

AMCL

Generated by Doxygen 1.8.11

Contents

1	Apache Milagro Crypto Library (AMCL)	1
1.1	Project page	1
1.2	License	1
1.3	Platforms	2
1.4	Downloads	2
1.5	Installation	2
2	Linux	3
3	Mac OS	5
4	Windows	7
5	File Index	9
5.1	File List	9
6	File Documentation	11
6.1	ecdh.h File Reference	11
6.1.1	Detailed Description	12
6.1.2	Macro Definition Documentation	12
6.1.2.1	EAS	12
6.1.2.2	ECDH_ERROR	12
6.1.2.3	ECDH_INVALID	12
6.1.2.4	ECDH_INVALID_PUBLIC_KEY	12
6.1.2.5	ECDH_OK	13
6.1.2.6	EFS	13

6.1.2.7	EGS	13
6.1.3	Function Documentation	13
6.1.3.1	ECP_AES_CBC_IV0_DECRYPT(octet *K, octet *C, octet *P)	13
6.1.3.2	ECP_AES_CBC_IV0_ENCRYPT(octet *K, octet *P, octet *C)	13
6.1.3.3	ECP_CREATE_CSPRNG(csprng *R, octet *S)	14
6.1.3.4	ECP_ECIES_DECRYPT(octet *P1, octet *P2, octet *V, octet *C, octet *T, octet *U, octet *M)	14
6.1.3.5	ECP_ECIES_ENCRYPT(octet *P1, octet *P2, csprng *R, octet *W, octet *M, int len, octet *V, octet *C, octet *T)	14
6.1.3.6	ECP_HASH(octet *I, octet *O)	15
6.1.3.7	ECP_HMAC(octet *M, octet *K, int len, octet *tag)	15
6.1.3.8	ECP_KDF2(octet *Z, octet *P, int len, octet *K)	15
6.1.3.9	ECP_KEY_PAIR_GENERATE(csprng *R, octet *s, octet *W)	15
6.1.3.10	ECP_KILL_CSPRNG(csprng *R)	16
6.1.3.11	ECP_PBKDF2(octet *P, octet *S, int rep, int len, octet *K)	16
6.1.3.12	ECP_PUBLIC_KEY_VALIDATE(int f, octet *W)	16
6.1.3.13	ECP_SP_DSA(csprng *R, octet *s, octet *M, octet *c, octet *d)	17
6.1.3.14	ECP_SVDP_DH(octet *s, octet *W, octet *K)	17
6.1.3.15	ECP_VP_DSA(octet *W, octet *M, octet *c, octet *d)	17
6.2	mpin.h File Reference	18
6.2.1	Detailed Description	20
6.2.2	Macro Definition Documentation	20
6.2.2.1	HASH_BYTES	20
6.2.2.2	MAXPIN	20
6.2.2.3	MPIN_BAD_PIN	20
6.2.2.4	MPIN_INVALID_POINT	20
6.2.2.5	MPIN_OK	20
6.2.2.6	PAS	20
6.2.2.7	PBLLEN	20
6.2.2.8	PFS	21
6.2.2.9	PGS	21

6.2.2.10	TIME_SLOT_MINUTES	21
6.2.3	Function Documentation	21
6.2.3.1	MPIN_AES_GCM_DECRYPT(octet *K, octet *IV, octet *H, octet *C, octet *P, octet *T)	21
6.2.3.2	MPIN_AES_GCM_ENCRYPT(octet *K, octet *IV, octet *H, octet *P, octet *C, octet *T)	21
6.2.3.3	MPIN_CLIENT(int d, octet *ID, csprng *R, octet *x, int pin, octet *T, octet *V, octet *U, octet *UT, octet *TP, octet *MESSAGE, int t, octet *y)	21
6.2.3.4	MPIN_CLIENT_1(int d, octet *ID, csprng *R, octet *x, int pin, octet *T, octet *S, octet *U, octet *UT, octet *TP)	22
6.2.3.5	MPIN_CLIENT_2(octet *x, octet *y, octet *V)	23
6.2.3.6	MPIN_CLIENT_KEY(octet *g1, octet *g2, int pin, octet *r, octet *x, octet *p, octet *T, octet *K)	23
6.2.3.7	MPIN_CREATE_CSPRNG(csprng *R, octet *S)	23
6.2.3.8	MPIN_DECODING(octet *TP)	24
6.2.3.9	MPIN_ENCODING(csprng *R, octet *TP)	24
6.2.3.10	MPIN_EXTRACT_PIN(octet *ID, int pin, octet *CS)	24
6.2.3.11	MPIN_GET_CLIENT_PERMIT(int d, octet *S, octet *ID, octet *TP)	24
6.2.3.12	MPIN_GET_CLIENT_SECRET(octet *S, octet *ID, octet *CS)	25
6.2.3.13	MPIN_GET_G1_MULTIPLE(csprng *R, int type, octet *x, octet *G, octet *W)	25
6.2.3.14	MPIN_GET_SERVER_SECRET(octet *S, octet *SS)	25
6.2.3.15	MPIN_GET_TIME(void)	26
6.2.3.16	MPIN_GET_Y(int t, octet *O, octet *Y)	26
6.2.3.17	MPIN_HASH_ALL(octet *I, octet *U, octet *CU, octet *V, octet *Y, octet *R, octet *W, octet *H)	26
6.2.3.18	MPIN_HASH_ID(octet *ID, octet *HID)	27
6.2.3.19	MPIN_HMAC(octet *M, octet *K, int len, octet *tag)	27
6.2.3.20	MPIN_KANGAROO(octet *E, octet *F)	27
6.2.3.21	MPIN_KILL_CSPRNG(csprng *R)	27
6.2.3.22	MPIN_PBKDF2(octet *P, octet *S, int rep, int len, octet *K)	28
6.2.3.23	MPIN_PRECOMPUTE(octet *T, octet *ID, octet *g1, octet *g2)	28
6.2.3.24	MPIN_RANDOM_GENERATE(csprng *R, octet *S)	28
6.2.3.25	MPIN_RECOMBINE_G1(octet *Q1, octet *Q2, octet *Q)	29

6.2.3.26	MPIN_RECOMBINE_G2(octet *P1, octet *P2, octet *P)	29
6.2.3.27	MPIN_SERVER(int d, octet *HID, octet *HTID, octet *y, octet *SS, octet *U, octet *UT, octet *V, octet *E, octet *F, octet *ID, octet *MESSAGE, int t)	29
6.2.3.28	MPIN_SERVER_1(int d, octet *ID, octet *HID, octet *HTID)	30
6.2.3.29	MPIN_SERVER_2(int d, octet *HID, octet *HTID, octet *y, octet *SS, octet *U, octet *UT, octet *V, octet *E, octet *F)	30
6.2.3.30	MPIN_SERVER_KEY(octet *Z, octet *SS, octet *w, octet *p, octet *l, octet *U, octet *UT, octet *K)	31
6.2.3.31	MPIN_today(void)	31
6.3	rsa.h File Reference	31
6.3.1	Detailed Description	32
6.3.2	Macro Definition Documentation	32
6.3.2.1	RFS	32
6.3.3	Function Documentation	32
6.3.3.1	RSA_CREATE_CSPRNG(csprng *R, octet *S)	32
6.3.3.2	RSA_DECRYPT(rsa_private_key *PRIV, octet *G, octet *F)	33
6.3.3.3	RSA_ENCRYPT(rsa_public_key *PUB, octet *F, octet *G)	33
6.3.3.4	RSA_KEY_PAIR(csprng *R, sign32 e, rsa_private_key *PRIV, rsa_public_key *PUB)	33
6.3.3.5	RSA_KILL_CSPRNG(csprng *R)	33
6.3.3.6	RSA_OAEP_DECODE(octet *P, octet *F)	34
6.3.3.7	RSA_OAEP_ENCODE(octet *M, csprng *R, octet *P, octet *F)	34
6.3.3.8	RSA_PRIVATE_KEY_KILL(rsa_private_key *PRIV)	34
6.4	wcc.c File Reference	35
6.4.1	Detailed Description	36
6.4.2	Function Documentation	36
6.4.2.1	WCC_AES_GCM_DECRYPT(octet *K, octet *IV, octet *H, octet *C, octet *P, octet *T)	36
6.4.2.2	WCC_AES_GCM_ENCRYPT(octet *K, octet *IV, octet *H, octet *P, octet *C, octet *T)	36
6.4.2.3	WCC_CREATE_CSPRNG(csprng *RNG, octet *SEED)	37
6.4.2.4	WCC_GET_G1_MULTIPLE(int hashDone, octet *S, octet *ID, octet *VG1)	37
6.4.2.5	WCC_GET_G1_PERMIT(int date, octet *S, octet *HID, octet *TPG1)	37

