*\*\**

*\* \file rsa.h*

*\**

*\* \brief This file provides an API for the RSA public-key cryptosystem.*

*\**

*\* The RSA public-key cryptosystem is defined in* **<em>***Public-Key*

*\* Cryptography Standards (PKCS) #1 v1.5: RSA Encryption***</em>**

*\* and* **<em>***Public-Key Cryptography Standards (PKCS) #1 v2.1:*

*\* RSA Cryptography Specifications***</em>***.*

*\**

*\*/*

*/\**

*\* Copyright (C) 2006-2018, Arm Limited (or its affiliates), All Rights Reserved*

*\* SPDX-License-Identifier: Apache-2.0*

*\**

*\* Licensed 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.*

*\**

*\* This file is part of Mbed TLS (*[https://tls.mbed.org](https://tls.mbed.org/)*)*

*\*/*

**#ifndef** MBEDTLS\_RSA\_H

**#define** MBEDTLS\_RSA\_H

**#if** !**defined**(MBEDTLS\_CONFIG\_FILE)

**#include** "config.h"

**#else**

**#include** MBEDTLS\_CONFIG\_FILE

**#endif**

**#include** "bignum.h"

**#include** "md.h"

**#if** **defined**(MBEDTLS\_THREADING\_C)

**#include** "threading.h"

**#endif**

*/\**

*\* RSA Error codes*

*\*/*

**#define** MBEDTLS\_ERR\_RSA\_BAD\_INPUT\_DATA -0x4080 */\*\*< Bad input parameters to function. \*/*

**#define** MBEDTLS\_ERR\_RSA\_INVALID\_PADDING -0x4100 */\*\*< Input data contains invalid padding and is rejected. \*/*

**#define** MBEDTLS\_ERR\_RSA\_KEY\_GEN\_FAILED -0x4180 */\*\*< Something failed during generation of a key. \*/*

**#define** MBEDTLS\_ERR\_RSA\_KEY\_CHECK\_FAILED -0x4200 */\*\*< Key failed to pass the validity check of the library. \*/*

**#define** MBEDTLS\_ERR\_RSA\_PUBLIC\_FAILED -0x4280 */\*\*< The public key operation failed. \*/*

**#define** MBEDTLS\_ERR\_RSA\_PRIVATE\_FAILED -0x4300 */\*\*< The private key operation failed. \*/*

**#define** MBEDTLS\_ERR\_RSA\_VERIFY\_FAILED -0x4380 */\*\*< The PKCS#1 verification failed. \*/*

**#define** MBEDTLS\_ERR\_RSA\_OUTPUT\_TOO\_LARGE -0x4400 */\*\*< The output buffer for decryption is not large enough. \*/*

**#define** MBEDTLS\_ERR\_RSA\_RNG\_FAILED -0x4480 */\*\*< The random generator failed to generate non-zeros. \*/*

*/\* MBEDTLS\_ERR\_RSA\_UNSUPPORTED\_OPERATION is deprecated and should not be used.*

*\*/*

**#define** MBEDTLS\_ERR\_RSA\_UNSUPPORTED\_OPERATION -0x4500 */\*\*< The implementation does not offer the requested operation, for example, because of security violations or lack of functionality. \*/*

*/\* MBEDTLS\_ERR\_RSA\_HW\_ACCEL\_FAILED is deprecated and should not be used. \*/*

**#define** MBEDTLS\_ERR\_RSA\_HW\_ACCEL\_FAILED -0x4580 */\*\*< RSA hardware accelerator failed. \*/*

*/\**

*\* RSA constants*

*\*/*

**#define** MBEDTLS\_RSA\_PUBLIC 0 */\*\*< Request private key operation. \*/*

**#define** MBEDTLS\_RSA\_PRIVATE 1 */\*\*< Request public key operation. \*/*

**#define** MBEDTLS\_RSA\_PKCS\_V15 0 */\*\*< Use PKCS#1 v1.5 encoding. \*/*

**#define** MBEDTLS\_RSA\_PKCS\_V21 1 */\*\*< Use PKCS#1 v2.1 encoding. \*/*

**#define** MBEDTLS\_RSA\_SIGN 1 */\*\*< Identifier for RSA signature operations. \*/*

**#define** MBEDTLS\_RSA\_CRYPT 2 */\*\*< Identifier for RSA encryption and decryption operations. \*/*

**#define** MBEDTLS\_RSA\_SALT\_LEN\_ANY -1

*/\**

*\* The above constants may be used even if the RSA module is compile out,*

*\* eg for alternative (PKCS#11) RSA implemenations in the PK layers.*

*\*/*

**#ifdef** \_\_cplusplus

**extern** "C" {

**#endif**

**#if** !**defined**(MBEDTLS\_RSA\_ALT)

*// Regular implementation*

*//*

*/\*\**

*\* \brief The RSA context structure.*

*\**

*\* \note Direct manipulation of the members of this structure*

*\* is deprecated. All manipulation should instead be done through*

*\* the public interface functions.*

*\*/*

**typedef** **struct** mbedtls\_rsa\_context

{

int ver; */\*!< Always 0.\*/*

size\_t len; */\*!< The size of \p N in Bytes. \*/*

mbedtls\_mpi N; */\*!< The public modulus. \*/*

mbedtls\_mpi E; */\*!< The public exponent. \*/*

mbedtls\_mpi D; */\*!< The private exponent. \*/*

mbedtls\_mpi P; */\*!< The first prime factor. \*/*

mbedtls\_mpi Q; */\*!< The second prime factor. \*/*

mbedtls\_mpi DP; */\*!< <code>D % (P - 1)</code>. \*/*

mbedtls\_mpi DQ; */\*!< <code>D % (Q - 1)</code>. \*/*

mbedtls\_mpi QP; */\*!< <code>1 / (Q % P)</code>. \*/*

mbedtls\_mpi RN; */\*!< cached <code>R^2 mod N</code>. \*/*

mbedtls\_mpi RP; */\*!< cached <code>R^2 mod P</code>. \*/*

mbedtls\_mpi RQ; */\*!< cached <code>R^2 mod Q</code>. \*/*

mbedtls\_mpi Vi; */\*!< The cached blinding value. \*/*

mbedtls\_mpi Vf; */\*!< The cached un-blinding value. \*/*

int padding; */\*!< Selects padding mode:*

*#MBEDTLS\_RSA\_PKCS\_V15 for 1.5 padding and*

*#MBEDTLS\_RSA\_PKCS\_V21 for OAEP or PSS. \*/*

int hash\_id; */\*!< Hash identifier of mbedtls\_md\_type\_t type,*

*as specified in md.h for use in the MGF*

*mask generating function used in the*

*EME-OAEP and EMSA-PSS encodings. \*/*

**#if** **defined**(MBEDTLS\_THREADING\_C)

mbedtls\_threading\_mutex\_t mutex; */\*!< Thread-safety mutex. \*/*

**#endif**

}

mbedtls\_rsa\_context;

