *scert_path, const char *ca_cert_path, const char *label, unsigned long max_counter)
- int seclog_log (seclog_ctx_t *ctx, event_t *event, char **json, unsigned int *json_len)
- int seclog_free_ctx (seclog_ctx_t *ctx)

- int seclog_open_step_1 (seclog_ctx_t *ctx, unsigned char **out, unsigned int *out_len, char **logentry, unsigned int *logentry_len)
- int seclog_open_step_2 (seclog_ctx_t *ctx, unsigned char *in, unsigned int in_len, char **logentry, unsigned int *logentry_len)
- int seclog_open (seclog_ctx_t *ctx, unsigned char *in, unsigned int in_len, unsigned char **out, unsigned int *out_len)
- void seclog_err_string (int rv)
- int seclog_close (seclog_ctx_t *ctx, char **json, unsigned int *json_len)
- int seclog_verify (seclog_ctx_t *ctx, char *machine_id, char *session_id, const char *logfile, vrfy_ctx_t **vrfy, const char *dumpfile)
- int seclog_retrieve_sessions (seclog_ctx_t *ctx, char **json, unsigned int *json_len)
- vrfy_ctx_t * vrfy_ctx_new (void)
- int vrfy_ctx_free (vrfy_ctx_t *ctx)

11.4.1 Detailed Description

Author

Paolo Smiraglia paolo.smiraglia@gmail.com

11.4.2 Function Documentation

11.4.2.1 seclog_ctx_t* seclog_new_ctx (void)

This function allocates the memory for a new libseclog context which is represented by the seclog_ctx_t struct.

Returns

In case of success the function returns a valid pointer to a seclog_ctx_t struct, otherwise NULL.

11.4.2.2 int seclog_init (seclog_ctx_t * *ctx*, int *flags*, const char * *cert_path*, const char * *privkey_path*, const char * *scert_path*, const char * *ca_cert_path*, const char * *label*, unsigned long *max_counter*)

This function initialises an already allocated libseclog context.

Parameters

in	*ctx*	Valid pointer to a libseclog context.
in	*flags*	Flags to select the secure logging scheme and the mode. See SCHEME_AND_MODE_FLAGS section.
in	*cert_path*	Certificate file path.
in	*privkey_path*	Private key file path.
in	*scert_path*	Trusted server certificate file path.
in	*ca_cert_path*	Certification authority certificate file path.
in	*label*	Human readable label that may be used to identify the session. Can be NULL.
in	*max_counter*	Maximum number of log entries per session.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES.

11.4.2.3 int seclog_log (seclog_ctx_t * *ctx*, event_t * *event*, char ** *json*, unsigned int * *json_len*)

This function computes a new log entries following the logging scheme specified in the context.

Parameters

in	*ctx*	Valid pointer to a libseclog context.
in	*event*	Valid pointer to an event_t struct. See event section.
out	*json*	A '\0' terminated string that will contain the generated log entry. Before calling this function json must be not allocated and after the usage it must be free by the user.
out	*json_len*	Lenght of the generated log entry.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES.

11.4.2.4 int seclog_free_ctx (seclog_ctx_t * *ctx*)

This function free the memory allocated by the function seclog_new_ctx().

Parameters

in	*ctx*	Valid pointer to a libseclog context.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES.

11.4.2.5 int seclog_open_step_1 (seclog_ctx_t * *ctx*, unsigned char ** *out*, unsigned int * *out_len*, char ** *logentry*, unsigned int * *logentry_len*)

This function must be called by a client application to initialise a new logging session and represents the first step of the initialisation procedure.

Parameters

in	*ctx*	Valid pointer to a libseclog context.
out	*out*	An array of bytes that contains the **initialisation request** message. Such message should be sent to the trusted server.
out	*out_len*	Length of the initialisation request message.
out	*logentry*	A '\0' terminated string that will contains the log entry that track such event.
out	*logentry_len*	Length of the generated log entry.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES.