6.4.2.6	WCC_GET_G1_TPMULT(int date, octet *S, octet *ID, octet *VG1)	38
6.4.2.7	WCC_GET_G2_MULTIPLE(int hashDone, octet *S, octet *ID, octet *VG2)	38
6.4.2.8	WCC_GET_G2_PERMIT(int date, octet *S, octet *HID, octet *TPG2)	38
6.4.2.9	WCC_GET_G2_TPMULT(int date, octet *S, octet *ID, octet *VG2)	39
6.4.2.10	WCC_HASH_ID(octet *ID, octet *HID)	39
6.4.2.11	WCC_Hq(octet *A, octet *B, octet *C, octet *D, octet *h)	40
6.4.2.12	WCC_KILL_CSPRNG(csprng *RNG)	40
6.4.2.13	WCC_RANDOM_GENERATE(csprng *RNG, octet *S)	40
6.4.2.14	WCC_RECEIVER_KEY(int date, octet *yOct, octet *wOct, octet *piaOct, octet *pibOct, octet *PaG1Oct, octet *PgG1Oct, octet *BKeyG2Oct, octet *BTPG2Oct, octet *IdAOct, octet *AESKeyOct)	41
6.4.2.15	WCC_RECOMBINE_G1(octet *R1, octet *R2, octet *R)	41
6.4.2.16	WCC_RECOMBINE_G2(octet *W1, octet *W2, octet *W)	42
6.4.2.17	WCC_SENDER_KEY(int date, octet *xOct, octet *piaOct, octet *pibOct, octet *PbG2Oct, octet *PgG1Oct, octet *AKeyG1Oct, octet *ATPG1Oct, octet *IdBOct, octet *AESKeyOct)	42
6.4.2.18	WCC_today(void)	42

Chapter 1

Apache Milagro Crypto Library (AMCL)

Read [AMCL.pdf](#) for an introduction to AMCL

AMCL is provided in these languages;

- C
- JAVA
- JavaScript
- C#
- Swift
- GO

There is also a Python wrapper provided that requires [CFFI](#)

1.1 Project page

The official project page is hosted at [MIRACL Products](#)

1.2 License

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to you under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

1.3 Platforms

The software can be compiled and installed for these operating systems;

- Linux
- Windows
- Mac OS

1.4 Downloads

The source code is available from the GIT repository:

git clone <https://github.com/miracl/milagro-crypto>

1.5 Installation

There are instructions for building for [Linux](#), [Mac OS](#) and [Windows](#).

Chapter 2

Linux

Software dependencies

CMake is required to build the library and can usually be installed from the operating system package manager.

- `sudo apt-get install cmake`

If not, then you can download it from www.cmake.org

The C Foreign Function Interface for Python [CFFI](#) module is also required if you wish to use the Python module.

- `sudo pip install cffi`

In order to build the documentation [doxygen](#) is required.

Build Instructions

The default build is for 32 bit machines

1. `mkdir Release`
2. `cd Release`
3. `cmake ..`
4. `make`
5. `make test`
6. `make doc`
7. `sudo make install`

The build can be configured using by setting flags on the command line i.e.

1. `cmake -D CMAKE_INSTALL_PREFIX=/opt/amcl -D WORD_LENGTH=64 ..`

Uninstall software

- `sudo make uninstall`

Building an installer

After having built the libraries you can build a binary installer and a source distribution by running this command

- `make package`

Chapter 3

Mac OS

Software dependencies

Install **Homebrew**

- `ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"`

Install **cmake**

- `brew install cmake`

The C Foreign Function Interface for Python **CFFI** module is also required if you wish to use the Python module.

- `brew install pkg-config libffi`
- `sudo pip install cffi`

In order to build the documentation **doxygen** is required.

- `brew install doxygen`

Build Instructions

The default build is for 32 bit machines

1. `mkdir Release`
2. `cd Release`
3. `cmake ..`
4. `make`
5. `make test`
6. `make doc`
7. `sudo make install`

The build can be configured using by setting flags on the command line i.e.

1. `cmake -DWORD_LENGTH=64 ..`

Uninstall software

- `sudo make uninstall`

Chapter 4

Windows

Software dependencies

Minimalist GNU for Windows **MinGW** provides the tool set used to build the library and should be installed. When the MinGW installer starts select the mingw32-base and mingw32-gcc-g++ components. From the menu select "Installation" -> "Apply Changes", then click "Apply". Finally add C:\MinGW\bin to the PATH variable.

CMake is required to build the library and can be downloaded from www.cmake.org

The C Foreign Function Interface for Python **CFFI** module is also required, if you wish to use the Python module.

- pip install cffi

In order to build the documentation **doxygen** is required.

Build Instructions

Start a command prompt as an administrator

The default build is for 32 bit machines

1. mkdir Release
2. cd Release
3. cmake -G "MinGW Makefiles" ..
4. mingw32-make
5. mingw32-make test
6. mingw32-make doc
7. mingw32-make install

Post install append the PATH system variable to point to the install ./lib.

My Computer -> Properties -> Advanced > Environment Variables

The build can be configured using by setting flags on the command line i.e.

1. cmake -G "MinGW Makefiles" -DWORD_LENGTH=64 ..

Uninstall software

- `mingw32-make uninstall`

Building an installer

After having built the libraries you can build a Windows installer using this command

- `sudo mingw32-make package`

In order for this to work `NSSI` has to have been installed

Chapter 5

File Index

5.1 File List

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

ecdh.h	ECDH Header file for implementation of standard EC protocols	11
mpin.h	M-Pin Header file	18
rsa.h	RSA Header file for implementation of RSA protocol	31
wcc.c	Wang / Chow Choo (WCC) definitions	35

Chapter 6

File Documentation

6.1 ecdh.h File Reference

ECDH Header file for implementation of standard EC protocols.

```
#include "amcl.h"
```

Macros

- #define [EAS](#) 16
- #define [EGS](#) 32
- #define [EFS](#) 32
- #define [ECDH_OK](#) 0
- #define [ECDH_INVALID_PUBLIC_KEY](#) -2
- #define [ECDH_ERROR](#) -3
- #define [ECDH_INVALID](#) -4

Functions

- void [ECP_CREATE_CSPRNG](#) (csprng *R, octet *S)
Initialise a random number generator.
- void [ECP_KILL_CSPRNG](#) (csprng *R)
Kill a random number generator.
- void [ECP_HASH](#) (octet *I, octet *O)
hash an octet into another octet
- int [ECP_HMAC](#) (octet *M, octet *K, int len, octet *tag)
HMAC of message M using key K to create tag of length len in octet tag.
- void [ECP_KDF2](#) (octet *Z, octet *P, int len, octet *K)
Key Derivation Function - generates key K from inputs Z and P.
- void [ECP_PBKDF2](#) (octet *P, octet *S, int rep, int len, octet *K)
Password Based Key Derivation Function - generates key K from password, salt and repeat counter.
- void [ECP_AES_CBC_IV0_ENCRYPT](#) (octet *K, octet *P, octet *C)
AES encrypts a plaintext to a ciphertext.
- int [ECP_AES_CBC_IV0_DECRYPT](#) (octet *K, octet *C, octet *P)

- AES encrypts a plaintext to a ciphertext.*

 - int [ECP_KEY_PAIR_GENERATE](#) (csprng *R, octet *s, octet *W)

Generate an ECC public/private key pair.
- int [ECP_PUBLIC_KEY_VALIDATE](#) (int f, octet *W)

Validate an ECC public key.
- int [ECP_SVDP_DH](#) (octet *s, octet *W, octet *K)

Generate Diffie-Hellman shared key.
- void [ECP_ECIES_ENCRYPT](#) (octet *P1, octet *P2, csprng *R, octet *W, octet *M, int len, octet *V, octet *C, octet *T)

ECIES Encryption.
- int [ECP_ECIES_DECRYPT](#) (octet *P1, octet *P2, octet *V, octet *C, octet *T, octet *U, octet *M)

ECIES Decryption.
- int [ECP_SP_DSA](#) (csprng *R, octet *s, octet *M, octet *c, octet *d)

ECDSA Signature.
- int [ECP_VP_DSA](#) (octet *W, octet *M, octet *c, octet *d)

ECDSA Signature Verification.

6.1.1 Detailed Description

ECDH Header file for implementation of standard EC protocols.

Author

Mike Scott and Kealan McCusker

Date

2nd June 2015 declares functions

6.1.2 Macro Definition Documentation

6.1.2.1 #define EAS 16

Symmetric Key size - 128 bits

6.1.2.2 #define ECDH_ERROR -3

ECDH Internal Error

6.1.2.3 #define ECDH_INVALID -4

ECDH Internal Error

6.1.2.4 #define ECDH_INVALID_PUBLIC_KEY -2

Public Key is Invalid

6.1.2.5 #define ECDH_OK 0

Function completed without error

6.1.2.6 #define EFS 32

ECC Field Size

6.1.2.7 #define EGS 32

ECC Group Size

6.1.3 Function Documentation

6.1.3.1 int ECP_AES_CBC_IV0_DECRYPT (octet * *K*, octet * *C*, octet * *P*)

AES encrypts a plaintext to a ciphertext.

IEEE-1363 AES_CBC_IV0_DECRYPT function. Decrypts in CBC mode with a zero IV.

Parameters

<i>K</i>	AES key
<i>C</i>	input ciphertext octet
<i>P</i>	output plaintext octet

Returns

0 if bad input, else 1

6.1.3.2 void ECP_AES_CBC_IV0_ENCRYPT (octet * *K*, octet * *P*, octet * *C*)

AES encrypts a plaintext to a ciphertext.

IEEE-1363 AES_CBC_IV0_ENCRYPT function. Encrypts in CBC mode with a zero IV, padding as necessary to create a full final block.

Parameters

<i>K</i>	AES key
<i>P</i>	input plaintext octet
<i>C</i>	output ciphertext octet

6.1.3.3 void ECP_CREATE_CSPRNG (csprng * *R*, octet * *S*)

Initialise a random number generator.