**#else** */\* MBEDTLS\_RSA\_ALT \*/*

**#include** "rsa\_alt.h"

**#endif** */\* MBEDTLS\_RSA\_ALT \*/*

*/\*\**

*\* \brief This function initializes an RSA context.*

*\**

*\* \note Set padding to #MBEDTLS\_RSA\_PKCS\_V21 for the RSAES-OAEP*

*\* encryption scheme and the RSASSA-PSS signature scheme.*

*\**

*\* \note The \p hash\_id parameter is ignored when using*

*\* #MBEDTLS\_RSA\_PKCS\_V15 padding.*

*\**

*\* \note The choice of padding mode is strictly enforced for private key*

*\* operations, since there might be security concerns in*

*\* mixing padding modes. For public key operations it is*

*\* a default value, which can be overriden by calling specific*

*\* \c rsa\_rsaes\_xxx or \c rsa\_rsassa\_xxx functions.*

*\**

*\* \note The hash selected in \p hash\_id is always used for OEAP*

*\* encryption. For PSS signatures, it is always used for*

*\* making signatures, but can be overriden for verifying them.*

*\* If set to #MBEDTLS\_MD\_NONE, it is always overriden.*

*\**

*\* \param ctx The RSA context to initialize.*

*\* \param padding Selects padding mode: #MBEDTLS\_RSA\_PKCS\_V15 or*

*\* #MBEDTLS\_RSA\_PKCS\_V21.*

*\* \param hash\_id The hash identifier of #mbedtls\_md\_type\_t type, if*

*\* \p padding is #MBEDTLS\_RSA\_PKCS\_V21.*

*\*/*

void **mbedtls\_rsa\_init**( mbedtls\_rsa\_context \*ctx,

int padding,

int hash\_id);

*/\*\**

*\* \brief This function imports a set of core parameters into an*

*\* RSA context.*

*\**

*\* \note This function can be called multiple times for successive*

*\* imports, if the parameters are not simultaneously present.*

*\**

*\* Any sequence of calls to this function should be followed*

*\* by a call to mbedtls\_rsa\_complete(), which checks and*

*\* completes the provided information to a ready-for-use*

*\* public or private RSA key.*

*\**

*\* \note See mbedtls\_rsa\_complete() for more information on which*

*\* parameters are necessary to set up a private or public*

*\* RSA key.*

*\**

*\* \note The imported parameters are copied and need not be preserved*

*\* for the lifetime of the RSA context being set up.*

*\**

*\* \param ctx The initialized RSA context to store the parameters in.*

*\* \param N The RSA modulus, or NULL.*

*\* \param P The first prime factor of \p N, or NULL.*

*\* \param Q The second prime factor of \p N, or NULL.*

*\* \param D The private exponent, or NULL.*

*\* \param E The public exponent, or NULL.*

*\**

*\* \return \c 0 on success.*

*\* \return A non-zero error code on failure.*

*\*/*

int **mbedtls\_rsa\_import**( mbedtls\_rsa\_context \*ctx,

**const** mbedtls\_mpi \*N,

**const** mbedtls\_mpi \*P, **const** mbedtls\_mpi \*Q,

**const** mbedtls\_mpi \*D, **const** mbedtls\_mpi \*E );

*/\*\**

*\* \brief This function imports core RSA parameters, in raw big-endian*

*\* binary format, into an RSA context.*

*\**

*\* \note This function can be called multiple times for successive*

*\* imports, if the parameters are not simultaneously present.*

*\**

*\* Any sequence of calls to this function should be followed*

*\* by a call to mbedtls\_rsa\_complete(), which checks and*

*\* completes the provided information to a ready-for-use*

*\* public or private RSA key.*

*\**

*\* \note See mbedtls\_rsa\_complete() for more information on which*

*\* parameters are necessary to set up a private or public*

*\* RSA key.*

*\**

*\* \note The imported parameters are copied and need not be preserved*

*\* for the lifetime of the RSA context being set up.*

*\**

*\* \param ctx The initialized RSA context to store the parameters in.*

*\* \param N The RSA modulus, or NULL.*

*\* \param N\_len The Byte length of \p N, ignored if \p N == NULL.*

*\* \param P The first prime factor of \p N, or NULL.*

*\* \param P\_len The Byte length of \p P, ignored if \p P == NULL.*

*\* \param Q The second prime factor of \p N, or NULL.*

*\* \param Q\_len The Byte length of \p Q, ignored if \p Q == NULL.*

*\* \param D The private exponent, or NULL.*

*\* \param D\_len The Byte length of \p D, ignored if \p D == NULL.*

*\* \param E The public exponent, or NULL.*

*\* \param E\_len The Byte length of \p E, ignored if \p E == NULL.*

*\**

*\* \return \c 0 on success.*

*\* \return A non-zero error code on failure.*

*\*/*

int **mbedtls\_rsa\_import\_raw**( mbedtls\_rsa\_context \*ctx,

unsigned char **const** \*N, size\_t N\_len,

unsigned char **const** \*P, size\_t P\_len,

unsigned char **const** \*Q, size\_t Q\_len,

unsigned char **const** \*D, size\_t D\_len,

unsigned char **const** \*E, size\_t E\_len );

*/\*\**

*\* \brief This function completes an RSA context from*

*\* a set of imported core parameters.*

*\**

*\* To setup an RSA public key, precisely \p N and \p E*

*\* must have been imported.*

*\**

*\* To setup an RSA private key, sufficient information must*

*\* be present for the other parameters to be derivable.*

*\**

*\* The default implementation supports the following:*

*\** **<ul><li>***Derive \p P, \p Q from \p N, \p D, \p E.***</li>**

*\** **<li>***Derive \p N, \p D from \p P, \p Q, \p E.***</li></ul>**

*\* Alternative implementations need not support these.*

*\**

*\* If this function runs successfully, it guarantees that*

*\* the RSA context can be used for RSA operations without*

*\* the risk of failure or crash.*

*\**

*\* \warning This function need not perform consistency checks*

*\* for the imported parameters. In particular, parameters that*

*\* are not needed by the implementation might be silently*

*\* discarded and left unchecked. To check the consistency*

*\* of the key material, see mbedtls\_rsa\_check\_privkey().*

*\**

*\* \param ctx The initialized RSA context holding imported parameters.*

*\**

*\* \return \c 0 on success.*

*\* \return #MBEDTLS\_ERR\_RSA\_BAD\_INPUT\_DATA if the attempted derivations*

*\* failed.*

*\**

*\*/*

int **mbedtls\_rsa\_complete**( mbedtls\_rsa\_context \*ctx );

