NAME

EVP_SignInit, EVP_SignInit_ex, EVP_SignUpdate, EVP_SignFinal – EVP signing functions

SYNOPSIS

```
 #include <openssl/evp.h>

 int EVP_SignInit_ex(EVP_MD_CTX *ctx, const EVP_MD *type, ENGINE *impl);
 int EVP_SignUpdate(EVP_MD_CTX *ctx, const void *d, unsigned int cnt);
 int EVP_SignFinal(EVP_MD_CTX *ctx, unsigned char *sig, unsigned int *s, EVP_PKEY *p

 void EVP_SignInit(EVP_MD_CTX *ctx, const EVP_MD *type);
```

DESCRIPTION

The EVP signature routines are a high-level interface to digital signatures.

EVP_SignInit_ex() sets up signing context *ctx* to use digest *type* from **ENGINE** *impl*. *ctx* must be created with **EVP_MD_CTX_new()** before calling this function.

EVP_SignUpdate() hashes *cnt* bytes of data at *d* into the signature context *ctx*. This function can be called several times on the same *ctx* to include additional data.

EVP_SignFinal() signs the data in *ctx* using the private key *pkey* and places the signature in *sig*. *sig* must be at least `EVP_PKEY_size(pkey)` bytes in size. *s* is an OUT parameter, and not used as an IN parameter. The number of bytes of data written (i.e. the length of the signature) will be written to the integer at *s*, at most `EVP_PKEY_size(pkey)` bytes will be written.

EVP_SignInit() initializes a signing context *ctx* to use the default implementation of digest *type*.

RETURN VALUES

EVP_SignInit_ex(), **EVP_SignUpdate()** and **EVP_SignFinal()** return 1 for success and 0 for failure.

The error codes can be obtained by **ERR_get_error(3)** .

NOTES

The **EVP** interface to digital signatures should almost always be used in preference to the low-level interfaces. This is because the code then becomes transparent to the algorithm used and much more flexible.

When signing with DSA private keys the random number generator must be seeded. If the automatic seeding or reseeding of the OpenSSL CSPRNG fails due to external circumstances (see **RAND(7)**), the operation will fail. This requirement does not hold for RSA signatures.

The call to **EVP_SignFinal()** internally finalizes a copy of the digest context. This means that calls to **EVP_SignUpdate()** and **EVP_SignFinal()** can be called later to digest and sign additional data.

Since only a copy of the digest context is ever finalized the context must be cleaned up after use by calling **EVP_MD_CTX_free()** or a memory leak will occur.

BUGS

Older versions of this documentation wrongly stated that calls to **EVP_SignUpdate()** could not be made after calling **EVP_SignFinal()**.

Since the private key is passed in the call to **EVP_SignFinal()** any error relating to the private key (for example an unsuitable key and digest combination) will not be indicated until after potentially large amounts of data have been passed through **EVP_SignUpdate()**.

It is not possible to change the signing parameters using these function.

The previous two bugs are fixed in the newer EVP_SignDigest*() function.

SEE ALSO

EVP_PKEY_size(3) , **EVP_PKEY_bits(3)** , **EVP_PKEY_security_bits(3)** , **EVP_VerifyInit(3)** , **EVP_DigestInit(3)** , **evp(7)** , **HMAC(3)** , **MD2(3)** , **MD5(3)** , **MDC2(3)** , **RIPEMD160(3)** , **SHA1(3)** , **dgst(1)**

COPYRIGHT