Parameters

<i>R</i>	is a pointer to a cryptographically secure random number generator
<i>S</i>	is an input truly random seed value

6.1.3.4 int ECP_ECIES_DECRYPT (octet * *P1*, octet * *P2*, octet * *V*, octet * *C*, octet * *T*, octet * *U*, octet * *M*)

ECIES Decryption.

IEEE-1363 ECIES Decryption

Parameters

<i>P1</i>	input Key Derivation parameters
<i>P2</i>	input Encoding parameters
<i>V</i>	component of the input ciphertext
<i>C</i>	the input ciphertext
<i>T</i>	the input HMAC tag, part of the ciphertext
<i>U</i>	the input private key for decryption
<i>M</i>	the output plaintext message

Returns

1 if successful, else 0

6.1.3.5 void ECP_ECIES_ENCRYPT (octet * *P1*, octet * *P2*, csprng * *R*, octet * *W*, octet * *M*, int *len*, octet * *V*, octet * *C*, octet * *T*)

ECIES Encryption.

IEEE-1363 ECIES Encryption

Parameters

<i>P1</i>	input Key Derivation parameters
<i>P2</i>	input Encoding parameters
<i>R</i>	is a pointer to a cryptographically secure random number generator
<i>W</i>	the input public key of the receiving party
<i>M</i>	is the plaintext message to be encrypted
<i>len</i>	the length of the HMAC tag
<i>V</i>	component of the output ciphertext
<i>C</i>	the output ciphertext
<i>T</i>	the output HMAC tag, part of the ciphertext

6.1.3.6 void ECP_HASH (octet * *I*, octet * *O*)

hash an octet into another octet

Parameters

<i>I</i>	input octet
<i>O</i>	output octet - H(<i>I</i>)

6.1.3.7 int ECP_HMAC (octet * *M*, octet * *K*, int *len*, octet * *tag*)

HMAC of message *M* using key *K* to create tag of length *len* in octet *tag*.

IEEE-1363 MAC1 function. Uses SHA256 internally.

Parameters

<i>M</i>	input message octet
<i>K</i>	input encryption key
<i>len</i>	is output desired length of HMAC tag
<i>tag</i>	is the output HMAC

Returns

0 for bad parameters, else 1

6.1.3.8 void ECP_KDF2 (octet * *Z*, octet * *P*, int *len*, octet * *K*)

Key Derivation Function - generates key *K* from inputs *Z* and *P*.

IEEE-1363 KDF2 Key Derivation Function. Uses SHA256 internally.

Parameters

<i>Z</i>	input octet
<i>P</i>	input key derivation parameters - can be NULL
<i>len</i>	is output desired length of key
<i>K</i>	is the derived key

6.1.3.9 int ECP_KEY_PAIR_GENERATE (csprng * *R*, octet * *s*, octet * *W*)

Generate an ECC public/private key pair.

Parameters

<i>R</i>	is a pointer to a cryptographically secure random number generator
<i>s</i>	the private key, an output internally randomly generated if <i>R</i> !=NULL, otherwise must be provided as an input
<i>W</i>	the output public key, which is <i>s.G</i> , where <i>G</i> is a fixed generator

Returns

0 or an error code

6.1.3.10 void ECP_KILL_CSPRNG (csprng * *R*)

Kill a random number generator.

Deletes all internal state

Parameters

<i>R</i>	is a pointer to a cryptographically secure random number generator
----------	--

6.1.3.11 void ECP_PBKDF2 (octet * *P*, octet * *S*, int *rep*, int *len*, octet * *K*)

Password Based Key Derivation Function - generates key *K* from password, salt and repeat counter.

PBKDF2 Password Based Key Derivation Function. Uses SHA256 internally.

Parameters

<i>P</i>	input password
<i>S</i>	input salt
<i>rep</i>	Number of times to be iterated.
<i>len</i>	is output desired length of key
<i>K</i>	is the derived key

6.1.3.12 int ECP_PUBLIC_KEY_VALIDATE (int *f*, octet * *W*)

Validate an ECC public key.

Parameters

<i>f</i>	if = 0 just does some simple checks, else tests that <i>W</i> is of the correct order
<i>W</i>	the input public key to be validated

Returns

0 if public key is OK, or an error code

6.1.3.13 int ECP_SP_DSA (csprng * *R*, octet * *s*, octet * *M*, octet * *c*, octet * *d*)

ECDSA Signature.

IEEE-1363 ECDSA Signature

Parameters

<i>R</i>	is a pointer to a cryptographically secure random number generator
<i>s</i>	the input private signing key
<i>M</i>	the input message to be signed
<i>c</i>	component of the output signature
<i>d</i>	component of the output signature

6.1.3.14 int ECP_SVDP_DH (octet * *s*, octet * *W*, octet * *K*)

Generate Diffie-Hellman shared key.

IEEE-1363 Diffie-Hellman shared secret calculation

Parameters

<i>s</i>	is the input private key,
<i>W</i>	the input public key of the other party
<i>K</i>	the output shared key, in fact the x-coordinate of s.W

Returns

0 or an error code

6.1.3.15 int ECP_VP_DSA (octet * *W*, octet * *M*, octet * *c*, octet * *d*)

ECDSA Signature Verification.

IEEE-1363 ECDSA Signature Verification

Parameters

<i>W</i>	the input public key
<i>M</i>	the input message
<i>c</i>	component of the input signature
<i>d</i>	component of the input signature

Returns

0 or an error code

6.2 mpin.h File Reference

M-Pin Header file.

```
#include "amcl.h"
```

Macros

- #define [PGS](#) 32
- #define [PFS](#) 32
- #define [PAS](#) 16
- #define [MPIN_OK](#) 0
- #define [MPIN_INVALID_POINT](#) -14
- #define [MPIN_BAD_PIN](#) -19
- #define [MAXPIN](#) 10000
- #define [PBLN](#) 14
- #define [TIME_SLOT_MINUTES](#) 1440
- #define [HASH_BYTES](#) 32

Functions

- DLL_EXPORT void [MPIN_HASH_ID](#) (octet *ID, octet *HID)
Hash an M-Pin Identity to an octet string.
- DLL_EXPORT unsigned int [MPIN_GET_TIME](#) (void)
Get epoch time as unsigned integer.
- DLL_EXPORT void [MPIN_GET_Y](#) (int t, octet *O, octet *Y)
Generate $Y=H(t,O)$, where t is epoch time, O is an octet, and $H(.)$ is a hash function.
- DLL_EXPORT int [MPIN_EXTRACT_PIN](#) (octet *ID, int pin, octet *CS)
Extract a PIN number from a client secret.
- DLL_EXPORT int [MPIN_CLIENT](#) (int d, octet *ID, csprng *R, octet *x, int pin, octet *T, octet *V, octet *U, octet *UT, octet *TP, octet *MESSAGE, int t, octet *y)
Perform client side of the one-pass version of the M-Pin protocol.
- DLL_EXPORT int [MPIN_CLIENT_1](#) (int d, octet *ID, csprng *R, octet *x, int pin, octet *T, octet *S, octet *U, octet *UT, octet *TP)
Perform first pass of the client side of the 3-pass version of the M-Pin protocol.
- DLL_EXPORT int [MPIN_RANDOM_GENERATE](#) (csprng *R, octet *S)
Generate a random group element.
- DLL_EXPORT int [MPIN_CLIENT_2](#) (octet *x, octet *y, octet *V)
Perform second pass of the client side of the 3-pass version of the M-Pin protocol.
- DLL_EXPORT int [MPIN_SERVER](#) (int d, octet *HID, octet *HTID, octet *y, octet *SS, octet *U, octet *UT, octet *V, octet *E, octet *F, octet *ID, octet *MESSAGE, int t)
Perform server side of the one-pass version of the M-Pin protocol.
- DLL_EXPORT void [MPIN_SERVER_1](#) (int d, octet *ID, octet *HID, octet *HTID)
Perform first pass of the server side of the 3-pass version of the M-Pin protocol.