*/\*\**

*\* \brief This function exports the core parameters of an RSA key.*

*\**

*\* If this function runs successfully, the non-NULL buffers*

*\* pointed to by \p N, \p P, \p Q, \p D, and \p E are fully*

*\* written, with additional unused space filled leading by*

*\* zero Bytes.*

*\**

*\* Possible reasons for returning*

*\* #MBEDTLS\_ERR\_PLATFORM\_FEATURE\_UNSUPPORTED:***<ul>**

*\** **<li>***An alternative RSA implementation is in use, which*

*\* stores the key externally, and either cannot or should*

*\* not export it into RAM.***</li>**

*\** **<li>***A SW or HW implementation might not support a certain*

*\* deduction. For example, \p P, \p Q from \p N, \p D,*

*\* and \p E if the former are not part of the*

*\* implementation.***</li></ul>**

*\**

*\* If the function fails due to an unsupported operation,*

*\* the RSA context stays intact and remains usable.*

*\**

*\* \param ctx The initialized RSA context.*

*\* \param N The MPI to hold the RSA modulus, or NULL.*

*\* \param P The MPI to hold the first prime factor of \p N, or NULL.*

*\* \param Q The MPI to hold the second prime factor of \p N, or NULL.*

*\* \param D The MPI to hold the private exponent, or NULL.*

*\* \param E The MPI to hold the public exponent, or NULL.*

*\**

*\* \return \c 0 on success.*

*\* \return #MBEDTLS\_ERR\_PLATFORM\_FEATURE\_UNSUPPORTED if exporting the*

*\* requested parameters cannot be done due to missing*

*\* functionality or because of security policies.*

*\* \return A non-zero return code on any other failure.*

*\**

*\*/*

int **mbedtls\_rsa\_export**( **const** mbedtls\_rsa\_context \*ctx,

mbedtls\_mpi \*N, mbedtls\_mpi \*P, mbedtls\_mpi \*Q,

mbedtls\_mpi \*D, mbedtls\_mpi \*E );

*/\*\**

*\* \brief This function exports core parameters of an RSA key*

*\* in raw big-endian binary format.*

*\**

*\* If this function runs successfully, the non-NULL buffers*

*\* pointed to by \p N, \p P, \p Q, \p D, and \p E are fully*

*\* written, with additional unused space filled leading by*

*\* zero Bytes.*

*\**

*\* Possible reasons for returning*

*\* #MBEDTLS\_ERR\_PLATFORM\_FEATURE\_UNSUPPORTED:***<ul>**

*\** **<li>***An alternative RSA implementation is in use, which*

*\* stores the key externally, and either cannot or should*

*\* not export it into RAM.***</li>**

*\** **<li>***A SW or HW implementation might not support a certain*

*\* deduction. For example, \p P, \p Q from \p N, \p D,*

*\* and \p E if the former are not part of the*

*\* implementation.***</li></ul>**

*\* If the function fails due to an unsupported operation,*

*\* the RSA context stays intact and remains usable.*

*\**

*\* \note The length parameters are ignored if the corresponding*

*\* buffer pointers are NULL.*

*\**

*\* \param ctx The initialized RSA context.*

*\* \param N The Byte array to store the RSA modulus, or NULL.*

*\* \param N\_len The size of the buffer for the modulus.*

*\* \param P The Byte array to hold the first prime factor of \p N, or*

*\* NULL.*

*\* \param P\_len The size of the buffer for the first prime factor.*

*\* \param Q The Byte array to hold the second prime factor of \p N, or*

*\* NULL.*

*\* \param Q\_len The size of the buffer for the second prime factor.*

*\* \param D The Byte array to hold the private exponent, or NULL.*

*\* \param D\_len The size of the buffer for the private exponent.*

*\* \param E The Byte array to hold the public exponent, or NULL.*

*\* \param E\_len The size of the buffer for the public exponent.*

*\**

*\* \return \c 0 on success.*

*\* \return #MBEDTLS\_ERR\_PLATFORM\_FEATURE\_UNSUPPORTED if exporting the*

*\* requested parameters cannot be done due to missing*

*\* functionality or because of security policies.*

*\* \return A non-zero return code on any other failure.*

*\*/*

int **mbedtls\_rsa\_export\_raw**( **const** mbedtls\_rsa\_context \*ctx,

unsigned char \*N, size\_t N\_len,

unsigned char \*P, size\_t P\_len,

unsigned char \*Q, size\_t Q\_len,

unsigned char \*D, size\_t D\_len,

unsigned char \*E, size\_t E\_len );

*/\*\**

*\* \brief This function exports CRT parameters of a private RSA key.*

*\**

*\* \note Alternative RSA implementations not using CRT-parameters*

*\* internally can implement this function based on*

*\* mbedtls\_rsa\_deduce\_opt().*

*\**

*\* \param ctx The initialized RSA context.*

*\* \param DP The MPI to hold D modulo P-1, or NULL.*

*\* \param DQ The MPI to hold D modulo Q-1, or NULL.*

*\* \param QP The MPI to hold modular inverse of Q modulo P, or NULL.*

*\**

*\* \return \c 0 on success.*

*\* \return A non-zero error code on failure.*

*\**

*\*/*

int **mbedtls\_rsa\_export\_crt**( **const** mbedtls\_rsa\_context \*ctx,

mbedtls\_mpi \*DP, mbedtls\_mpi \*DQ, mbedtls\_mpi \*QP );

*/\*\**

*\* \brief This function sets padding for an already initialized RSA*

*\* context. See mbedtls\_rsa\_init() for details.*

*\**

*\* \param ctx The RSA context to be set.*

*\* \param padding Selects padding mode: #MBEDTLS\_RSA\_PKCS\_V15 or*

*\* #MBEDTLS\_RSA\_PKCS\_V21.*

*\* \param hash\_id The #MBEDTLS\_RSA\_PKCS\_V21 hash identifier.*

*\*/*

void **mbedtls\_rsa\_set\_padding**( mbedtls\_rsa\_context \*ctx, int padding,

int hash\_id);

*/\*\**

*\* \brief This function retrieves the length of RSA modulus in Bytes.*

*\**

*\* \param ctx The initialized RSA context.*

*\**

*\* \return The length of the RSA modulus in Bytes.*

*\**

*\*/*

size\_t **mbedtls\_rsa\_get\_len**( **const** mbedtls\_rsa\_context \*ctx );

*/\*\**

*\* \brief This function generates an RSA keypair.*

*\**

*\* \note mbedtls\_rsa\_init() must be called before this function,*

*\* to set up the RSA context.*

*\**

*\* \param ctx The RSA context used to hold the key.*

*\* \param f\_rng The RNG function.*

*\* \param p\_rng The RNG context.*

*\* \param nbits The size of the public key in bits.*

*\* \param exponent The public exponent. For example, 65537.*

*\**

*\* \return \c 0 on success.*

*\* \return An \c MBEDTLS\_ERR\_RSA\_XXX error code on failure.*

*\*/*

int **mbedtls\_rsa\_gen\_key**( mbedtls\_rsa\_context \*ctx,

int (\*f\_rng)(void \*, unsigned char \*, size\_t),

void \*p\_rng,

unsigned int nbits, int exponent );

*/\*\**

*\* \brief This function checks if a context contains at least an RSA*

*\* public key.*

*\**

*\* If the function runs successfully, it is guaranteed that*

*\* enough information is present to perform an RSA public key*

*\* operation using mbedtls\_rsa\_public().*

*\**

*\* \param ctx The RSA context to check.*

*\**

*\* \return \c 0 on success.*

*\* \return An \c MBEDTLS\_ERR\_RSA\_XXX error code on failure.*

*\**

*\*/*

int **mbedtls\_rsa\_check\_pubkey**( **const** mbedtls\_rsa\_context \*ctx );

