NAME

ASN1_item_sign, ASN1_item_sign_ex, ASN1_item_sign_ctx, ASN1_item_verify,
ASN1_item_verify_ex, ASN1_item_verify_ctx - ASN1 sign and verify

SYNOPSIS

#include <openssl/x509.h>

int ASN1_item_sign_ex(const ASN1_ITEM *it, X509_ALGOR *algor1,
            X509_ALGOR *algor2, ASN1_BIT_STRING *signature,
            const void *data, const ASN1_OCTET_STRING *id,
            EVP_PKEY *pkey, const EVP_MD *md, OSSL_LIB_CTX *libctx,
            const char *propq);

int ASN1_item_sign(const ASN1_ITEM *it, X509_ALGOR *algor1, X509_ALGOR *algor2,
            ASN1_BIT_STRING *signature, const void *data,
            EVP_PKEY *pkey, const EVP_MD *md);

int ASN1_item_sign_ctx(const ASN1_ITEM *it, X509_ALGOR *algor1,
            X509_ALGOR *algor2, ASN1_BIT_STRING *signature,
            const void *data, EVP_MD_CTX *ctx);

int ASN1_item_verify_ex(const ASN1_ITEM *it, const X509_ALGOR *alg,
            const ASN1_BIT_STRING *signature, const void *data,
            const ASN1_OCTET_STRING *id, EVP_PKEY *pkey,
            OSSL_LIB_CTX *libctx, const char *propq);

int ASN1_item_verify(const ASN1_ITEM *it, const X509_ALGOR *alg,
            const ASN1_BIT_STRING *signature, const void *data,
            EVP_PKEY *pkey);

int ASN1_item_verify_ctx(const ASN1_ITEM *it, const X509_ALGOR *alg,
            const ASN1_BIT_STRING *signature, const void *data,
            EVP_MD_CTX *ctx);

DESCRIPTION

ASN1_item_sign_ex() is used to sign arbitrary ASN1 data using a data object *data*, the ASN.1 structure
it, private key *pkey* and message digest *md*. The data that is signed is formed by taking the data object
in *data* and converting it to der format using the ASN.1 structure *it*. The *data* that will be signed, and a
structure containing the signature may both have a copy of the **X509_ALGOR**. The
ASN1_item_sign_ex() function will write the correct **X509_ALGOR** to the structs based on the

algorithms and parameters that have been set up. If one of *algor1* or *algor2* points to the **X509_ALGOR** of the *data* to be signed, then that **X509_ALGOR** will first be written before the signature is generated. Examples of valid values that can be used by the ASN.1 structure *it* are ASN1_ITEM_rptr(X509_CINF), ASN1_ITEM_rptr(X509_REQ_INFO) and ASN1_ITEM_rptr(X509_CRL_INFO). The **OSSL_LIB_CTX** specified in *libctx* and the property query string specified in *props* are used when searching for algorithms in providers. The generated signature is set into *signature*. The optional parameter *id* can be NULL, but can be set for special key types. See **EVP_PKEY_CTX_set1_id()** for further info. The output parameters <algor1> and *algor2* are ignored if they are NULL.

ASN1_item_sign() is similar to **ASN1_item_sign_ex()** but uses default values of NULL for the *id*, *libctx* and *propq*.

ASN1_item_sign_ctx() is similar to **ASN1_item_sign()** but uses the parameters contained in digest context *ctx*.

ASN1_item_verify_ex() is used to verify the signature *signature* of internal data *data* using the public key *pkey* and algorithm identifier *alg*. The data that is verified is formed by taking the data object in *data* and converting it to der format using the ASN.1 structure *it*. The **OSSL_LIB_CTX** specified in *libctx* and the property query string specified in *props* are used when searching for algorithms in providers. The optional parameter *id* can be NULL, but can be set for special key types. See **EVP_PKEY_CTX_set1_id()** for further info.

ASN1_item_verify() is similar to **ASN1_item_verify_ex()** but uses default values of NULL for the *id*, *libctx* and *propq*.

ASN1_item_verify_ctx() is similar to **ASN1_item_verify()** but uses the parameters contained in digest context *ctx*.

RETURN VALUES

All sign functions return the size of the signature in bytes for success and zero for failure.

All verify functions return 1 if the signature is valid and 0 if the signature check fails. If the signature could not be checked at all because it was ill-formed or some other error occurred then -1 is returned.