- DLL_EXPORT int [MPIN_SERVER_2](#) (int d, octet *HID, octet *HTID, octet *y, octet *SS, octet *U, octet *UT, octet *V, octet *E, octet *F)
Perform third pass on the server side of the 3-pass version of the M-Pin protocol.
- DLL_EXPORT int [MPIN_RECOMBINE_G1](#) (octet *Q1, octet *Q2, octet *Q)
Add two members from the group G1.
- DLL_EXPORT int [MPIN_RECOMBINE_G2](#) (octet *P1, octet *P2, octet *P)
Add two members from the group G2.
- DLL_EXPORT int [MPIN_KANGAROO](#) (octet *E, octet *F)
Use Kangaroos to find PIN error.
- DLL_EXPORT int [MPIN_ENCODING](#) (csprng *R, octet *TP)
Encoding of a Time Permit to make it indistinguishable from a random string.
- DLL_EXPORT int [MPIN_DECODING](#) (octet *TP)
Encoding of an obfuscated Time Permit.
- DLL_EXPORT unsigned32 [MPIN_today](#) (void)
Supply today's date as days from the epoch.
- DLL_EXPORT void [MPIN_CREATE_CSPRNG](#) (csprng *R, octet *S)
Initialise a random number generator.
- DLL_EXPORT void [MPIN_KILL_CSPRNG](#) (csprng *R)
Kill a random number generator.
- DLL_EXPORT int [MPIN_GET_G1_MULTIPLE](#) (csprng *R, int type, octet *x, octet *G, octet *W)
Find a random multiple of a point in G1.
- DLL_EXPORT int [MPIN_GET_CLIENT_SECRET](#) (octet *S, octet *ID, octet *CS)
Create a client secret in G1 from a master secret and the client ID.
- DLL_EXPORT int [MPIN_GET_CLIENT_PERMIT](#) (int d, octet *S, octet *ID, octet *TP)
Create a Time Permit in G1 from a master secret and the client ID.
- DLL_EXPORT int [MPIN_GET_SERVER_SECRET](#) (octet *S, octet *SS)
Create a server secret in G2 from a master secret.
- DLL_EXPORT int [MPIN_PRECOMPUTE](#) (octet *T, octet *ID, octet *g1, octet *g2)
Precompute values for use by the client side of M-Pin Full.
- DLL_EXPORT int [MPIN_SERVER_KEY](#) (octet *Z, octet *SS, octet *w, octet *p, octet *l, octet *U, octet *UT, octet *K)
Calculate Key on Server side for M-Pin Full.
- DLL_EXPORT int [MPIN_CLIENT_KEY](#) (octet *g1, octet *g2, int pin, octet *r, octet *x, octet *p, octet *T, octet *K)
Calculate Key on Client side for M-Pin Full.
- DLL_EXPORT void [MPIN_AES_GCM_ENCRYPT](#) (octet *K, octet *IV, octet *H, octet *P, octet *C, octet *T)
AES-GCM Encryption.
- DLL_EXPORT void [MPIN_AES_GCM_DECRYPT](#) (octet *K, octet *IV, octet *H, octet *C, octet *P, octet *T)
AES-GCM Decryption.
- DLL_EXPORT int [MPIN_HMAC](#) (octet *M, octet *K, int len, octet *tag)
HMAC of message M using key K to create tag of length len in octet tag.
- DLL_EXPORT void [MPIN_PBKDF2](#) (octet *P, octet *S, int rep, int len, octet *K)
Password Based Key Derivation Function - generates key K from password, salt and repeat counter.
- DLL_EXPORT void [MPIN_HASH_ALL](#) (octet *I, octet *U, octet *CU, octet *V, octet *Y, octet *R, octet *W, octet *H)
Hash the session transcript.

6.2.1 Detailed Description

M-Pin Header file.

Author

Mike Scott and Kealan McCusker

Date

2nd June 2015 Allows some user configuration defines structures declares functions

6.2.2 Macro Definition Documentation

6.2.2.1 `#define HASH_BYTES 32`

Number of bytes output by Hash function

6.2.2.2 `#define MAXPIN 10000`

max PIN

6.2.2.3 `#define MPIN_BAD_PIN -19`

Bad PIN number entered

6.2.2.4 `#define MPIN_INVALID_POINT -14`

Point is NOT on the curve

6.2.2.5 `#define MPIN_OK 0`

Function completed without error

6.2.2.6 `#define PAS 16`

MPIN Symmetric Key Size

6.2.2.7 `#define PBLLEN 14`

max length of PIN in bits

6.2.2.8 #define PFS 32

MPIN Field Size

6.2.2.9 #define PGS 32

MPIN Group Size

6.2.2.10 #define TIME_SLOT_MINUTES 1440

Time Slot = 1 day

6.2.3 Function Documentation

6.2.3.1 DLL_EXPORT void MPIN_AES_GCM_DECRYPT (octet * *K*, octet * *IV*, octet * *H*, octet * *C*, octet * *P*, octet * *T*)

AES-GCM Decryption.

Parameters

<i>K</i>	AES key
<i>IV</i>	Initialization vector
<i>H</i>	Header
<i>P</i>	Plaintext
<i>C</i>	Ciphertext
<i>T</i>	Checksum

6.2.3.2 DLL_EXPORT void MPIN_AES_GCM_ENCRYPT (octet * *K*, octet * *IV*, octet * *H*, octet * *P*, octet * *C*, octet * *T*)

AES-GCM Encryption.

Parameters

<i>K</i>	AES key
<i>IV</i>	Initialization vector
<i>H</i>	Header
<i>P</i>	Plaintext
<i>C</i>	Ciphertext
<i>T</i>	Checksum

6.2.3.3 DLL_EXPORT int MPIN_CLIENT (int *d*, octet * *ID*, csprng * *R*, octet * *x*, int *pin*, octet * *T*, octet * *V*, octet * *U*, octet * *UT*, octet * *TP*, octet * *MESSAGE*, int *t*, octet * *y*)

Perform client side of the one-pass version of the M-Pin protocol.

If Time Permits are disabled, set $d = 0$, and UT is not generated and can be set to `NULL`. If Time Permits are enabled, and PIN error detection is OFF, U is not generated and can be set to `NULL`. If Time Permits are enabled, and PIN error detection is ON, U and UT are both generated.

Parameters

d	is input date, in days since the epoch. Set to 0 if Time permits disabled
ID	is the input client identity
R	is a pointer to a cryptographically secure random number generator
x	an output internally randomly generated if $R \neq \text{NULL}$, otherwise must be provided as an input
pin	is the input PIN number
T	is the input M-Pin token (the client secret with PIN portion removed)
V	is output $= -(x+y)(CS+TP)$, where CS is the reconstructed client secret, and TP is the time permit
U	is output $= x.H(ID)$
UT	is output $= x.(H(ID)+H(d H(ID)))$
TP	is the input time permit
$MESSAGE$	is the message to be signed
t	is input epoch time in seconds - a timestamp
y	is output $H(t U)$ or $H(t UT)$ if Time Permits enabled

Returns

0 or an error code

6.2.3.4 `DLL_EXPORT int MPIN_CLIENT_1 (int d , octet * ID , csprng * R , octet * x , int pin , octet * T , octet * S , octet * U , octet * UT , octet * TP)`

Perform first pass of the client side of the 3-pass version of the M-Pin protocol.

If Time Permits are disabled, set $d = 0$, and UT is not generated and can be set to `NULL`. If Time Permits are enabled, and PIN error detection is OFF, U is not generated and can be set to `NULL`. If Time Permits are enabled, and PIN error detection is ON, U and UT are both generated.

Parameters

d	is input date, in days since the epoch. Set to 0 if Time permits disabled
ID	is the input client identity
R	is a pointer to a cryptographically secure random number generator
x	an output internally randomly generated if $R \neq \text{NULL}$, otherwise must be provided as an input
pin	is the input PIN number
T	is the input M-Pin token (the client secret with PIN portion removed)
S	is output $= CS+TP$, where CS =is the reconstructed client secret, and TP is the time permit
U	is output $= x.H(ID)$
UT	is output $= x.(H(ID)+H(d H(ID)))$
TP	is the input time permit

Returns

0 or an error code

6.2.3.5 DLL_EXPORT int MPIN_CLIENT_2 (octet * *x*, octet * *y*, octet * *V*)

Perform second pass of the client side of the 3-pass version of the M-Pin protocol.

Parameters

<i>x</i>	an input, a locally generated random number
<i>y</i>	an input random challenge from the server
<i>V</i>	on output = $-(x+y).V$

Returns

0 or an error code

6.2.3.6 DLL_EXPORT int MPIN_CLIENT_KEY (octet * *g1*, octet * *g2*, int *pin*, octet * *r*, octet * *x*, octet * *p*, octet * *T*, octet * *K*)

Calculate Key on Client side for M-Pin Full.

Parameters

<i>g1</i>	precomputed input
<i>g2</i>	precomputed input
<i>pin</i>	is the input PIN number
<i>r</i>	is an input, a locally generated random number
<i>x</i>	is an input, a locally generated random number
<i>p</i>	is an input, hash of the protocol transcript
<i>T</i>	is the input Server-side Diffie-Hellman component
<i>K</i>	is the output calculated shared key

Returns

0 or an error code

6.2.3.7 DLL_EXPORT void MPIN_CREATE_CSPRNG (csprng * *R*, octet * *S*)

Initialise a random number generator.

Parameters

<i>R</i>	is a pointer to a cryptographically secure random number generator
<i>S</i>	is an input truly random seed value

6.2.3.8 DLL_EXPORT int MPIN_DECODING (octet * *TP*)

Encoding of an obfuscated Time Permit.

Parameters

<i>TP</i>	is the input obfuscated time permit, restored on output
-----------	---

Returns

0 or an error code

6.2.3.9 DLL_EXPORT int MPIN_ENCODING (csprng * *R*, octet * *TP*)

Encoding of a Time Permit to make it indistinguishable from a random string.

Parameters

<i>R</i>	is a pointer to a cryptographically secure random number generator
<i>TP</i>	is the input time permit, obfuscated on output

Returns

0 or an error code

6.2.3.10 DLL_EXPORT int MPIN_EXTRACT_PIN (octet * *ID*, int *pin*, octet * *CS*)

Extract a PIN number from a client secret.

Parameters

<i>ID</i>	is the input client identity
<i>pin</i>	is an input PIN number
<i>CS</i>	is the client secret from which the PIN is to be extracted

Returns

0 or an error code

6.2.3.11 DLL_EXPORT int MPIN_GET_CLIENT_PERMIT (int *d*, octet * *S*, octet * *ID*, octet * *TP*)

Create a Time Permit in G1 from a master secret and the client ID.