11.4.2.6 int seclog_open_step_2 (seclog_ctx_t * *ctx*, unsigned char * *in*, unsigned int *in_len*, char ** *logentry*, unsigned int * *logentry_len*)

This function must be called by a client application when the M1 message is received from the trusted server. Such function represents the second and last phase of the logging session initialisation.

Parameters

in	*ctx*	Valid pointer to a libseclog context.
in	*in*	An array of bytes that contains the **initialisation response** message.
in	*in_len*	Length of the initialisation response message.

out	*logentry*	A '\0' terminated string that will contains the log entry that track such event.
out	*logentry_len*	Length of the generated log entry.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES or RETURN_CODES.

11.4.2.7 int seclog_open (seclog_ctx_t * *ctx*, unsigned char * *in*, unsigned int *in_len*, unsigned char ** *out*, unsigned int * *out_len*)

This function manages on server-side the initialisation of a new logging session.

Parameters

in	*ctx*	Valid pointer to a libseclog context.
in	*in*	An array of bytes that contains the **initialisation request** message.
in	*in_len*	Length of the initialisation request message.
out	*out*	An array of bytes that contains the **initialisation response** message.
out	*out_len*	Length of the initialisation response message.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES.

11.4.2.8 void seclog_err_string (int *rv*)

This function print on the standard error a message that describe the error codes in ERROR_CODES.

Parameters

in	*rv*	Error code.

11.4.2.9 int seclog_close (seclog_ctx_t * *ctx*, char ** *json*, unsigned int * *json_len*)

This function closes an already opened logging session.

Parameters

in	*ctx*	Valid pointer to a libseclog context.
out	*json*	A '\0' terminated string that will contains the log entry that track such event.
out	*json_len*	Length of the generated log entry.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES.

11.4.2.10 int seclog_verify (seclog_ctx_t * *ctx*, char * *machine_id*, char * *session_id*, const char * *logfile*, vrfy_ctx_t ** *vrfy*, const char * *dumpfile*)

This function is called by the trusted server to manage the logging session verification requests.

Parameters

in	*ctx*	Valid pointer to a libseclog context.
in	*machine_id*	ID of the client that initialised the logging session.
in	*session_id*	ID of the logging session.
in	*logfile*	Path of the file that contains the log entries of the session.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES or LOG_VERIFICATION_FAILURE.

11.4.2.11 int seclog_retrieve_sessions (seclog_ctx_t * *ctx*, char ** *json*, unsigned int * *json_len*)

This function returns a string that contains a JSON structure where are reported all the opened logging sessions.

Parameters

in	*ctx*	Valid pointer to a libseclog context.
out	*json*	A '\0' terminated string that contains the JSON structure.
out	*json_len*	Length of the JSON string.

11.4.2.12 vrfy_ctx_t* vrfy_ctx_new (void)

This function allocates the memory for a new verification context.

11.4.2.13 int vrfy_ctx_free (vrfy_ctx_t * *ctx*)

This function frees the allocated memory for a verification context.

11.5 /home/paolo/ps/Personal/projects/libseclog/lib/seclog.h File Reference

```
#include "common.h"
#include "event.h"
#include "mt/mt.h"
#include "sk/sk.h"
```

Data Structures

- struct __seclog_ctx_t
- struct __vrfy_ctx_t

Functions

- seclog_ctx_t * seclog_new_ctx (void)
- int seclog_init (seclog_ctx_t *ctx, int flags, const char *cert_path, const char *privkey_path, const char *scert_path, const char *ca_cert_path, const char *label, unsigned long max_counter)
- int seclog_log (seclog_ctx_t *ctx, event_t *event, char **json, unsigned int *json_len)
- int seclog_free_ctx (seclog_ctx_t *ctx)
- int seclog_open_step_1 (seclog_ctx_t *ctx, unsigned char **out, unsigned int *out_len, char **logentry, unsigned int *logentry_len)