*/\*\**

*\* \brief This function checks if a context contains an RSA private key*

*\* and perform basic consistency checks.*

*\**

*\* \note The consistency checks performed by this function not only*

*\* ensure that mbedtls\_rsa\_private() can be called successfully*

*\* on the given context, but that the various parameters are*

*\* mutually consistent with high probability, in the sense that*

*\* mbedtls\_rsa\_public() and mbedtls\_rsa\_private() are inverses.*

*\**

*\* \warning This function should catch accidental misconfigurations*

*\* like swapping of parameters, but it cannot establish full*

*\* trust in neither the quality nor the consistency of the key*

*\* material that was used to setup the given RSA context:*

*\** **<ul><li>***Consistency: Imported parameters that are irrelevant*

*\* for the implementation might be silently dropped. If dropped,*

*\* the current function does not have access to them,*

*\* and therefore cannot check them. See mbedtls\_rsa\_complete().*

*\* If you want to check the consistency of the entire*

*\* content of an PKCS1-encoded RSA private key, for example, you*

*\* should use mbedtls\_rsa\_validate\_params() before setting*

*\* up the RSA context.*

*\* Additionally, if the implementation performs empirical checks,*

*\* these checks substantiate but do not guarantee consistency.***</li>**

*\** **<li>***Quality: This function is not expected to perform*

*\* extended quality assessments like checking that the prime*

*\* factors are safe. Additionally, it is the responsibility of the*

*\* user to ensure the trustworthiness of the source of his RSA*

*\* parameters, which goes beyond what is effectively checkable*

*\* by the library.***</li></ul>**

*\**

*\* \param ctx The RSA context to check.*

*\**

*\* \return \c 0 on success.*

*\* \return An \c MBEDTLS\_ERR\_RSA\_XXX error code on failure.*

*\*/*

int **mbedtls\_rsa\_check\_privkey**( **const** mbedtls\_rsa\_context \*ctx );

*/\*\**

*\* \brief This function checks a public-private RSA key pair.*

*\**

*\* It checks each of the contexts, and makes sure they match.*

*\**

*\* \param pub The RSA context holding the public key.*

*\* \param prv The RSA context holding the private key.*

*\**

*\* \return \c 0 on success.*

*\* \return An \c MBEDTLS\_ERR\_RSA\_XXX error code on failure.*

*\*/*

int **mbedtls\_rsa\_check\_pub\_priv**( **const** mbedtls\_rsa\_context \*pub,

**const** mbedtls\_rsa\_context \*prv );

*/\*\**

*\* \brief This function performs an RSA public key operation.*

*\**

*\* \note This function does not handle message padding.*

*\**

*\* \note Make sure to set \p input[0] = 0 or ensure that*

*\* input is smaller than \p N.*

*\**

*\* \note The input and output buffers must be large*

*\* enough. For example, 128 Bytes if RSA-1024 is used.*

*\**

*\* \param ctx The RSA context.*

*\* \param input The input buffer.*

*\* \param output The output buffer.*

*\**

*\* \return \c 0 on success.*

*\* \return An \c MBEDTLS\_ERR\_RSA\_XXX error code on failure.*

*\*/*

int **mbedtls\_rsa\_public**( mbedtls\_rsa\_context \*ctx,

**const** unsigned char \*input,

unsigned char \*output );

*/\*\**

*\* \brief This function performs an RSA private key operation.*

*\**

*\* \note The input and output buffers must be large*

*\* enough. For example, 128 Bytes if RSA-1024 is used.*

*\**

*\* \note Blinding is used if and only if a PRNG is provided.*

*\**

*\* \note If blinding is used, both the base of exponentation*

*\* and the exponent are blinded, providing protection*

*\* against some side-channel attacks.*

*\**

*\* \warning It is deprecated and a security risk to not provide*

*\* a PRNG here and thereby prevent the use of blinding.*

*\* Future versions of the library may enforce the presence*

*\* of a PRNG.*

*\**

*\* \param ctx The RSA context.*

*\* \param f\_rng The RNG function. Needed for blinding.*

*\* \param p\_rng The RNG context.*

*\* \param input The input buffer.*

*\* \param output The output buffer.*

*\**

*\* \return \c 0 on success.*

*\* \return An \c MBEDTLS\_ERR\_RSA\_XXX error code on failure.*

*\**

*\*/*

int **mbedtls\_rsa\_private**( mbedtls\_rsa\_context \*ctx,

int (\*f\_rng)(void \*, unsigned char \*, size\_t),

void \*p\_rng,

**const** unsigned char \*input,

unsigned char \*output );

*/\*\**

*\* \brief This function adds the message padding, then performs an RSA*

*\* operation.*

*\**

*\* It is the generic wrapper for performing a PKCS#1 encryption*

*\* operation using the \p mode from the context.*

*\**

*\* \note The input and output buffers must be as large as the size*

*\* of \p ctx->N. For example, 128 Bytes if RSA-1024 is used.*

*\**

*\* \deprecated It is deprecated and discouraged to call this function*

*\* in #MBEDTLS\_RSA\_PRIVATE mode. Future versions of the library*

*\* are likely to remove the \p mode argument and have it*

*\* implicitly set to #MBEDTLS\_RSA\_PUBLIC.*

*\**

*\* \note Alternative implementations of RSA need not support*

*\* mode being set to #MBEDTLS\_RSA\_PRIVATE and might instead*

*\* return #MBEDTLS\_ERR\_PLATFORM\_FEATURE\_UNSUPPORTED.*

*\**

*\* \param ctx The RSA context.*

*\* \param f\_rng The RNG function. Needed for padding, PKCS#1 v2.1*

*\* encoding, and #MBEDTLS\_RSA\_PRIVATE.*

*\* \param p\_rng The RNG context.*

*\* \param mode #MBEDTLS\_RSA\_PUBLIC or #MBEDTLS\_RSA\_PRIVATE.*

*\* \param ilen The length of the plaintext.*

*\* \param input The buffer holding the data to encrypt.*

*\* \param output The buffer used to hold the ciphertext.*

*\**

*\* \return \c 0 on success.*

*\* \return An \c MBEDTLS\_ERR\_RSA\_XXX error code on failure.*

*\*/*

int **mbedtls\_rsa\_pkcs1\_encrypt**( mbedtls\_rsa\_context \*ctx,

int (\*f\_rng)(void \*, unsigned char \*, size\_t),

void \*p\_rng,

int mode, size\_t ilen,

**const** unsigned char \*input,

unsigned char \*output );