Parameters

<i>d</i>	is input date, in days since the epoch.
<i>S</i>	is an input master secret
<i>ID</i>	is the input client identity
<i>TP</i>	is a Time Permit for the given date = $s.H(d H(ID))$

Returns

0 or an error code

6.2.3.12 DLL_EXPORT int MPIN_GET_CLIENT_SECRET (*octet * S*, *octet * ID*, *octet * CS*)

Create a client secret in G1 from a master secret and the client ID.

Parameters

<i>S</i>	is an input master secret
<i>ID</i>	is the input client identity
<i>CS</i>	is the full client secret = $s.H(ID)$

Returns

0 or an error code

6.2.3.13 DLL_EXPORT int MPIN_GET_G1_MULTIPLE (*csprng * R*, *int type*, *octet * x*, *octet * G*, *octet * W*)

Find a random multiple of a point in G1.

Parameters

<i>R</i>	is a pointer to a cryptographically secure random number generator
<i>type</i>	determines type of action to be taken
<i>x</i>	an output internally randomly generated if $R \neq \text{NULL}$, otherwise must be provided as an input
<i>G</i>	if $\text{type}=0$ a point in G1, else an octet to be mapped to G1
<i>W</i>	the output $=x.G$ or $x.M(G)$, where $M(.)$ is a mapping

Returns

0 or an error code

6.2.3.14 DLL_EXPORT int MPIN_GET_SERVER_SECRET (*octet * S*, *octet * SS*)

Create a server secret in G2 from a master secret.

Parameters

<i>S</i>	is an input master secret
<i>SS</i>	is the server secret = $s.Q$ where Q is a fixed generator of G_2

Returns

0 or an error code

6.2.3.15 DLL_EXPORT unsigned MPIN_GET_TIME (void)

Get epoch time as unsigned integer.

Returns

current epoch time in seconds

6.2.3.16 DLL_EXPORT void MPIN_GET_Y (int *t*, octet * *O*, octet * *Y*)

Generate $Y=H(t,O)$, where t is epoch time, O is an octet, and $H(.)$ is a hash function.

Parameters

<i>t</i>	is epoch time in seconds
<i>O</i>	is an input octet
<i>Y</i>	is the output octet

6.2.3.17 DLL_EXPORT void MPIN_HASH_ALL (octet * *I*, octet * *U*, octet * *CU*, octet * *V*, octet * *Y*, octet * *R*, octet * *W*, octet * *H*)

Hash the session transcript.

Parameters

<i>I</i>	is the hashed input client ID = $H(ID)$
<i>U</i>	is the client output = $x.H(ID)$
<i>CU</i>	is the client output = $x.(H(ID)+H(T H(ID)))$
<i>Y</i>	is the server challenge
<i>V</i>	is the client part response
<i>R</i>	is the client part response
<i>W</i>	is the server part response
<i>H</i>	the output is the hash of all of the above that apply

6.2.3.18 DLL_EXPORT void MPIN_HASH_ID (octet * *ID*, octet * *HID*)

Hash an M-Pin Identity to an octet string.

Parameters

<i>ID</i>	an octet containing the identity
<i>HID</i>	an octet containing the hashed identity

6.2.3.19 DLL_EXPORT int MPIN_HMAC (octet * *M*, octet * *K*, int *len*, octet * *tag*)

HMAC of message M using key K to create tag of length len in octet tag.

IEEE-1363 MAC1 function. Uses SHA256 internally.

Parameters

<i>M</i>	input message octet
<i>K</i>	input encryption key
<i>len</i>	is output desired length of HMAC tag
<i>tag</i>	is the output HMAC

Returns

0 for bad parameters, else 1

6.2.3.20 DLL_EXPORT int MPIN_KANGAROO (octet * *E*, octet * *F*)

Use Kangaroos to find PIN error.

Parameters

<i>E</i>	a member of the group GT
<i>F</i>	a member of the group GT = E^e

Returns

0 if Kangaroos failed, or the PIN error e

6.2.3.21 DLL_EXPORT void MPIN_KILL_CSPRNG (csprng * *R*)

Kill a random number generator.

Deletes all internal state

Parameters

<i>R</i>	is a pointer to a cryptographically secure random number generator
----------	--

6.2.3.22 DLL_EXPORT void MPIN_PBKDF2 (*octet * P*, *octet * S*, *int rep*, *int len*, *octet * K*)

Password Based Key Derivation Function - generates key K from password, salt and repeat counter.

PBKDF2 Password Based Key Derivation Function. Uses SHA256 internally.

Parameters

<i>P</i>	input password
<i>S</i>	input salt
<i>rep</i>	Number of times to be iterated.
<i>len</i>	is output desired length of key
<i>K</i>	is the derived key

6.2.3.23 DLL_EXPORT int MPIN_PRECOMPUTE (*octet * T*, *octet * ID*, *octet * g1*, *octet * g2*)

Precompute values for use by the client side of M-Pin Full.

Parameters

<i>T</i>	is the input M-Pin token (the client secret with PIN portion removed)
<i>ID</i>	is the input client identity
<i>g1</i>	precomputed output
<i>g2</i>	precomputed output

Returns

0 or an error code

6.2.3.24 DLL_EXPORT int MPIN_RANDOM_GENERATE (*cspng * R*, *octet * S*)

Generate a random group element.

Parameters

<i>R</i>	is a pointer to a cryptographically secure random number generator
<i>S</i>	is the output random octet

Returns

0 or an error code

6.2.3.25 DLL_EXPORT int MPIN_RECOMBINE_G1 (octet * *Q1*, octet * *Q2*, octet * *Q*)

Add two members from the group G1.

Parameters

<i>Q1</i>	an input member of G1
<i>Q2</i>	an input member of G1
<i>Q</i>	an output member of $G1 = Q1 + Q2$

Returns

0 or an error code

6.2.3.26 DLL_EXPORT int MPIN_RECOMBINE_G2 (octet * *P1*, octet * *P2*, octet * *P*)

Add two members from the group G2.

Parameters

<i>P1</i>	an input member of G2
<i>P2</i>	an input member of G2
<i>P</i>	an output member of $G2 = P1 + P2$

Returns

0 or an error code

6.2.3.27 DLL_EXPORT int MPIN_SERVER (int *d*, octet * *HID*, octet * *HTID*, octet * *y*, octet * *SS*, octet * *U*, octet * *UT*, octet * *V*, octet * *E*, octet * *F*, octet * *ID*, octet * *MESSAGE*, int *t*)

Perform server side of the one-pass version of the M-Pin protocol.

If Time Permits are disabled, set $d = 0$, and UT and $HTID$ are not generated and can be set to `NULL`. If Time Permits are enabled, and PIN error detection is OFF, U and HID are not needed and can be set to `NULL`. If Time Permits are enabled, and PIN error detection is ON, U , UT , HID and $HTID$ are all required.

Parameters

<i>d</i>	is input date, in days since the epoch. Set to 0 if Time permits disabled
<i>HID</i>	is output $H(ID)$, a hash of the client ID
<i>HTID</i>	is output $H(ID) + H(d H(ID))$
<i>y</i>	is output $H(t U)$ or $H(t UT)$ if Time Permits enabled

Parameters

<i>SS</i>	is the input server secret
<i>U</i>	is input from the client = $x.H(ID)$
<i>UT</i>	is input from the client = $x.(H(ID)+H(d H(ID)))$
<i>V</i>	is an input from the client
<i>E</i>	is an output to help the Kangaroos to find the PIN error, or NULL if not required
<i>F</i>	is an output to help the Kangaroos to find the PIN error, or NULL if not required
<i>ID</i>	is the input claimed client identity
<i>MESSAGE</i>	is the message to be signed
<i>t</i>	is input epoch time in seconds - a timestamp

Returns

0 or an error code

6.2.3.28 `DLL_EXPORT void MPIN_SERVER_1 (int d, octet * ID, octet * HID, octet * HTID)`

Perform first pass of the server side of the 3-pass version of the M-Pin protocol.

Parameters

<i>d</i>	is input date, in days since the epoch. Set to 0 if Time permits disabled
<i>ID</i>	is the input claimed client identity
<i>HID</i>	is output $H(ID)$, a hash of the client ID
<i>HTID</i>	is output $H(ID)+H(d H(ID))$

Returns

0 or an error code

6.2.3.29 `DLL_EXPORT int MPIN_SERVER_2 (int d, octet * HID, octet * HTID, octet * y, octet * SS, octet * U, octet * UT, octet * V, octet * E, octet * F)`

Perform third pass on the server side of the 3-pass version of the M-Pin protocol.

If Time Permits are disabled, set $d = 0$, and UT and $HTID$ are not needed and can be set to NULL. If Time Permits are enabled, and PIN error detection is OFF, U and HID are not needed and can be set to NULL. If Time Permits are enabled, and PIN error detection is ON, U , UT , HID and $HTID$ are all required.