- int seclog_open_step_2 (seclog_ctx_t *ctx, unsigned char *in, unsigned int in_len, char **logentry, unsigned int *logentry_len)
- int seclog_open (seclog_ctx_t *ctx, unsigned char *in, unsigned int in_len, unsigned char **out, unsigned int *out_len)
- void seclog_err_string (int rv)
- int seclog_close (seclog_ctx_t *ctx, char **json, unsigned int *json_len)
- int seclog_verify (seclog_ctx_t *ctx, char *machine_id, char *session_id, const char *logfile, vrfy_ctx_t **vrfy, const char *dumpfile)
- int seclog_retrieve_sessions (seclog_ctx_t *ctx, char **json, unsigned int *json_len)
- vrfy_ctx_t * vrfy_ctx_new (void)
- int vrfy_ctx_free (vrfy_ctx_t *ctx)

11.5.1 Detailed Description

Author

Paolo Smiraglia paolo.smiraglia@gmail.com

11.5.2 Function Documentation

11.5.2.1 seclog_ctx_t* seclog_new_ctx (void)

This function allocates the memory for a new libseclog context which is represented by the seclog_ctx_t struct.

Returns

In case of success the function returns a valid pointer to a seclog_ctx_t struct, otherwise NULL.

11.5.2.2 int seclog_init (seclog_ctx_t * *ctx*, int *flags*, const char * *cert_path*, const char * *privkey_path*, const char * *scert_path*, const char * *ca_cert_path*, const char * *label*, unsigned long *max_counter*)

This function initialises an already allocated libseclog context.

Parameters

in	*ctx*	Valid pointer to a libseclog context.
in	*flags*	Flags to select the secure logging scheme and the mode. See SCHEME_AND_MODE_FLAGS section.
in	*cert_path*	Certificate file path.
in	*privkey_path*	Private key file path.
in	*scert_path*	Trusted server certificate file path.
in	*ca_cert_path*	Certification authority certificate file path.
in	*label*	Human readable label that may be used to identify the session. Can be NULL.
in	*max_counter*	Maximum number of log entries per session.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES.

11.5.2.3 int seclog_log (seclog_ctx_t * *ctx*, event_t * *event*, char ** *json*, unsigned int * *json_len*)

This function computes a new log entries following the logging scheme specified in the context.

Parameters

in	*ctx*	Valid pointer to a libseclog context.
in	*event*	Valid pointer to an event_t struct. See event section.
out	*json*	A '\0' terminated string that will contain the generated log entry. Before calling this function json must be not allocated and after the usage it must be free by the user.
out	*json_len*	Lenght of the generated log entry.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES.

11.5.2.4 int seclog_free_ctx (seclog_ctx_t * *ctx*)

This function free the memory allocated by the function seclog_new_ctx().

Parameters

in	*ctx*	Valid pointer to a libseclog context.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES.

11.5.2.5 int seclog_open_step_1 (seclog_ctx_t * *ctx*, unsigned char ** *out*, unsigned int * *out_len*, char ** *logentry*, unsigned int * *logentry_len*)

This function must be called by a client application to initialise a new logging session and represents the first step of the initialisation procedure.

Parameters

in	*ctx*	Valid pointer to a libseclog context.
out	*out*	An array of bytes that contains the **initialisation request** message. Such message should be sent to the trusted server.
out	*out_len*	Length of the initialisation request message.
out	*logentry*	A '\0' terminated string that will contains the log entry that track such event.
out	*logentry_len*	Length of the generated log entry.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES.

11.5.2.6 int seclog_open_step_2 (seclog_ctx_t * *ctx*, unsigned char * *in*, unsigned int *in_len*, char ** *logentry*, unsigned int * *logentry_len*)

This function must be called by a client application when the M1 message is received from the trusted server. Such function represents the second and last phase of the logging session initialisation.