*/\*\**

*\* \brief This function performs a PKCS#1 v1.5 encryption operation*

*\* (RSAES-PKCS1-v1\_5-ENCRYPT).*

*\**

*\* \note The output buffer must be as large as the size*

*\* of \p ctx->N. For example, 128 Bytes if RSA-1024 is used.*

*\**

*\* \deprecated It is deprecated and discouraged to call this function*

*\* in #MBEDTLS\_RSA\_PRIVATE mode. Future versions of the library*

*\* are likely to remove the \p mode argument and have it*

*\* implicitly set to #MBEDTLS\_RSA\_PUBLIC.*

*\**

*\* \note Alternative implementations of RSA need not support*

*\* mode being set to #MBEDTLS\_RSA\_PRIVATE and might instead*

*\* return #MBEDTLS\_ERR\_PLATFORM\_FEATURE\_UNSUPPORTED.*

*\**

*\* \param ctx The RSA context.*

*\* \param f\_rng The RNG function. Needed for padding and*

*\* #MBEDTLS\_RSA\_PRIVATE.*

*\* \param p\_rng The RNG context.*

*\* \param mode #MBEDTLS\_RSA\_PUBLIC or #MBEDTLS\_RSA\_PRIVATE.*

*\* \param ilen The length of the plaintext.*

*\* \param input The buffer holding the data to encrypt.*

*\* \param output The buffer used to hold the ciphertext.*

*\**

*\* \return \c 0 on success.*

*\* \return An \c MBEDTLS\_ERR\_RSA\_XXX error code on failure.*

*\*/*

int **mbedtls\_rsa\_rsaes\_pkcs1\_v15\_encrypt**( mbedtls\_rsa\_context \*ctx,

int (\*f\_rng)(void \*, unsigned char \*, size\_t),

void \*p\_rng,

int mode, size\_t ilen,

**const** unsigned char \*input,

unsigned char \*output );

*/\*\**

*\* \brief This function performs a PKCS#1 v2.1 OAEP encryption*

*\* operation (RSAES-OAEP-ENCRYPT).*

*\**

*\* \note The output buffer must be as large as the size*

*\* of ctx->N. For example, 128 Bytes if RSA-1024 is used.*

*\**

*\* \deprecated It is deprecated and discouraged to call this function*

*\* in #MBEDTLS\_RSA\_PRIVATE mode. Future versions of the library*

*\* are likely to remove the \p mode argument and have it*

*\* implicitly set to #MBEDTLS\_RSA\_PUBLIC.*

*\**

*\* \note Alternative implementations of RSA need not support*

*\* mode being set to #MBEDTLS\_RSA\_PRIVATE and might instead*

*\* return #MBEDTLS\_ERR\_PLATFORM\_FEATURE\_UNSUPPORTED.*

*\**

*\* \param ctx The RSA context.*

*\* \param f\_rng The RNG function. Needed for padding and PKCS#1 v2.1*

*\* encoding and #MBEDTLS\_RSA\_PRIVATE.*

*\* \param p\_rng The RNG context.*

*\* \param mode #MBEDTLS\_RSA\_PUBLIC or #MBEDTLS\_RSA\_PRIVATE.*

*\* \param label The buffer holding the custom label to use.*

*\* \param label\_len The length of the label.*

*\* \param ilen The length of the plaintext.*

*\* \param input The buffer holding the data to encrypt.*

*\* \param output The buffer used to hold the ciphertext.*

*\**

*\* \return \c 0 on success.*

*\* \return An \c MBEDTLS\_ERR\_RSA\_XXX error code on failure.*

*\*/*

int **mbedtls\_rsa\_rsaes\_oaep\_encrypt**( mbedtls\_rsa\_context \*ctx,

int (\*f\_rng)(void \*, unsigned char \*, size\_t),

void \*p\_rng,

int mode,

**const** unsigned char \*label, size\_t label\_len,

size\_t ilen,

**const** unsigned char \*input,

unsigned char \*output );

*/\*\**

*\* \brief This function performs an RSA operation, then removes the*

*\* message padding.*

*\**

*\* It is the generic wrapper for performing a PKCS#1 decryption*

*\* operation using the \p mode from the context.*

*\**

*\* \note The output buffer length \c output\_max\_len should be*

*\* as large as the size \p ctx->len of \p ctx->N (for example,*

*\* 128 Bytes if RSA-1024 is used) to be able to hold an*

*\* arbitrary decrypted message. If it is not large enough to*

*\* hold the decryption of the particular ciphertext provided,*

*\* the function returns \c MBEDTLS\_ERR\_RSA\_OUTPUT\_TOO\_LARGE.*

*\**

*\* \note The input buffer must be as large as the size*

*\* of \p ctx->N. For example, 128 Bytes if RSA-1024 is used.*

*\**

*\* \deprecated It is deprecated and discouraged to call this function*

*\* in #MBEDTLS\_RSA\_PUBLIC mode. Future versions of the library*

*\* are likely to remove the \p mode argument and have it*

*\* implicitly set to #MBEDTLS\_RSA\_PRIVATE.*

*\**

*\* \note Alternative implementations of RSA need not support*

*\* mode being set to #MBEDTLS\_RSA\_PUBLIC and might instead*

*\* return #MBEDTLS\_ERR\_PLATFORM\_FEATURE\_UNSUPPORTED.*

*\**

*\* \param ctx The RSA context.*

*\* \param f\_rng The RNG function. Only needed for #MBEDTLS\_RSA\_PRIVATE.*

*\* \param p\_rng The RNG context.*

*\* \param mode #MBEDTLS\_RSA\_PUBLIC or #MBEDTLS\_RSA\_PRIVATE.*

*\* \param olen The length of the plaintext.*

*\* \param input The buffer holding the encrypted data.*

*\* \param output The buffer used to hold the plaintext.*

*\* \param output\_max\_len The maximum length of the output buffer.*

*\**

*\* \return \c 0 on success.*

*\* \return An \c MBEDTLS\_ERR\_RSA\_XXX error code on failure.*

*\*/*

int **mbedtls\_rsa\_pkcs1\_decrypt**( mbedtls\_rsa\_context \*ctx,

int (\*f\_rng)(void \*, unsigned char \*, size\_t),

void \*p\_rng,

int mode, size\_t \*olen,

**const** unsigned char \*input,

unsigned char \*output,

size\_t output\_max\_len );

*/\*\**

*\* \brief This function performs a PKCS#1 v1.5 decryption*

*\* operation (RSAES-PKCS1-v1\_5-DECRYPT).*

*\**

*\* \note The output buffer length \c output\_max\_len should be*

*\* as large as the size \p ctx->len of \p ctx->N, for example,*

*\* 128 Bytes if RSA-1024 is used, to be able to hold an*

*\* arbitrary decrypted message. If it is not large enough to*

*\* hold the decryption of the particular ciphertext provided,*

*\* the function returns #MBEDTLS\_ERR\_RSA\_OUTPUT\_TOO\_LARGE.*

*\**

*\* \note The input buffer must be as large as the size*

*\* of \p ctx->N. For example, 128 Bytes if RSA-1024 is used.*

*\**

*\* \deprecated It is deprecated and discouraged to call this function*

*\* in #MBEDTLS\_RSA\_PUBLIC mode. Future versions of the library*

*\* are likely to remove the \p mode argument and have it*

*\* implicitly set to #MBEDTLS\_RSA\_PRIVATE.*

*\**

*\* \note Alternative implementations of RSA need not support*

*\* mode being set to #MBEDTLS\_RSA\_PUBLIC and might instead*

*\* return #MBEDTLS\_ERR\_PLATFORM\_FEATURE\_UNSUPPORTED.*

*\**

*\* \param ctx The RSA context.*

*\* \param f\_rng The RNG function. Only needed for #MBEDTLS\_RSA\_PRIVATE.*

*\* \param p\_rng The RNG context.*

*\* \param mode #MBEDTLS\_RSA\_PUBLIC or #MBEDTLS\_RSA\_PRIVATE.*

*\* \param olen The length of the plaintext.*

*\* \param input The buffer holding the encrypted data.*

*\* \param output The buffer to hold the plaintext.*

*\* \param output\_max\_len The maximum length of the output buffer.*

*\**

*\* \return \c 0 on success.*

*\* \return An \c MBEDTLS\_ERR\_RSA\_XXX error code on failure.*

*\**

*\*/*

int **mbedtls\_rsa\_rsaes\_pkcs1\_v15\_decrypt**( mbedtls\_rsa\_context \*ctx,

int (\*f\_rng)(void \*, unsigned char \*, size\_t),

void \*p\_rng,

int mode, size\_t \*olen,

**const** unsigned char \*input,

unsigned char \*output,

size\_t output\_max\_len );