Parameters

<i>d</i>	is input date, in days since the epoch. Set to 0 if Time permits disabled
<i>HID</i>	is input $H(ID)$, a hash of the client ID
<i>HTID</i>	is input $H(ID)+H(d H(ID))$
<i>y</i>	is the input server's randomly generated challenge
<i>SS</i>	is the input server secret

Parameters

<i>U</i>	is input from the client = $x.H(ID)$
<i>UT</i>	is input from the client = $x.(H(ID)+H(d H(ID)))$
<i>V</i>	is an input from the client
<i>E</i>	is an output to help the Kangaroos to find the PIN error, or NULL if not required
<i>F</i>	is an output to help the Kangaroos to find the PIN error, or NULL if not required

Returns

0 or an error code

6.2.3.30 `DLL_EXPORT int MPIN_SERVER_KEY (octet * Z, octet * SS, octet * w, octet * p, octet * I, octet * U, octet * UT, octet * K)`

Calculate Key on Server side for M-Pin Full.

Uses UT internally for the key calculation, unless not available in which case U is used

Parameters

<i>Z</i>	is the input Client-side Diffie-Hellman component
<i>SS</i>	is the input server secret
<i>w</i>	is an input random number generated by the server
<i>p</i>	is an input, hash of the protocol transcript
<i>I</i>	is the hashed input client ID = $H(ID)$
<i>U</i>	is input from the client = $x.H(ID)$
<i>UT</i>	is input from the client = $x.(H(ID)+H(d H(ID)))$
<i>K</i>	is the output calculated shared key

Returns

0 or an error code

6.2.3.31 `DLL_EXPORT unsigned MPIN_today (void)`

Supply today's date as days from the epoch.

Returns

today's date, as number of days elapsed since the epoch

6.3 rsa.h File Reference

RSA Header file for implementation of RSA protocol.

```
#include "amcl.h"
```

Macros

- #define [RFS](#) MODBYTES*FFLEN

Functions

- void [RSA_CREATE_CSPRNG](#) (csprng *R, octet *S)
Initialise a random number generator.
- void [RSA_KILL_CSPRNG](#) (csprng *R)
Kill a random number generator.
- void [RSA_KEY_PAIR](#) (csprng *R, sign32 e, rsa_private_key *PRIV, rsa_public_key *PUB)
RSA Key Pair Generator.
- int [RSA_OAEP_ENCODE](#) (octet *M, csprng *R, octet *P, octet *F)
OAEP padding of a message prior to RSA encryption.
- int [RSA_OAEP_DECODE](#) (octet *P, octet *F)
OAEP unpadding of a message after RSA decryption.
- void [RSA_ENCRYPT](#) (rsa_public_key *PUB, octet *F, octet *G)
RSA encryption of suitably padded plaintext.
- void [RSA_DECRYPT](#) (rsa_private_key *PRIV, octet *G, octet *F)
RSA decryption of ciphertext.
- void [RSA_PRIVATE_KEY_KILL](#) (rsa_private_key *PRIV)
Destroy an RSA private Key.

6.3.1 Detailed Description

RSA Header file for implementation of RSA protocol.

Author

Mike Scott and Kealan McCusker

Date

2nd June 2015 declares functions

6.3.2 Macro Definition Documentation

6.3.2.1 #define RFS MODBYTES*FFLEN

RSA Public Key Size in bytes

6.3.3 Function Documentation

6.3.3.1 void RSA_CREATE_CSPRNG (csprng * R, octet * S)

Initialise a random number generator.

Parameters

<i>R</i>	is a pointer to a cryptographically secure random number generator
<i>S</i>	is an input truly random seed value

6.3.3.2 void RSA_DECRYPT (rsa_private_key * *PRIV*, octet * *G*, octet * *F*)

RSA decryption of ciphertext.

Parameters

P_{\leftarrow} <i>RIV</i>	the input RSA private key
<i>G</i>	is the input ciphertext
<i>F</i>	is output plaintext (requires unpadding)

6.3.3.3 void RSA_ENCRYPT (rsa_public_key * *PUB*, octet * *F*, octet * *G*)

RSA encryption of suitably padded plaintext.

Parameters

<i>PUB</i>	the input RSA public key
<i>F</i>	is input padded message
<i>G</i>	is the output ciphertext

6.3.3.4 void RSA_KEY_PAIR (csprng * *R*, sign32 *e*, rsa_private_key * *PRIV*, rsa_public_key * *PUB*)

RSA Key Pair Generator.

Parameters

<i>R</i>	is a pointer to a cryptographically secure random number generator
<i>e</i>	the encryption exponent
P_{\leftarrow} <i>RIV</i>	the output RSA private key
<i>PUB</i>	the output RSA public key

6.3.3.5 void RSA_KILL_CSPRNG (csprng * *R*)

Kill a random number generator.

Deletes all internal state

Parameters

<i>R</i>	is a pointer to a cryptographically secure random number generator
----------	--

6.3.3.6 int RSA_OAEP_DECODE (octet * *P*, octet * *F*)

OAEP unpadding of a message after RSA decryption.

Unpadding is done in-place

Parameters

<i>P</i>	are input encoding parameter string (could be NULL)
<i>F</i>	is input padded message, unpadded on output

Returns

1 if OK, else 0

6.3.3.7 int RSA_OAEP_ENCODE (octet * *M*, csprng * *R*, octet * *P*, octet * *F*)

OAEP padding of a message prior to RSA encryption.

Parameters

<i>M</i>	is the input message
<i>R</i>	is a pointer to a cryptographically secure random number generator
<i>P</i>	are input encoding parameter string (could be NULL)
<i>F</i>	is the output encoding, ready for RSA encryption

Returns

1 if OK, else 0

6.3.3.8 void RSA_PRIVATE_KEY_KILL (rsa_private_key * *PRIV*)

Destroy an RSA private Key.

Parameters

<i>P</i> ↔ <i>RIV</i>	the input RSA private key. Destroyed on output.
--------------------------	---

