

NAME

EVP_SIGNATURE-ED25519, EVP_SIGNATURE-ED448, Ed25519, Ed448 - EVP_PKEY Ed25519 and Ed448 support

DESCRIPTION

The **Ed25519** and **Ed448** EVP_PKEY implementation supports key generation, one-shot digest sign and digest verify using PureEdDSA and **Ed25519** or **Ed448** (see RFC8032). It has associated private and public key formats compatible with RFC 8410.

ED25519 and ED448 Signature Parameters

No additional parameters can be set during one-shot signing or verification. In particular, because PureEdDSA is used, a digest must **NOT** be specified when signing or verifying. See **EVP_PKEY-X25519(7)** for information related to **X25519** and **X448** keys.

The following signature parameters can be retrieved using **EVP_PKEY_CTX_get_params()**.

"algorithm-id" (**OSSL_SIGNATURE_PARAM_ALGORITHM_ID**) <octet string>

The parameters are described in **provider-signature(7)**.

NOTES

The PureEdDSA algorithm does not support the streaming mechanism of other signature algorithms using, for example, **EVP_DigestUpdate()**. The message to sign or verify must be passed using the one-shot **EVP_DigestSign()** and **EVP_DigestVerify()** functions.

When calling **EVP_DigestSignInit()** or **EVP_DigestVerifyInit()**, the digest *type* parameter **MUST** be set to NULL.

Applications wishing to sign certificates (or other structures such as CRLs or certificate requests) using Ed25519 or Ed448 can either use **X509_sign()** or **X509_sign_ctx()** in the usual way.

Ed25519 or Ed448 private keys can be set directly using **EVP_PKEY_new_raw_private_key(3)** or loaded from a PKCS#8 private key file using **PEM_read_bio_PrivateKey(3)** (or similar function). Completely new keys can also be generated (see the example below). Setting a private key also sets the associated public key.

Ed25519 or Ed448 public keys can be set directly using **EVP_PKEY_new_raw_public_key(3)** or loaded from a SubjectPublicKeyInfo structure in a PEM file using **PEM_read_bio_PUBKEY(3)** (or similar function).

Ed25519 and Ed448 can be tested with the **openssl-speed(1)** application since version 1.1.1. Valid

algorithm names are **ed25519**, **ed448** and **eddsa**. If **eddsa** is specified, then both Ed25519 and Ed448 are benchmarked.

EXAMPLES

To sign a message using a ED25519 or ED448 key:

```
void do_sign(EVP_PKEY *ed_key, unsigned char *msg, size_t msg_len)
{
    size_t sig_len;
    unsigned char *sig = NULL;
    EVP_MD_CTX *md_ctx = EVP_MD_CTX_new();

    EVP_DigestSignInit(md_ctx, NULL, NULL, NULL, ed_key);
    /* Calculate the requires size for the signature by passing a NULL buffer */
    EVP_DigestSign(md_ctx, NULL, &sig_len, msg, msg_len);
    sig = OPENSSL_zalloc(sig_len);

    EVP_DigestSign(md_ctx, sig, &sig_len, msg, msg_len);
    ...
    OPENSSL_free(sig);
    EVP_MD_CTX_free(md_ctx);
}
```

SEE ALSO

EVP_PKEY-X25519(7) provider-signature(7), EVP_DigestSignInit(3), EVP_DigestVerifyInit(3),

COPYRIGHT

Copyright 2017-2021 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>>.