Parameters

in	*ctx*	Valid pointer to a libseclog context.
in	*in*	An array of bytes that contains the **initialisation response** message.
in	*in_len*	Length of the initialisation response message.

out	*logentry*	A '\0' terminated string that will contains the log entry that track such event.
out	*logentry_len*	Length of the generated log entry.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES or RETURN_CODES.

11.5.2.7 int seclog_open (seclog_ctx_t * *ctx*, unsigned char * *in*, unsigned int *in_len*, unsigned char ** *out*, unsigned int * *out_len*)

This function manages on server-side the initialisation of a new logging session.

Parameters

in	*ctx*	Valid pointer to a libseclog context.
in	*in*	An array of bytes that contains the **initialisation request** message.
in	*in_len*	Length of the initialisation request message.
out	*out*	An array of bytes that contains the **initialisation response** message.
out	*out_len*	Length of the initialisation response message.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES.

11.5.2.8 void seclog_err_string (int *rv*)

This function print on the standard error a message that describe the error codes in ERROR_CODES.

Parameters

in	*rv*	Error code.

11.5.2.9 int seclog_close (seclog_ctx_t * *ctx*, char ** *json*, unsigned int * *json_len*)

This function closes an already opened logging session.

Parameters

in	*ctx*	Valid pointer to a libseclog context.
out	*json*	A '\0' terminated string that will contains the log entry that track such event.
out	*json_len*	Length of the generated log entry.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES.

11.5.2.10 int seclog_verify (seclog_ctx_t * *ctx*, char * *machine_id*, char * *session_id*, const char * *logfile*, vrfy_ctx_t ** *vrfy*, const char * *dumpfile*)

This function is called by the trusted server to manage the logging session verification requests.

Parameters

in	*ctx*	Valid pointer to a libseclog context.
in	*machine_id*	ID of the client that initialised the logging session.
in	*session_id*	ID of the logging session.
in	*logfile*	Path of the file that contains the log entries of the session.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES or LOG_VERIFICATION_FAILURE.

11.5.2.11 int seclog_retrieve_sessions (seclog_ctx_t * *ctx*, char ** *json*, unsigned int * *json_len*)

This function returns a string that contains a JSON structure where are reported all the opened logging sessions.

Parameters

in	*ctx*	Valid pointer to a libseclog context.
out	*json*	A '\0' terminated string that contains the JSON structure.
out	*json_len*	Length of the JSON string.

11.5.2.12 vrfy_ctx_t* vrfy_ctx_new (void)

This function allocates the memory for a new verification context.

11.5.2.13 int vrfy_ctx_free (vrfy_ctx_t * *ctx*)

This function frees the allocated memory for a verification context.

11.6 /home/paolo/ps/Personal/projects/libseclog/lib/sk/sk.c File Reference

```
#include "sk.h"
#include "../helper.h"
#include "../seclog.h"
#include "internal.h"
#include <jansson.h>
#include <string.h>
#include <openssl/err.h>
#include <openssl/rand.h>
```

Functions

- sk_ctx_t * sk_new_ctx (void)
- sk_ctx_t * sk_new_vrfy_ctx (unsigned char *in, unsigned int len)
- int sk_init_ctx (sk_ctx_t *ctx, int mode, unsigned int timeout)
- int sk_log (sk_ctx_t *ctx, event_t *e, char **json, unsigned int *json_len)
- int sk_free_ctx (sk_ctx_t *ctx)
- int sk_open_step_1 (sk_ctx_t *ctx, unsigned char *in, unsigned int in_len, unsigned char **out1, unsigned int *out1_len, char **out2, unsigned int *out2_len)
- int sk_open_step_2 (sk_ctx_t *ctx, unsigned char *in, unsigned int in_len, char **out, unsigned int *out_len)

- int sk_close (sk_ctx_t *ctx, char **out, unsigned int *out_len)
- int sk_verify (sk_ctx_t *ctx, sk_vrfy_ctx_t *vrfy, FILE *fp, const char *dumpfile)
- int sk_retrieve_sessions (sk_ctx_t *ctx, char **json, unsigned int *json_len)
- sk_vrfy_ctx_t * sk_vrfy_ctx_new (void)
- int sk_vrfy_ctx_free (sk_vrfy_ctx_t *ctx)

11.6.1 Detailed Description

Author

Paolo Smiraglia paolo.smiraglia@gmail.com

11.6.2 Function Documentation

11.6.2.1 sk_ctx_t* sk_new_ctx (void)

Allocates a new SK context.