EXAMPLES

In the following example a 'MyObject' object is signed using the key contained in an EVP_MD_CTX. The signature is written to MyObject.signature. The object is then output in DER format and then loaded back in and verified.

```
#include <openssl/x509.h>
#include <openssl/asn1t.h>

/* An object used to store the ASN1 data fields that will be signed */
typedef struct MySignInfoObject_st
{
    ASN1_INTEGER *version;
    X509_ALGOR sig_alg;
} MySignInfoObject;

DECLARE_ASN1_FUNCTIONS(MySignInfoObject)
/*
 * A higher level object containing the ASN1 fields, signature alg and
 * output signature.
 */
typedef struct MyObject_st
{
    MySignInfoObject info;
    X509_ALGOR sig_alg;
    ASN1_BIT_STRING *signature;
} MyObject;

DECLARE_ASN1_FUNCTIONS(MyObject)

/* The ASN1 definition of MySignInfoObject */
ASN1_SEQUENCE_cb(MySignInfoObject, NULL) = {
    ASN1_SIMPLE(MySignInfoObject, version, ASN1_INTEGER)
    ASN1_EMBED(MySignInfoObject, sig_alg, X509_ALGOR),
} ASN1_SEQUENCE_END_cb(MySignInfoObject, MySignInfoObject)

/* new, free, d2i & i2d functions for MySignInfoObject */
IMPLEMENT_ASN1_FUNCTIONS(MySignInfoObject)

/* The ASN1 definition of MyObject */
ASN1_SEQUENCE_cb(MyObject, NULL) = {
    ASN1_EMBED(MyObject, info, MySignInfoObject),
    ASN1_EMBED(MyObject, sig_alg, X509_ALGOR),
    ASN1_SIMPLE(MyObject, signature, ASN1_BIT_STRING)
} ASN1_SEQUENCE_END_cb(MyObject, MyObject)
```

```
/* new, free, d2i & i2d functions for MyObject */
IMPLEMENT_ASN1_FUNCTIONS(MyObject)

int test_asn1_item_sign_verify(const char *mdname, EVP_PKEY *pkey, long version)
{
  int ret = 0;
  unsigned char *obj_der = NULL;
  const unsigned char *p = NULL;
  MyObject *obj = NULL, *loaded_obj = NULL;
  const ASN1_ITEM *it = ASN1_ITEM_rptr(MySignInfoObject);
  EVP_MD_CTX *sctx = NULL, *vctx = NULL;
  int len;

  /* Create MyObject and set its version */
  obj = MyObject_new();
  if (obj == NULL)
     goto err;
  if (!ASN1_INTEGER_set(obj->info.version, version))
     goto err;

  /* Set the key and digest used for signing */
  sctx = EVP_MD_CTX_new();
  if (sctx == NULL
     || !EVP_DigestSignInit_ex(sctx, NULL, mdname, NULL, NULL, pkey))
     goto err;

  /*
   * it contains the mapping between ASN.1 data and an object MySignInfoObject
   * obj->info is the 'MySignInfoObject' object that will be
   *   converted into DER data and then signed.
   * obj->signature will contain the output signature.
   * obj->sig_alg is filled with the private key's signing algorithm id.
   * obj->info.sig_alg is another copy of the signing algorithm id that sits
   * within MyObject.
   */
  len = ASN1_item_sign_ctx(it, &obj->sig_alg, &obj->info.sig_alg,
                   obj->signature, &obj->info, sctx);
  if (len <= 0
     || X509_ALGOR_cmp(&obj->sig_alg, &obj->info.sig_alg) != 0)
     goto err;
```

```
        /* Output MyObject in der form */
        len = i2d_MyObject(obj, &obj_der);
        if (len <= 0)
            goto err;

        /* Set the key and digest used for verifying */
        vctx = EVP_MD_CTX_new();
        if (vctx == NULL
            || !EVP_DigestVerifyInit_ex(vctx, NULL, mdname, NULL, NULL, pkey))
            goto err;

        /* Load the der data back into an object */
        p = obj_der;
        loaded_obj = d2i_MyObject(NULL, &p, len);
        if (loaded_obj == NULL)
            goto err;
        /* Verify the loaded object */
        ret = ASN1_item_verify_ctx(it, &loaded_obj->sig_alg, loaded_obj->signature,
                       &loaded_obj->info, vctx);
    err:
      OPENSSL_free(obj_der);
      MyObject_free(loaded_obj);
      MyObject_free(obj);
      EVP_MD_CTX_free(sctx);
      EVP_MD_CTX_free(vctx);
      return ret;
     }
```

SEE ALSO
X509_sign(3), **X509_verify**(3)

HISTORY
ASN1_item_sign_ex() and **ASN1_item_verify_ex()** were added in OpenSSL 3.0.

COPYRIGHT
Copyright 2020-2023 The OpenSSL Project Authors. All Rights Reserved.

Licensed under the Apache License 2.0 (the "License"). You may not use this file except in compliance with the License. You can obtain a copy in the file LICENSE in the source distribution or at <https://www.openssl.org/source/license.html>.