*/\*\**

*\* \brief This function performs a PKCS#1 v2.1 OAEP decryption*

*\* operation (RSAES-OAEP-DECRYPT).*

*\**

*\* \note The output buffer length \c output\_max\_len should be*

*\* as large as the size \p ctx->len of \p ctx->N, for*

*\* example, 128 Bytes if RSA-1024 is used, to be able to*

*\* hold an arbitrary decrypted message. If it is not*

*\* large enough to hold the decryption of the particular*

*\* ciphertext provided, the function returns*

*\* #MBEDTLS\_ERR\_RSA\_OUTPUT\_TOO\_LARGE.*

*\**

*\* \note The input buffer must be as large as the size*

*\* of \p ctx->N. For example, 128 Bytes if RSA-1024 is used.*

*\**

*\* \deprecated It is deprecated and discouraged to call this function*

*\* in #MBEDTLS\_RSA\_PUBLIC mode. Future versions of the library*

*\* are likely to remove the \p mode argument and have it*

*\* implicitly set to #MBEDTLS\_RSA\_PRIVATE.*

*\**

*\* \note Alternative implementations of RSA need not support*

*\* mode being set to #MBEDTLS\_RSA\_PUBLIC and might instead*

*\* return #MBEDTLS\_ERR\_PLATFORM\_FEATURE\_UNSUPPORTED.*

*\**

*\* \param ctx The RSA context.*

*\* \param f\_rng The RNG function. Only needed for #MBEDTLS\_RSA\_PRIVATE.*

*\* \param p\_rng The RNG context.*

*\* \param mode #MBEDTLS\_RSA\_PUBLIC or #MBEDTLS\_RSA\_PRIVATE.*

*\* \param label The buffer holding the custom label to use.*

*\* \param label\_len The length of the label.*

*\* \param olen The length of the plaintext.*

*\* \param input The buffer holding the encrypted data.*

*\* \param output The buffer to hold the plaintext.*

*\* \param output\_max\_len The maximum length of the output buffer.*

*\**

*\* \return \c 0 on success.*

*\* \return An \c MBEDTLS\_ERR\_RSA\_XXX error code on failure.*

*\*/*

int **mbedtls\_rsa\_rsaes\_oaep\_decrypt**( mbedtls\_rsa\_context \*ctx,

int (\*f\_rng)(void \*, unsigned char \*, size\_t),

void \*p\_rng,

int mode,

**const** unsigned char \*label, size\_t label\_len,

size\_t \*olen,

**const** unsigned char \*input,

unsigned char \*output,

size\_t output\_max\_len );

*/\*\**

*\* \brief This function performs a private RSA operation to sign*

*\* a message digest using PKCS#1.*

*\**

*\* It is the generic wrapper for performing a PKCS#1*

*\* signature using the \p mode from the context.*

*\**

*\* \note The \p sig buffer must be as large as the size*

*\* of \p ctx->N. For example, 128 Bytes if RSA-1024 is used.*

*\**

*\* \note For PKCS#1 v2.1 encoding, see comments on*

*\* mbedtls\_rsa\_rsassa\_pss\_sign() for details on*

*\* \p md\_alg and \p hash\_id.*

*\**

*\* \deprecated It is deprecated and discouraged to call this function*

*\* in #MBEDTLS\_RSA\_PUBLIC mode. Future versions of the library*

*\* are likely to remove the \p mode argument and have it*

*\* implicitly set to #MBEDTLS\_RSA\_PRIVATE.*

*\**

*\* \note Alternative implementations of RSA need not support*

*\* mode being set to #MBEDTLS\_RSA\_PUBLIC and might instead*

*\* return #MBEDTLS\_ERR\_PLATFORM\_FEATURE\_UNSUPPORTED.*

*\**

*\* \param ctx The RSA context.*

*\* \param f\_rng The RNG function. Needed for PKCS#1 v2.1 encoding and for*

*\* #MBEDTLS\_RSA\_PRIVATE.*

*\* \param p\_rng The RNG context.*

*\* \param mode #MBEDTLS\_RSA\_PUBLIC or #MBEDTLS\_RSA\_PRIVATE.*

*\* \param md\_alg The message-digest algorithm used to hash the original data.*

*\* Use #MBEDTLS\_MD\_NONE for signing raw data.*

*\* \param hashlen The length of the message digest. Only used if \p md\_alg is #MBEDTLS\_MD\_NONE.*

*\* \param hash The buffer holding the message digest.*

*\* \param sig The buffer to hold the ciphertext.*

*\**

*\* \return \c 0 if the signing operation was successful.*

*\* \return An \c MBEDTLS\_ERR\_RSA\_XXX error code on failure.*

*\*/*

int **mbedtls\_rsa\_pkcs1\_sign**( mbedtls\_rsa\_context \*ctx,

int (\*f\_rng)(void \*, unsigned char \*, size\_t),

void \*p\_rng,

int mode,

mbedtls\_md\_type\_t md\_alg,

unsigned int hashlen,

**const** unsigned char \*hash,

unsigned char \*sig );

*/\*\**

*\* \brief This function performs a PKCS#1 v1.5 signature*

*\* operation (RSASSA-PKCS1-v1\_5-SIGN).*

*\**

*\* \note The \p sig buffer must be as large as the size*

*\* of \p ctx->N. For example, 128 Bytes if RSA-1024 is used.*

*\**

*\* \deprecated It is deprecated and discouraged to call this function*

*\* in #MBEDTLS\_RSA\_PUBLIC mode. Future versions of the library*

*\* are likely to remove the \p mode argument and have it*

*\* implicitly set to #MBEDTLS\_RSA\_PRIVATE.*

*\**

*\* \note Alternative implementations of RSA need not support*

*\* mode being set to #MBEDTLS\_RSA\_PUBLIC and might instead*

*\* return #MBEDTLS\_ERR\_PLATFORM\_FEATURE\_UNSUPPORTED.*

*\**

*\* \param ctx The RSA context.*

*\* \param f\_rng The RNG function. Only needed for #MBEDTLS\_RSA\_PRIVATE.*

*\* \param p\_rng The RNG context.*

*\* \param mode #MBEDTLS\_RSA\_PUBLIC or #MBEDTLS\_RSA\_PRIVATE.*

*\* \param md\_alg The message-digest algorithm used to hash the original data.*

*\* Use #MBEDTLS\_MD\_NONE for signing raw data.*

*\* \param hashlen The length of the message digest. Only used if \p md\_alg is #MBEDTLS\_MD\_NONE.*