Returns

In case of success the function returns a valid pointer to a sk_ctx_t struct, otherwise NULL.

11.6.2.2 sk_ctx_t* sk_new_vrfy_ctx (unsigned char * *in*, unsigned int *len*)

Allocates a new SK verification context.

Returns

In case of success the function returns a valid pointer to a sk_ctx_t struct, otherwise NULL.

11.6.2.3 int sk_init_ctx (sk_ctx_t * *ctx*, int *mode*, unsigned int *timeout*)

Initialize an SK context

Parameters

in	*ctx*	Valid pointer to a SK context.
in	*mode*	Flags to select the secure logging scheme and the mode. See SCHEME_AND_MODE_FLAGS section.
in	*timeout*	Set the timeout expressed in seconds for the initialisation procedure.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES.

11.6.2.4 int sk_log (sk_ctx_t * *ctx*, event_t * *e*, char ** *json*, unsigned int * *json_len*)

Generates a secure log entry following the SK scheme.

Parameters

in	*ctx*	Valid pointer to a SK context.
in	*e*	Valid pointer to a event_t struct.

out	*json*	A '\0' terminated string that will contain the generated log entry. Before calling this function json must be not allocated and after the usage it must be free by the user.
out	*json_len*	Length of json.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES.

11.6.2.5 int sk_free_ctx (sk_ctx_t * *ctx*)

Releases the resources allocated for a SK context.

Parameters

in	*ctx*	Valid pointer to a SK context.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES.

11.6.2.6 int sk_open_step_1 (sk_ctx_t * *ctx*, unsigned char * *in*, unsigned int *in_len*, unsigned char ** *out1*, unsigned int * *out1_len*, char ** *out2*, unsigned int * *out2_len*)

Performs the first step of the initialisation process. The function behaviour changes if the caller application plays the role of T or U.

Parameters

in	*ctx*	Valid pointer to a SK context.
in	*in*	Array containing the M0 message. If the caller plays as U such parametr is ignored.
in	*in_len*	Length of the in array.
in	*out1*	If the caller is U this contains the M0 message. If the caller is T, this contains the M1 message.
in	*out1_len*	Length of the out1 array.
in	*out2*	If the caller is U this contains the log entry tracking the initialisation of the session. If the caller is T, this is ignored.
in	*out2_len*	Length of the out1 array.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES.

11.6.2.7 int sk_open_step_2 (sk_ctx_t * *ctx*, unsigned char * *in*, unsigned int *in_len*, char ** *out*, unsigned int * *out_len*)

Performs the second step of the initialisation process. The function behaviour changes if the caller application plays the role of T or U.

Parameters

in	*ctx*	Valid pointer to a SK context.
in	*in*	Array containing the M1 message. If the caller plays as T such parametr is ignored.

in	*in_len*	Length of the in array.
in	*out*	If the caller is U this contains the log entry tracking the initialisation of the session. If the caller is T, this is ignored.
in	*out_len*	Length of the out array.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES.

11.6.2.8 int sk_close (sk_ctx_t * *ctx*, char ** *out*, unsigned int * *out_len*)

Performs the closure of a logging session following the SK scheme.

Parameters

in	*ctx*	Valid pointer to a SK context.
in	*out*	A '\0' terminated string containing the log entry representing the closure of the session.
in	*out_len*	Length of the out array.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES.

11.6.2.9 int sk_verify (sk_ctx_t * *ctx*, sk_vrfy_ctx_t * *vrfy*, FILE * *fp*, const char * *dumpfile*)

Executes the verification of a logging session following the SK scheme.

