AMCL

Generated by Doxygen 1.8.11

Contents

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

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:

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 ciphtertext.
- int ECP_AES_CBC_IV0_DECRYPT (octet *K, octet *C, octet *P)

AES encrypts a plaintext to a ciphtertext.

- 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 ciphtertext.

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

Parameters

K	AES key
C	input ciphertext octet
P	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 ciphtertext.

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

K	AES key
P	input plaintext octet
C	output ciphertext octet

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

Initialise a random number generator.

Parameters

R	is a pointer to a cryptographically secure random number generator
S	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

P1	input Key Derivation parameters
P2	input Encoding parameters
V	component of the input ciphertext
C	the input ciphertext
T	the input HMAC tag, part of the ciphertext
U	the input private key for decryption
M	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

P1	input Key Derivation parameters
P2	input Encoding parameters
R	is a pointer to a cryptographically secure random number generator
W	the input public key of the recieving party
M	is the plaintext message to be encrypted
len	the length of the HMAC tag
V	component of the output ciphertext
C	the output ciphertext
T	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	input octet
O	output octet - H(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

M	input message octet
K	input encryption key
len	is output desired length of HMAC tag
tag	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

Z	input octet
P	input key derivation parameters - can be NULL
len	is output desired length of key
K	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

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

R	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

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

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

Validate an ECC public key.

Parameters

f	if = 0 just does some simple checks, else tests that W is of the correct order
W	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

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

s	is the input private key,
W	the input public key of the other party
K	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

W	the input public key
M	the input message
c	component of the input signature
d	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 PBLEN 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 unsign32 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 unsign32 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 *I, 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 PBLEN 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

K	AES key
IV	Initialization vector
H	Header
P	Plaintext
C	Ciphertext
T	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

K	AES key
IV	Initialization vector
H	Header
P	Plaintext
C	Ciphertext
T	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!=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!=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

x	an input, a locally generated random number
y	an input random challenge from the server
V	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

g1	precomputed input
g2	precomputed input
pin	is the input PIN number
r	is an input, a locally generated random number
x	is an input, a locally generated random number
p	is an input, hash of the protocol transcript
T	is the input Server-side Diffie-Hellman component
K	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

R	is a pointer to a cryptographically secure random number generator
S	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

TP	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

R	is a pointer to a cryptographically secure random number generator
TP	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

ID	is the input client identity
pin	is an input PIN number
CS	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

d	is input date, in days since the epoch.
S	is an input master secret
ID	is the input client identity
TP	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

S	is an input master secret
ID	is the input client identity
CS	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

R	is a pointer to a cryptographically secure random number generator
type	determines type of action to be taken
x	an output internally randomly generated if R!=NULL, otherwise must be provided as an input
G	if type=0 a point in G1, else an octet to be mapped to G1
W	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

S	is an input master secret
SS	is the server secret = s.Q where Q is a fixed generator of G2

Returns

0 or an error code

6.2.3.15 DLL_EXPORT unsign32 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

t	is epoch time in seconds
O	is an input octet
Y	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	is the hashed input client ID = H(ID)
U	is the client output = x.H(ID)
CU	is the client output = x.(H(ID)+H(T|H(ID)))
Y	is the server challenge
V	is the client part response
R	is the client part response
W	is the server part response
H	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

ID	an octet containing the identity
HID	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

M	input message octet
K	input encryption key
len	is output desired length of HMAC tag
tag	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

E	a member of the group GT
F	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

R	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

P	input password
S	input salt
rep	Number of times to be iterated.
len	is output desired length of key
K	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

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

Returns

0 or an error code

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

Generate a random group element.

Parameters

R	is a pointer to a cryptographically secure random number generator
S	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

Q1	an input member of G1
Q2	an input member of G1
Q	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

P1	an input member of G2
P2	an input member of G2
P	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

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

Parameters

SS	is the input server secret
U	is input from the client = x.H(ID)
UT	is input from the client= x.(H(ID)+H(d|H(ID)))
V	is an input from the client
E	is an output to help the Kangaroos to find the PIN error, or NULL if not required
F	is an output to help the Kangaroos to find the PIN error, or NULL if not required
ID	is the input claimed client identity
MESSAGE	is the message to be signed
t	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

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

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

Parameters

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

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

Returns

0 or an error code

6.2.3.31 DLL_EXPORT unsign32 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

R	is a pointer to a cryptographically secure random number generator
S	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↩ RIV	the input RSA private key
G	is the input ciphertext
F	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

PUB	the input RSA public key
F	is input padded message
G	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

R	is a pointer to a cryptographically secure random number generator
e	the encryption exponent
P↩ RIV	the output RSA private key
PUB	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

R	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

P	are input encoding parameter string (could be NULL)
F	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

M	is the input message
R	is a pointer to a cryptographically secure random number generator
P	are input encoding parameter string (could be NULL)
F	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

P↩ RIV	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 *Pg↩ G1Oct, 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.
- unsign32 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

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

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

RNG	cryptographically secure random number generator
SEED	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

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

Returns

rtn 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

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

Returns

rtn 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

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

Returns

rtn 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

hashDone	ID is value is already hashed if set to 1
S	integer modulus curve order
ID	ID Value or sha256(ID)
VG2	EC Point VG2 = s∗H2(ID)

Returns

rtn 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

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

Returns

rtn 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

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

Returns

rtn 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

ID	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 = toInteger(sha256(A,B,C,D))
2. h = x % q where q is the curve order

Parameters

A	EC Point
B	EC Point
C	EC Point
D	Identity

Returns

h Integer

6.4.2.12 void WCC_KILL_CSPRNG (csprng * *RNG*)

Kill a random number generator.

Deletes all internal state

Parameters

RNG	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

RNG	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(pia.AG1+PaG1,(y+pib).BKeyG2)
2. K=H(j,w.PaG1)

Parameters

date	Epoch days
yOct	Random y $<$ q where q is the curve order
wOct	Random w $<$ q where q is the curve order
piaOct	Hq(PaG1,PbG2,PgG1)
pibOct	Hq(PbG2,PaG1,PgG1)
PaG1Oct	x.AG1 where x $<$ q
PgG1Oct	w.AG1 where w $<$ q
BKeyG2Oct	Receiver key
BTPG2Oct	Receiver time permit
IdAOct	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

R1	member of G1
R2	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

W1	member of G2
W2	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

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

Returns

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

6.4.2.18 unsign32 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