*\* \param hash The buffer holding the message digest.*

*\* \param sig The buffer to hold the ciphertext.*

*\**

*\* \return \c 0 if the signing operation was successful.*

*\* \return An \c MBEDTLS\_ERR\_RSA\_XXX error code on failure.*

*\*/*

int **mbedtls\_rsa\_rsassa\_pkcs1\_v15\_sign**( mbedtls\_rsa\_context \*ctx,

int (\*f\_rng)(void \*, unsigned char \*, size\_t),

void \*p\_rng,

int mode,

mbedtls\_md\_type\_t md\_alg,

unsigned int hashlen,

**const** unsigned char \*hash,

unsigned char \*sig );

*/\*\**

*\* \brief This function performs a PKCS#1 v2.1 PSS signature*

*\* operation (RSASSA-PSS-SIGN).*

*\**

*\* \note The \p sig buffer must be as large as the size*

*\* of \p ctx->N. For example, 128 Bytes if RSA-1024 is used.*

*\**

*\* \note The \p hash\_id in the RSA context is the one used for the*

*\* encoding. \p md\_alg in the function call is the type of hash*

*\* that is encoded. According to* **<em>***RFC-3447: Public-Key*

*\* Cryptography Standards (PKCS) #1 v2.1: RSA Cryptography*

*\* Specifications***</em>** *it is advised to keep both hashes the*

*\* same.*

*\**

*\* \note This function always uses the maximum possible salt size,*

*\* up to the length of the payload hash. This choice of salt*

*\* size complies with FIPS 186-4 §5.5 (e) and RFC 8017 (PKCS#1*

*\* v2.2) §9.1.1 step 3. Furthermore this function enforces a*

*\* minimum salt size which is the hash size minus 2 bytes. If*

*\* this minimum size is too large given the key size (the salt*

*\* size, plus the hash size, plus 2 bytes must be no more than*

*\* the key size in bytes), this function returns*

*\* #MBEDTLS\_ERR\_RSA\_BAD\_INPUT\_DATA.*

*\**

*\* \deprecated It is deprecated and discouraged to call this function*

*\* in #MBEDTLS\_RSA\_PUBLIC mode. Future versions of the library*

*\* are likely to remove the \p mode argument and have it*

*\* implicitly set to #MBEDTLS\_RSA\_PRIVATE.*

*\**

*\* \note Alternative implementations of RSA need not support*

*\* mode being set to #MBEDTLS\_RSA\_PUBLIC and might instead*

*\* return #MBEDTLS\_ERR\_PLATFORM\_FEATURE\_UNSUPPORTED.*

*\**

*\* \param ctx The RSA context.*

*\* \param f\_rng The RNG function. Needed for PKCS#1 v2.1 encoding and for*

*\* #MBEDTLS\_RSA\_PRIVATE.*

*\* \param p\_rng The RNG context.*

*\* \param mode #MBEDTLS\_RSA\_PUBLIC or #MBEDTLS\_RSA\_PRIVATE.*

*\* \param md\_alg The message-digest algorithm used to hash the original data.*

*\* Use #MBEDTLS\_MD\_NONE for signing raw data.*

*\* \param hashlen The length of the message digest. Only used if \p md\_alg is #MBEDTLS\_MD\_NONE.*

*\* \param hash The buffer holding the message digest.*

*\* \param sig The buffer to hold the ciphertext.*

*\**

*\* \return \c 0 if the signing operation was successful.*

*\* \return An \c MBEDTLS\_ERR\_RSA\_XXX error code on failure.*

*\*/*

int **mbedtls\_rsa\_rsassa\_pss\_sign**( mbedtls\_rsa\_context \*ctx,

int (\*f\_rng)(void \*, unsigned char \*, size\_t),

void \*p\_rng,

int mode,

mbedtls\_md\_type\_t md\_alg,

unsigned int hashlen,

**const** unsigned char \*hash,

unsigned char \*sig );

*/\*\**

*\* \brief This function performs a public RSA operation and checks*

*\* the message digest.*

*\**

*\* This is the generic wrapper for performing a PKCS#1*

*\* verification using the mode from the context.*

*\**

*\* \note The \p sig buffer must be as large as the size*

*\* of \p ctx->N. For example, 128 Bytes if RSA-1024 is used.*

*\**

*\* \note For PKCS#1 v2.1 encoding, see comments on*

*\* mbedtls\_rsa\_rsassa\_pss\_verify() about \p md\_alg and*

*\* \p hash\_id.*

*\**

*\* \deprecated It is deprecated and discouraged to call this function*

*\* in #MBEDTLS\_RSA\_PRIVATE mode. Future versions of the library*

*\* are likely to remove the \p mode argument and have it*

*\* set to #MBEDTLS\_RSA\_PUBLIC.*

*\**

*\* \note Alternative implementations of RSA need not support*

*\* mode being set to #MBEDTLS\_RSA\_PRIVATE and might instead*

*\* return #MBEDTLS\_ERR\_PLATFORM\_FEATURE\_UNSUPPORTED.*

*\**

*\* \param ctx The RSA public key context.*

*\* \param f\_rng The RNG function. Only needed for #MBEDTLS\_RSA\_PRIVATE.*

*\* \param p\_rng The RNG context.*

*\* \param mode #MBEDTLS\_RSA\_PUBLIC or #MBEDTLS\_RSA\_PRIVATE.*

*\* \param md\_alg The message-digest algorithm used to hash the original data.*

*\* Use #MBEDTLS\_MD\_NONE for signing raw data.*

*\* \param hashlen The length of the message digest. Only used if \p md\_alg is #MBEDTLS\_MD\_NONE.*

*\* \param hash The buffer holding the message digest.*

*\* \param sig The buffer holding the ciphertext.*

*\**

*\* \return \c 0 if the verify operation was successful.*

*\* \return An \c MBEDTLS\_ERR\_RSA\_XXX error code on failure.*

*\*/*

int **mbedtls\_rsa\_pkcs1\_verify**( mbedtls\_rsa\_context \*ctx,

int (\*f\_rng)(void \*, unsigned char \*, size\_t),

void \*p\_rng,

int mode,

mbedtls\_md\_type\_t md\_alg,

unsigned int hashlen,

**const** unsigned char \*hash,

**const** unsigned char \*sig );

*/\*\**

*\* \brief This function performs a PKCS#1 v1.5 verification*

*\* operation (RSASSA-PKCS1-v1\_5-VERIFY).*

*\**

*\* \note The \p sig buffer must be as large as the size*

*\* of \p ctx->N. For example, 128 Bytes if RSA-1024 is used.*

*\**

*\* \deprecated It is deprecated and discouraged to call this function*

*\* in #MBEDTLS\_RSA\_PRIVATE mode. Future versions of the library*

*\* are likely to remove the \p mode argument and have it*

*\* set to #MBEDTLS\_RSA\_PUBLIC.*

*\**

*\* \note Alternative implementations of RSA need not support*

*\* mode being set to #MBEDTLS\_RSA\_PRIVATE and might instead*

*\* return #MBEDTLS\_ERR\_PLATFORM\_FEATURE\_UNSUPPORTED.*

*\**

*\* \param ctx The RSA public key context.*

*\* \param f\_rng The RNG function. Only needed for #MBEDTLS\_RSA\_PRIVATE.*

*\* \param p\_rng The RNG context.*

*\* \param mode #MBEDTLS\_RSA\_PUBLIC or #MBEDTLS\_RSA\_PRIVATE.*

*\* \param md\_alg The message-digest algorithm used to hash the original data.*

*\* Use #MBEDTLS\_MD\_NONE for signing raw data.*

*\* \param hashlen The length of the message digest. Only used if \p md\_alg is #MBEDTLS\_MD\_NONE.*