Parameters

in	*ctx*	Valid pointer to a SK context.
in	*vrfy*	Valid pointer to a SK verification context.
in	*fp*	File pointer to a file containing the log entries that will be verified.
in	*dumpfile*	Path of the file where the dump will be generated.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES.

11.6.2.10 int sk_retrieve_sessions (sk_ctx_t * *ctx*, char ** *json*, unsigned int * *json_len*)

Retrieves the list containing the logging session already initialised.

Parameters

in	*ctx*	Valid pointer to a SK context.
out	*json*	Char string representing a JSON message containing the list of the initialised sessions.
out	*json_len*	Length of the json string.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES.

11.6.2.11 sk_vrfy_ctx_t* sk_vrfy_ctx_new (void)

Allocates a new SK verification context.

Returns

In case of success the function returns a valid pointer to a ::sk_vrf_ctx_t struct, otherwise NULL.

11.6.2.12 int sk_vrfy_ctx_free (sk_vrfy_ctx_t * *ctx*)

Releases the resources allocated for a SK verification context.

Parameters

in	*vrfy*	Valid pointer to a SK verification context.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES.

11.7 /home/paolo/ps/Personal/projects/libseclog/lib/sk/sk.h File Reference

```
#include "../event.h"
#include "../common.h"
#include <stdio.h>
#include <sys/time.h>
```

Data Structures

- struct __sk_ctx_t
- struct __sk_vrfy_ctx_t

Typedefs

- typedef struct __sk_ctx_t sk_ctx_t
- typedef struct __sk_vrfy_ctx_t sk_vrfy_ctx_t

Functions

- sk_ctx_t * sk_new_ctx (void)
- int sk_init_ctx (sk_ctx_t *ctx, int mode, unsigned int timeout)
- int sk_log (sk_ctx_t *ctx, event_t *e, char **json, unsigned int *json_len)
- int sk_free_ctx (sk_ctx_t *ctx)
- int sk_open_step_1 (sk_ctx_t *ctx, unsigned char *in, unsigned int in_len, unsigned char **out1, unsigned int *out1_len, char **out2, unsigned int *out2_len)
- int sk_open_step_2 (sk_ctx_t *ctx, unsigned char *in, unsigned int in_len, char **out, unsigned int *out_len)
- int sk_close (sk_ctx_t *ctx, char **out, unsigned int *out_len)

- int sk_verify (sk_ctx_t *ctx, sk_vrfy_ctx_t *vrfy, FILE *fp, const char *dumpfile)
- int sk_retrieve_sessions (sk_ctx_t *ctx, char **json, unsigned int *json_len)
- sk_vrfy_ctx_t * sk_vrfy_ctx_new (void)
- int sk_vrfy_ctx_free (sk_vrfy_ctx_t *ctx)

11.7.1 Detailed Description

Author

Paolo Smiraglia paolo.smiraglia@gmail.com

11.7.2 Typedef Documentation

11.7.2.1 typedef struct __sk_ctx_t sk_ctx_t

Schneier and Kelsey context structure

11.7.2.2 typedef struct __sk_vrfy_ctx_t sk_vrfy_ctx_t

Schneier and Kelsey verification context structure

11.7.3 Function Documentation

11.7.3.1 sk_ctx_t* sk_new_ctx (void)

Allocates a new SK context.

Returns

In case of success the function returns a valid pointer to a sk_ctx_t struct, otherwise NULL.

11.7.3.2 int sk_init_ctx (sk_ctx_t * *ctx*, int *mode*, unsigned int *timeout*)

Initialize an SK context

Parameters

in	*ctx*	Valid pointer to a SK context.
in	*mode*	Flags to select the secure logging scheme and the mode. See SCHEME_AND_MODE_FLAGS section.
in	*timeout*	Set the timeout expressed in seconds for the initialisation procedure.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES.