6.4 wcc.c File Reference

Wang / Chow Choo (WCC) definitions.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include "wcc.h"
```

Functions

- void [WCC_Hq](#) (octet *A, octet *B, octet *C, octet *D, octet *h)
Hash EC Points and Id to an integer.
- int [WCC_GET_G1_MULTIPLE](#) (int hashDone, octet *S, octet *ID, octet *VG1)
Calculate value in G1 multiplied by an integer.
- int [WCC_GET_G1_TPMULT](#) (int date, octet *S, octet *ID, octet *VG1)
Calculate a value in G1 used for when time permits are enabled.
- int [WCC_GET_G2_TPMULT](#) (int date, octet *S, octet *ID, octet *VG2)
Calculate a value in G2 used for when time permits are enabled.
- int [WCC_GET_G2_MULTIPLE](#) (int hashDone, octet *S, octet *ID, octet *VG2)
Calculate value in G2 multiplied by an integer.
- int [WCC_GET_G2_PERMIT](#) (int date, octet *S, octet *HID, octet *TPG2)
Calculate time permit in G2.
- int [WCC_SENDER_KEY](#) (int date, octet *xOct, octet *piaOct, octet *pibOct, octet *PbG2Oct, octet *PgG1Oct, octet *AKeyG1Oct, octet *ATPG1Oct, octet *IdBOct, octet *AESKeyOct)
Calculate the sender AES key.
- int [WCC_RECEIVER_KEY](#) (int date, octet *yOct, octet *wOct, octet *piaOct, octet *pibOct, octet *PaG1Oct, octet *PgG1Oct, octet *BKeyG2Oct, octet *BTPG2Oct, octet *IdAOct, octet *AESKeyOct)
Calculate the receiver AES key.
- void [WCC_AES_GCM_ENCRYPT](#) (octet *K, octet *IV, octet *H, octet *P, octet *C, octet *T)
Encrypt data using AES GCM.
- void [WCC_AES_GCM_DECRYPT](#) (octet *K, octet *IV, octet *H, octet *C, octet *P, octet *T)
Decrypt data using AES GCM.
- unsigned int [WCC_today](#) (void)
Get today's date as days from the epoch.
- void [WCC_CREATE_CSPRNG](#) (csprng *RNG, octet *SEED)
Initialise a random number generator.
- void [WCC_KILL_CSPRNG](#) (csprng *RNG)
Kill a random number generator.
- void [WCC_HASH_ID](#) (octet *ID, octet *HID)
Perform sha256.
- int [WCC_RANDOM_GENERATE](#) (csprng *RNG, octet *S)
Generate a random integer.
- int [WCC_GET_G1_PERMIT](#) (int date, octet *S, octet *HID, octet *TPG1)
Calculate time permit in G2.
- int [WCC_RECOMBINE_G1](#) (octet *R1, octet *R2, octet *R)
Add two members from the group G1.
- int [WCC_RECOMBINE_G2](#) (octet *W1, octet *W2, octet *W)
Add two members from the group G2.

6.4.1 Detailed Description

Wang / Chow Choo (WCC) definitions.

Author

Mike Scott and Kealan McCusker

Date

28th April 2016

6.4.2 Function Documentation

6.4.2.1 void WCC_AES_GCM_DECRYPT (*octet * K*, *octet * IV*, *octet * H*, *octet * C*, *octet * P*, *octet * T*)

Decrypt data using AES GCM.

AES is run as a block cypher in the GCM mode of operation. The key size is 128 bits. This function will decrypt any data length.

Parameters

<i>K</i>	128 bit secret key
<i>IV</i>	96 bit initialization vector
<i>H</i>	Additional authenticated data (AAD). This data is authenticated, but not encrypted.
<i>C</i>	Ciphertext.

Returns

P Decrypted data. It is the same length as the ciphertext.Plaintext
T 128 bit authentication tag.

6.4.2.2 void WCC_AES_GCM_ENCRYPT (*octet * K*, *octet * IV*, *octet * H*, *octet * P*, *octet * C*, *octet * T*)

Encrypt data using AES GCM.

AES is run as a block cypher in the GCM mode of operation. The key size is 128 bits. This function will encrypt any data length.

Parameters

<i>K</i>	128 bit secret key
<i>IV</i>	96 bit initialization vector
<i>H</i>	Additional authenticated data (AAD). This data is authenticated, but not encrypted.
<i>P</i>	Plaintext

Returns

C Ciphertext. It is the same length as the plaintext.
T 128 bit authentication tag.

6.4.2.3 void WCC_CREATE_CSPRNG (csprng * *RNG*, octet * *SEED*)

Initialise a random number generator.

Parameters

<i>RNG</i>	cryptographically secure random number generator
<i>SEED</i>	random seed value

6.4.2.4 int WCC_GET_G1_MULTIPLE (int *hashDone*, octet * *S*, octet * *ID*, octet * *VG1*)

Calculate value in G1 multiplied by an integer.

Calculate a value in G1. $VG1 = s * H1(ID)$ where ID is the identity.

1. $VG1 = s * H1(ID)$

Parameters

<i>hashDone</i>	ID value is already hashed if set to 1
<i>S</i>	integer modulus curve order
<i>ID</i>	ID value or sha256(ID)
<i>VG1</i>	EC point $VG1 = s * H1(ID)$

Returns

rtm Returns 0 if successful or else an error code

6.4.2.5 int WCC_GET_G1_PERMIT (int *date*, octet * *S*, octet * *HID*, octet * *TPG1*)

Calculate time permit in G2.

Calculate time permit in G2.

1. $TPG1 = s * H1(date || sha256(ID))$

Parameters

<i>date</i>	Epoch days
<i>S</i>	Master secret
<i>HID</i>	sha256(ID)
<i>TPG1</i>	Time Permit in G1

Returns

rtm Returns 0 if successful or else an error code

6.4.2.6 int WCC_GET_G1_TPMULT (int *date*, octet * *S*, octet * *ID*, octet * *VG1*)

Calculate a value in G1 used for when time permits are enabled.

Calculate a value in G1 used for when time permits are enabled

1. $VG1 = s * H1(ID) + s * H1(date || sha256(ID))$

Parameters

<i>date</i>	Epoch days
<i>S</i>	integer modulus curve order
<i>ID</i>	ID value or sha256(ID)
<i>VG1</i>	EC point in G1

Returns

rtm Returns 0 if successful or else an error code

6.4.2.7 int WCC_GET_G2_MULTIPLE (int *hashDone*, octet * *S*, octet * *ID*, octet * *VG2*)

Calculate value in G2 multiplied by an integer.

Calculate a value in G2. $VG2 = s * H2(ID)$ where ID is the identity.

1. $VG2 = s * H2(ID)$

Parameters

<i>hashDone</i>	ID is value is already hashed if set to 1
<i>S</i>	integer modulus curve order
<i>ID</i>	ID Value or sha256(ID)
<i>VG2</i>	EC Point $VG2 = s * H2(ID)$

Returns

rtm Returns 0 if successful or else an error code

6.4.2.8 int WCC_GET_G2_PERMIT (int *date*, octet * *S*, octet * *HID*, octet * *TPG2*)

Calculate time permit in G2.

Calculate time permit in G2.

1. $TPG2 = s * H2(date | sha256(ID))$

Parameters

<i>date</i>	Epoch days
<i>S</i>	Master secret
<i>HID</i>	sha256(ID)
<i>TPG2</i>	Time Permit in G2

Returns

rtm Returns 0 if successful or else an error code

6.4.2.9 `int WCC_GET_G2_TPMULT (int date, octet * S, octet * ID, octet * VG2)`

Calculate a value in G2 used for when time permits are enabled.

Calculate a value in G2 used for when time permits are enabled

1. $VG2 = s * H1(ID) + s * H1(date | sha256(ID))$

Parameters

<i>date</i>	Epoch days
<i>S</i>	integer modulus curve order
<i>ID</i>	ID value or sha256(ID)
<i>VG2</i>	EC point in G2

Returns

rtm Returns 0 if successful or else an error code

6.4.2.10 `void WCC_HASH_ID (octet * ID, octet * HID)`

Perform sha256.

Hash ID

Parameters

<i>ID</i>	Value to hash
-----------	---------------

Returns

HID sha256 hashed value

6.4.2.11 void WCC_Hq (octet * *A*, octet * *B*, octet * *C*, octet * *D*, octet * *h*)

Hash EC Points and Id to an integer.

Perform sha256 of EC Points and Id. Map to an integer modulo the curve order

1. $x = \text{toInteger}(\text{sha256}(A,B,C,D))$
2. $h = x \% q$ where q is the curve order

Parameters

<i>A</i>	EC Point
<i>B</i>	EC Point
<i>C</i>	EC Point
<i>D</i>	Identity

Returns

h Integer

6.4.2.12 void WCC_KILL_CSPRNG (csprng * *RNG*)

Kill a random number generator.

Deletes all internal state

Parameters

<i>RNG</i>	cryptographically secure random number generator
------------	--

6.4.2.13 int WCC_RANDOM_GENERATE (csprng * *RNG*, octet * *S*)

Generate a random integer.

Generate a random number modulus the group order

Parameters

<i>RNG</i>	cryptographically secure random number generator