*\* \param hash The buffer holding the message digest.*

*\* \param sig The buffer holding the ciphertext.*

*\**

*\* \return \c 0 if the verify operation was successful.*

*\* \return An \c MBEDTLS\_ERR\_RSA\_XXX error code on failure.*

*\*/*

int **mbedtls\_rsa\_rsassa\_pkcs1\_v15\_verify**( mbedtls\_rsa\_context \*ctx,

int (\*f\_rng)(void \*, unsigned char \*, size\_t),

void \*p\_rng,

int mode,

mbedtls\_md\_type\_t md\_alg,

unsigned int hashlen,

**const** unsigned char \*hash,

**const** unsigned char \*sig );

*/\*\**

*\* \brief This function performs a PKCS#1 v2.1 PSS verification*

*\* operation (RSASSA-PSS-VERIFY).*

*\**

*\* The hash function for the MGF mask generating function*

*\* is that specified in the RSA context.*

*\**

*\* \note The \p sig buffer must be as large as the size*

*\* of \p ctx->N. For example, 128 Bytes if RSA-1024 is used.*

*\**

*\* \note The \p hash\_id in the RSA context is the one used for the*

*\* verification. \p md\_alg in the function call is the type of*

*\* hash that is verified. According to* **<em>***RFC-3447: Public-Key*

*\* Cryptography Standards (PKCS) #1 v2.1: RSA Cryptography*

*\* Specifications***</em>** *it is advised to keep both hashes the*

*\* same. If \p hash\_id in the RSA context is unset,*

*\* the \p md\_alg from the function call is used.*

*\**

*\* \deprecated It is deprecated and discouraged to call this function*

*\* in #MBEDTLS\_RSA\_PRIVATE mode. Future versions of the library*

*\* are likely to remove the \p mode argument and have it*

*\* implicitly set to #MBEDTLS\_RSA\_PUBLIC.*

*\**

*\* \note Alternative implementations of RSA need not support*

*\* mode being set to #MBEDTLS\_RSA\_PRIVATE and might instead*

*\* return #MBEDTLS\_ERR\_PLATFORM\_FEATURE\_UNSUPPORTED.*

*\**

*\* \param ctx The RSA public key context.*

*\* \param f\_rng The RNG function. Only needed for #MBEDTLS\_RSA\_PRIVATE.*

*\* \param p\_rng The RNG context.*

*\* \param mode #MBEDTLS\_RSA\_PUBLIC or #MBEDTLS\_RSA\_PRIVATE.*

*\* \param md\_alg The message-digest algorithm used to hash the original data.*

*\* Use #MBEDTLS\_MD\_NONE for signing raw data.*

*\* \param hashlen The length of the message digest. Only used if \p md\_alg is #MBEDTLS\_MD\_NONE.*

*\* \param hash The buffer holding the message digest.*

*\* \param sig The buffer holding the ciphertext.*

*\**

*\* \return \c 0 if the verify operation was successful.*

*\* \return An \c MBEDTLS\_ERR\_RSA\_XXX error code on failure.*

*\*/*

int **mbedtls\_rsa\_rsassa\_pss\_verify**( mbedtls\_rsa\_context \*ctx,

int (\*f\_rng)(void \*, unsigned char \*, size\_t),

void \*p\_rng,

int mode,

mbedtls\_md\_type\_t md\_alg,

unsigned int hashlen,

**const** unsigned char \*hash,

**const** unsigned char \*sig );

*/\*\**

*\* \brief This function performs a PKCS#1 v2.1 PSS verification*

*\* operation (RSASSA-PSS-VERIFY).*

*\**

*\* The hash function for the MGF mask generating function*

*\* is that specified in \p mgf1\_hash\_id.*

*\**

*\* \note The \p sig buffer must be as large as the size*

*\* of \p ctx->N. For example, 128 Bytes if RSA-1024 is used.*

*\**

*\* \note The \p hash\_id in the RSA context is ignored.*

*\**

*\* \param ctx The RSA public key context.*

*\* \param f\_rng The RNG function. Only needed for #MBEDTLS\_RSA\_PRIVATE.*

*\* \param p\_rng The RNG context.*

*\* \param mode #MBEDTLS\_RSA\_PUBLIC or #MBEDTLS\_RSA\_PRIVATE.*

*\* \param md\_alg The message-digest algorithm used to hash the original data.*

*\* Use #MBEDTLS\_MD\_NONE for signing raw data.*

*\* \param hashlen The length of the message digest. Only used if \p md\_alg is*

*\* #MBEDTLS\_MD\_NONE.*

*\* \param hash The buffer holding the message digest.*

*\* \param mgf1\_hash\_id The message digest used for mask generation.*

*\* \param expected\_salt\_len The length of the salt used in padding. Use*

*\* #MBEDTLS\_RSA\_SALT\_LEN\_ANY to accept any salt length.*

*\* \param sig The buffer holding the ciphertext.*

*\**

*\* \return \c 0 if the verify operation was successful.*

*\* \return An \c MBEDTLS\_ERR\_RSA\_XXX error code on failure.*

*\*/*

int **mbedtls\_rsa\_rsassa\_pss\_verify\_ext**( mbedtls\_rsa\_context \*ctx,

int (\*f\_rng)(void \*, unsigned char \*, size\_t),

void \*p\_rng,

int mode,

mbedtls\_md\_type\_t md\_alg,

unsigned int hashlen,

**const** unsigned char \*hash,

mbedtls\_md\_type\_t mgf1\_hash\_id,

int expected\_salt\_len,

**const** unsigned char \*sig );

*/\*\**

*\* \brief This function copies the components of an RSA context.*

*\**

*\* \param dst The destination context.*

*\* \param src The source context.*

*\**

*\* \return \c 0 on success.*

*\* \return #MBEDTLS\_ERR\_MPI\_ALLOC\_FAILED on memory allocation failure.*

*\*/*

int **mbedtls\_rsa\_copy**( mbedtls\_rsa\_context \*dst, **const** mbedtls\_rsa\_context \*src );

*/\*\**

*\* \brief This function frees the components of an RSA key.*

*\**

*\* \param ctx The RSA Context to free.*

*\*/*

void **mbedtls\_rsa\_free**( mbedtls\_rsa\_context \*ctx );

*/\*\**

*\* \brief The RSA checkup routine.*

*\**

*\* \return \c 0 on success.*

*\* \return \c 1 on failure.*

*\*/*

int **mbedtls\_rsa\_self\_test**( int verbose );

**#ifdef** \_\_cplusplus

}

**#endif**

**#endif** */\* rsa.h \*/*