11.7.3.3 int sk_log (sk_ctx_t * *ctx*, event_t * *e*, char ** *json*, unsigned int * *json_len*)

Generates a secure log entry following the SK scheme.

Parameters

in	*ctx*	Valid pointer to a SK context.
in	*e*	Valid pointer to a event_t struct.
out	*json*	A '\0' terminated string that will contain the generated log entry. Before calling this function json must be not allocated and after the usage it must be free by the user.
out	*json_len*	Length of json.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES.

11.7.3.4 int sk_free_ctx (sk_ctx_t * *ctx*)

Releases the resources allocated for a SK context.

Parameters

in	*ctx*	Valid pointer to a SK context.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES.

11.7.3.5 int sk_open_step_1 (sk_ctx_t * *ctx*, unsigned char * *in*, unsigned int *in_len*, unsigned char ** *out1*, unsigned int * *out1_len*, char ** *out2*, unsigned int * *out2_len*)

Performs the first step of the initialisation process. The function behaviour changes if the caller application plays the role of T or U.

Parameters

in	*ctx*	Valid pointer to a SK context.
in	*in*	Array containing the M0 message. If the caller plays as U such parametr is ignored.
in	*in_len*	Length of the in array.
in	*out1*	If the caller is U this contains the M0 message. If the caller is T, this contains the M1 message.
in	*out1_len*	Length of the out1 array.
in	*out2*	If the caller is U this contains the log entry tracking the initialisation of the session. If the caller is T, this is ignored.
in	*out2_len*	Length of the out1 array.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES.

11.7.3.6 int sk_open_step_2 (sk_ctx_t * *ctx*, unsigned char * *in*, unsigned int *in_len*, char ** *out*, unsigned int * *out_len*)

Performs the second step of the initialisation process. The function behaviour changes if the caller application plays the role of T or U.

Parameters

in	*ctx*	Valid pointer to a SK context.
in	*in*	Array containing the M1 message. If the caller plays as T such parametr is ignored.
in	*in_len*	Length of the in array.
in	*out*	If the caller is U this contains the log entry tracking the initialisation of the session. If the caller is T, this is ignored.
in	*out_len*	Length of the out array.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES.

11.7.3.7 int sk_close (sk_ctx_t * *ctx*, char ** *out*, unsigned int * *out_len*)

Performs the closure of a logging session following the SK scheme.

Parameters

in	*ctx*	Valid pointer to a SK context.
in	*out*	A '\0' terminated string containing the log entry representing the closure of the session.
in	*out_len*	Length of the out array.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES.

11.7.3.8 int sk_verify (sk_ctx_t * *ctx*, sk_vrfy_ctx_t * *vrfy*, FILE * *fp*, const char * *dumpfile*)

Executes the verification of a logging session following the SK scheme.

Parameters

in	*ctx*	Valid pointer to a SK context.
in	*vrfy*	Valid pointer to a SK verification context.
in	*fp*	File pointer to a file containing the log entries that will be verified.
in	*dumpfile*	Path of the file where the dump will be generated.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES.

11.7.3.9 int sk_retrieve_sessions (sk_ctx_t * *ctx*, char ** *json*, unsigned int * *json_len*)

Retrieves the list containing the logging session already initialised.

Parameters

in	*ctx*	Valid pointer to a SK context.
out	*json*	Char string representing a JSON message containing the list of the initialised sessions.
out	*json_len*	Length of the json string.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES.

11.7.3.10 sk_vrfy_ctx_t* sk_vrfy_ctx_new (void)

Allocates a new SK verification context.

Returns

In case of success the function returns a valid pointer to a ::sk_vrf_ctx_t struct, otherwise NULL.

11.7.3.11 int sk_vrfy_ctx_free (sk_vrfy_ctx_t * *ctx*)

Releases the resources allocated for a SK verification context.

Parameters

in	*vrfy*	Valid pointer to a SK verification context.

Returns

The function returns SECLOG_SUCCESS in case of success, otherwise one of the values in ERROR_CODES.

Index