------------	--

Returns

S Random integer modulus the group order

6.4.2.14 `int WCC_RECEIVER_KEY (int date, octet * yOct, octet * wOct, octet * piaOct, octet * pibOct, octet * PaG1Oct, octet * PgG1Oct, octet * BKeyG2Oct, octet * BTPG2Oct, octet * IdAOct, octet * AESKeyOct)`

Calculate the receiver AES key.

Calculate time permit in G2.

1. $j = e(\text{pia}.\text{AG1} + \text{PaG1}, (y + \text{pib}).\text{BKeyG2})$
2. $K = H(j, w.\text{PaG1})$

Parameters

<i>date</i>	Epoch days
<i>yOct</i>	Random $y < q$ where q is the curve order
<i>wOct</i>	Random $w < q$ where q is the curve order
<i>piaOct</i>	$H_q(\text{PaG1}, \text{PbG2}, \text{PgG1})$
<i>pibOct</i>	$H_q(\text{PbG2}, \text{PaG1}, \text{PgG1})$
<i>PaG1Oct</i>	$x.\text{AG1}$ where $x < q$
<i>PgG1Oct</i>	$w.\text{AG1}$ where $w < q$
<i>BKeyG2Oct</i>	Receiver key
<i>BTPG2Oct</i>	Receiver time permit
<i>IdAOct</i>	Sender identity

Returns

AESKeyOct AES key
 rtn Returns 0 if successful or else an error code

6.4.2.15 `int WCC_RECOMBINE_G1 (octet * R1, octet * R2, octet * R)`

Add two members from the group G1.

Parameters

<i>R1</i>	member of G1
<i>R2</i>	member of G1

Returns

R member of G1 = $R1 + R2$
 Returns 0 if successful or else an error code

6.4.2.16 int WCC_RECOMBINE_G2 (octet * *W1*, octet * *W2*, octet * *W*)

Add two members from the group G2.

Parameters

<i>W1</i>	member of G2
<i>W2</i>	member of G2

Returns

W member of G2 = W1+W2

Weturns 0 if successful or else an error code

6.4.2.17 int WCC_SENDER_KEY (int *date*, octet * *xOct*, octet * *piaOct*, octet * *pibOct*, octet * *PbG2Oct*, octet * *PgG1Oct*, octet * *AKeyG1Oct*, octet * *ATPG1Oct*, octet * *IdBOct*, octet * *AESKeyOct*)

Calculate the sender AES key.

Calculate the sender AES Key

1. $j = e((x + pia).AKeyG1, pib.BG2 + PbG2)$
2. $K = H(j, x.PgG1)$

Parameters

<i>date</i>	Epoch days
<i>xOct</i>	Random $x < q$ where q is the curve order
<i>piaOct</i>	$H_q(PaG1, PbG2, PgG1)$
<i>pibOct</i>	$H_q(PbG2, PaG1, PgG1)$
<i>PbG2Oct</i>	$y.BG2$ where $y < q$
<i>PgG1Oct</i>	$w.AG1$ where $w < q$
<i>AKeyG1Oct</i>	Sender key
<i>ATPG1Oct</i>	Sender time permit
<i>IdBOct</i>	Receiver identity

Returns

AESKeyOct AES key

rtm Returns 0 if successful or else an error code

6.4.2.18 unsigned WCC_today (void)

Get today's date as days from the epoch.

Returns

today's date, as number of days elapsed since the epoch

Index

EAS

ecdh.h, [12](#)

ECDH_ERROR

ecdh.h, [12](#)

ECDH_INVALID_PUBLIC_KEY

ecdh.h, [12](#)

ECDH_INVALID

ecdh.h, [12](#)

ECDH_OK

ecdh.h, [12](#)

ECP_AES_CBC_IV0_DECRYPT

ecdh.h, [13](#)

ECP_AES_CBC_IV0_ENCRYPT

ecdh.h, [13](#)

ECP_CREATE_CSPRNG

ecdh.h, [13](#)

ECP_ECIES_DECRYPT

ecdh.h, [14](#)

ECP_ECIES_ENCRYPT

ecdh.h, [14](#)

ECP_HASH

ecdh.h, [15](#)

ECP_HMAC

ecdh.h, [15](#)

ECP_KDF2

ecdh.h, [15](#)

ECP_KEY_PAIR_GENERATE

ecdh.h, [15](#)

ECP_KILL_CSPRNG

ecdh.h, [16](#)

ECP_PBKDF2

ecdh.h, [16](#)

ECP_PUBLIC_KEY_VALIDATE

ecdh.h, [16](#)

ECP_SP_DSA

ecdh.h, [17](#)

ECP_SVDP_DH

ecdh.h, [17](#)

ECP_VP_DSA

ecdh.h, [17](#)

EFS

ecdh.h, [13](#)

EGS

ecdh.h, [13](#)

ecdh.h, [11](#)

EAS, [12](#)

ECDH_ERROR, [12](#)

ECDH_INVALID_PUBLIC_KEY, [12](#)

ECDH_INVALID, [12](#)

ECDH_OK, [12](#)

ECP_AES_CBC_IV0_DECRYPT, [13](#)

ECP_AES_CBC_IV0_ENCRYPT, [13](#)

ECP_CREATE_CSPRNG, [13](#)

ECP_ECIES_DECRYPT, [14](#)

ECP_ECIES_ENCRYPT, [14](#)

ECP_HASH, [15](#)

ECP_HMAC, [15](#)

ECP_KDF2, [15](#)

ECP_KEY_PAIR_GENERATE, [15](#)

ECP_KILL_CSPRNG, [16](#)

ECP_PBKDF2, [16](#)

ECP_PUBLIC_KEY_VALIDATE, [16](#)

ECP_SP_DSA, [17](#)

ECP_SVDP_DH, [17](#)

ECP_VP_DSA, [17](#)

EFS, [13](#)

EGS, [13](#)

HASH_BYTES

mpin.h, [20](#)

MAXPIN

mpin.h, [20](#)

MPIN_AES_GCM_DECRYPT

mpin.h, [21](#)

MPIN_AES_GCM_ENCRYPT

mpin.h, [21](#)

MPIN_BAD_PIN

mpin.h, [20](#)

MPIN_CLIENT_1

mpin.h, [22](#)

MPIN_CLIENT_2

mpin.h, [23](#)

MPIN_CLIENT_KEY

mpin.h, [23](#)

MPIN_CLIENT

mpin.h, [21](#)

MPIN_CREATE_CSPRNG

mpin.h, [23](#)

MPIN_DECODING

mpin.h, [24](#)

MPIN_ENCODING

mpin.h, [24](#)

MPIN_EXTRACT_PIN

mpin.h, [24](#)

MPIN_GET_CLIENT_PERMIT

mpin.h, [24](#)

MPIN_GET_CLIENT_SECRET

mpin.h, [25](#)

MPIN_GET_G1_MULTIPLE
 mpin.h, 25
 MPIN_GET_SERVER_SECRET
 mpin.h, 25
 MPIN_GET_TIME
 mpin.h, 26
 MPIN_GET_Y
 mpin.h, 26
 MPIN_HASH_ALL
 mpin.h, 26
 MPIN_HASH_ID
 mpin.h, 26
 MPIN_HMAC
 mpin.h, 27
 MPIN_INVALID_POINT
 mpin.h, 20
 MPIN_KANGAROO
 mpin.h, 27
 MPIN_KILL_CSPRNG
 mpin.h, 27
 MPIN_OK
 mpin.h, 20
 MPIN_PBKDF2
 mpin.h, 28
 MPIN_PRECOMPUTE
 mpin.h, 28
 MPIN_RANDOM_GENERATE
 mpin.h, 28
 MPIN_RECOMBINE_G1
 mpin.h, 29
 MPIN_RECOMBINE_G2
 mpin.h, 29
 MPIN_SERVER_1
 mpin.h, 30
 MPIN_SERVER_2
 mpin.h, 30
 MPIN_SERVER_KEY
 mpin.h, 31
 MPIN_SERVER
 mpin.h, 29
 MPIN_today
 mpin.h, 31
 mpin.h, 18
 HASH_BYTES, 20
 MAXPIN, 20
 MPIN_AES_GCM_DECRYPT, 21
 MPIN_AES_GCM_ENCRYPT, 21
 MPIN_BAD_PIN, 20
 MPIN_CLIENT_1, 22
 MPIN_CLIENT_2, 23
 MPIN_CLIENT_KEY, 23
 MPIN_CLIENT, 21
 MPIN_CREATE_CSPRNG, 23
 MPIN_DECODING, 24
 MPIN_ENCODING, 24
 MPIN_EXTRACT_PIN, 24
 MPIN_GET_CLIENT_PERMIT, 24
 MPIN_GET_CLIENT_SECRET, 25
 MPIN_GET_G1_MULTIPLE, 25
 MPIN_GET_SERVER_SECRET, 25
 MPIN_GET_TIME, 26
 MPIN_GET_Y, 26
 MPIN_HASH_ALL, 26
 MPIN_HASH_ID, 26
 MPIN_HMAC, 27
 MPIN_INVALID_POINT, 20
 MPIN_KANGAROO, 27
 MPIN_KILL_CSPRNG, 27
 MPIN_OK, 20
 MPIN_PBKDF2, 28
 MPIN_PRECOMPUTE, 28
 MPIN_RANDOM_GENERATE, 28
 MPIN_RECOMBINE_G1, 29
 MPIN_RECOMBINE_G2, 29
 MPIN_SERVER_1, 30
 MPIN_SERVER_2, 30
 MPIN_SERVER_KEY, 31
 MPIN_SERVER, 29
 MPIN_today, 31
 PAS, 20
 PBLN, 20
 PFS, 20
 PGS, 21
 TIME_SLOT_MINUTES, 21
 PAS
 mpin.h, 20
 PBLN
 mpin.h, 20
 PFS
 mpin.h, 20
 PGS
 mpin.h, 21
 RFS
 rsa.h, 32
 RSA_CREATE_CSPRNG
 rsa.h, 32
 RSA_DECRYPT
 rsa.h, 33
 RSA_ENCRYPT
 rsa.h, 33
 RSA_KEY_PAIR
 rsa.h, 33
 RSA_KILL_CSPRNG
 rsa.h, 33
 RSA_OAEP_DECODE
 rsa.h, 34
 RSA_OAEP_ENCODE
 rsa.h, 34
 RSA_PRIVATE_KEY_KILL
 rsa.h, 34
 rsa.h, 31
 RFS, 32
 RSA_CREATE_CSPRNG, 32
 RSA_DECRYPT, 33
 RSA_ENCRYPT, 33

- RSA_KEY_PAIR, [33](#)
- RSA_KILL_CSPRNG, [33](#)
- RSA_OAEP_DECODE, [34](#)
- RSA_OAEP_ENCODE, [34](#)
- RSA_PRIVATE_KEY_KILL, [34](#)
- WCC_RECEIVER_KEY, [41](#)
- WCC_RECOMBINE_G1, [41](#)
- WCC_RECOMBINE_G2, [41](#)
- WCC_SENDER_KEY, [42](#)
- WCC_today, [42](#)
- TIME_SLOT_MINUTES
 - mpin.h, [21](#)
- WCC_AES_GCM_DECRYPT
 - wcc.c, [36](#)
- WCC_AES_GCM_ENCRYPT
 - wcc.c, [36](#)
- WCC_CREATE_CSPRNG
 - wcc.c, [37](#)
- WCC_GET_G1_MULTIPLE
 - wcc.c, [37](#)
- WCC_GET_G1_PERMIT
 - wcc.c, [37](#)
- WCC_GET_G1_TPMULT
 - wcc.c, [38](#)
- WCC_GET_G2_MULTIPLE
 - wcc.c, [38](#)
- WCC_GET_G2_PERMIT
 - wcc.c, [38](#)
- WCC_GET_G2_TPMULT
 - wcc.c, [39](#)
- WCC_HASH_ID
 - wcc.c, [39](#)
- WCC_Hq
 - wcc.c, [40](#)
- WCC_KILL_CSPRNG
 - wcc.c, [40](#)
- WCC_RANDOM_GENERATE
 - wcc.c, [40](#)
- WCC_RECEIVER_KEY
 - wcc.c, [41](#)
- WCC_RECOMBINE_G1
 - wcc.c, [41](#)
- WCC_RECOMBINE_G2
 - wcc.c, [41](#)
- WCC_SENDER_KEY
 - wcc.c, [42](#)
- WCC_today
 - wcc.c, [42](#)
- wcc.c, [35](#)
 - WCC_AES_GCM_DECRYPT, [36](#)
 - WCC_AES_GCM_ENCRYPT, [36](#)
 - WCC_CREATE_CSPRNG, [37](#)
 - WCC_GET_G1_MULTIPLE, [37](#)
 - WCC_GET_G1_PERMIT, [37](#)
 - WCC_GET_G1_TPMULT, [38](#)
 - WCC_GET_G2_MULTIPLE, [38](#)
 - WCC_GET_G2_PERMIT, [38](#)
 - WCC_GET_G2_TPMULT, [39](#)
 - WCC_HASH_ID, [39](#)
 - WCC_Hq, [40](#)
 - WCC_KILL_CSPRNG, [40](#)
 - WCC_RANDOM_GENERATE, [40](#)