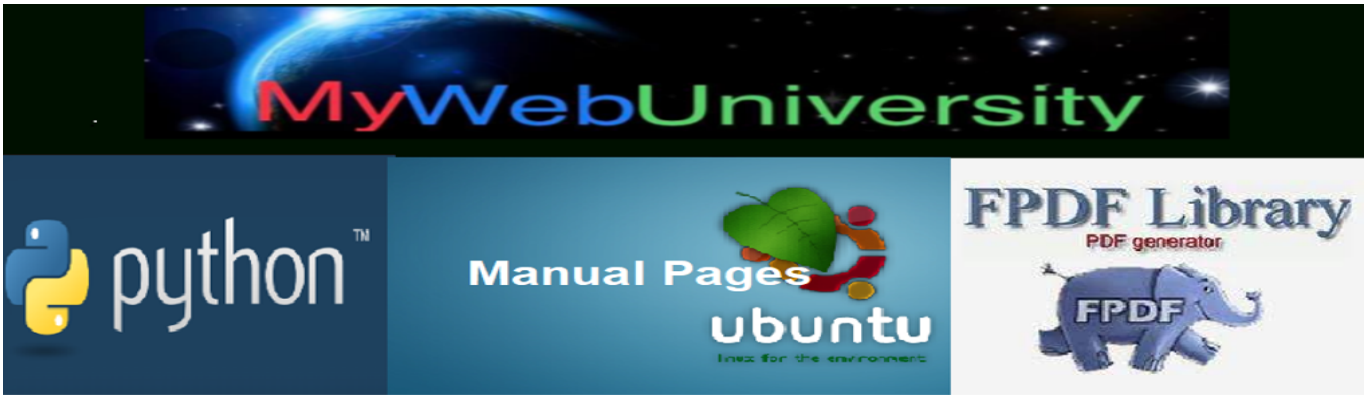

Rocky Enterprise Linux 9.2 Manual Pages on command 'Crypt::PK::Ed25519.3pm'

$ man Crypt::PK::Ed25519.3pm

Crypt::PK::Ed25519(3pm) User Contributed Perl Documentation Crypt::PK::Ed25519(3pm)

NAME

Crypt::PK::Ed25519 - Digital signature based on Ed25519

SYNOPSIS

```
use Crypt::PK::Ed25519;

#Signature: Alice
my $priv = Crypt::PK::Ed25519->new('Alice_priv_ed25519.der');
my $sig = $priv->sign_message($message);

#Signature: Bob (received $message + $sig)
my $pub = Crypt::PK::Ed25519->new('Alice_pub_ed25519.der');
$pub->verify_message($sig, $message) or die "ERROR";

#Load key
my $pk = Crypt::PK::Ed25519->new;
my $pk_hex = "A05D1AEA5830AC9A65CDFB384660D497E3697C46B419CF2CEC85DE8BD245459D";
$pk->import_key_raw(pack("H*", $pk_hex), "public");
my $sk = Crypt::PK::Ed25519->new;
my $sk_hex = "45C109BA6FD24E8B67D23EFB6B92D99CD457E2137172C0D749FE2B5A0C142DAD";
$sk->import_key_raw(pack("H*", $sk_hex), "private");
```

```
#Key generation
my $pk = Crypt::PK::Ed25519->new->generate_key;
my $private_der = $pk->export_key_der('private');
my $public_der  = $pk->export_key_der('public');
my $private_pem = $pk->export_key_pem('private');
my $public_pem  = $pk->export_key_pem('public');
my $private_raw = $pk->export_key_raw('private');
my $public_raw  = $pk->export_key_raw('public');
my $private_jwk = $pk->export_key_jwk('private');
my $public_jwk  = $pk->export_key_jwk('public');
```

DESCRIPTION

Since: CryptX-0.067

METHODS

new

```
my $pk = Crypt::PK::Ed25519->new();
#or
my $pk = Crypt::PK::Ed25519->new($priv_or_pub_key_filename);
#or
my $pk = Crypt::PK::Ed25519->new(\$buffer_containing_priv_or_pub_key);
```

Support for password protected PEM keys

```
my $pk = Crypt::PK::Ed25519->new($priv_pem_key_filename, $password);
#or
my $pk = Crypt::PK::Ed25519->new(\$buffer_containing_priv_pem_key, $password);
```

generate_key

Uses Yarrow-based cryptographically strong random number generator seeded with random data taken from "/dev/random" (UNIX) or "CryptGenRandom" (Win32).

```
$pk->generate_key;
```

import_key

Loads private or public key in DER or PEM format.

```
$pk->import_key($filename);
#or
$pk->import_key(\$buffer_containing_key);
```

Support for password protected PEM keys:

```
$pk->import_key($filename, $password);
#or
$pk->import_key(\$buffer_containing_key, $password);
```

Loading private or public keys form perl hash:

```
$pk->import_key($hashref);

# the $hashref is either a key exported via key2hash
$pk->import_key({
    curve => "ed25519",
    pub   => "A05D1AEA5830AC9A65CDFB384660D497E3697C46B419CF2CEC85DE8BD245459D",
    priv  => "45C109BA6FD24E8B67D23EFB6B92D99CD457E2137172C0D749FE2B5A0C142DAD",
});

# or a hash with items corresponding to JWK (JSON Web Key)
$pk->import_key({
     kty => "OKP",
     crv => "Ed25519",
     d   => "RcEJum_STotn0j77a5LZnNRX4hNxcsDXSf4rWgwULa0",
     x   => "oF0a6lgwrJplzfs4RmDUl-NpfEa0Gc8s7IXei9JFRZ0",
});
```

Supported key formats:

```
# all formats can be loaded from a file
my $pk = Crypt::PK::Ed25519->new($filename);

# or from a buffer containing the key
my $pk = Crypt::PK::Ed25519->new(\$buffer_with_key);
```

? Ed25519 private keys in PEM format

```
-----BEGIN ED25519 PRIVATE KEY-----
MC4CAQAwBQYDK2VwBCIEIEXBCbpv0k6LZ9I++2uS2ZzUV+ITcXLA10n+K1oMFC2t
-----END ED25519 PRIVATE KEY-----
```

? Ed25519 private keys in password protected PEM format

```
-----BEGIN ED25519 PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-CBC,6A64D756D49C1EFF

8xQ7OyfQ10IITNEKcJGZA53Z1yk+NJQU7hrKqXwChZtgWNInhMBJRl9pozLKDSkH
v7u6EOve8NY=
-----END ED25519 PRIVATE KEY-----
```

? PKCS#8 private keys

```
-----BEGIN PRIVATE KEY-----
MC4CAQAwBQYDK2VwBCIEIEXBCbpv0k6LZ9I++2uS2ZzUV+ITcXLA10n+K1oMFC2t
-----END PRIVATE KEY-----
```

? PKCS#8 encrypted private keys

-----BEGIN ENCRYPTED PRIVATE KEY-----

MIGHMEsGCSqGSIb3DQEFDTA+MCkGCSqGSIb3DQEFDDAcBAjPx9JkdpRH2QICCAAw

DAYIKoZIhvcNAgkFADARBgUrDgMCBwQIWWieQojaWTcEOGj43SxqHUys4Eb2M27N

AkhqpmhosOxKrpGi0L3h8m8ipHE8EwI94NeOMsjfVw60aJuCrssY5vKN

-----END ENCRYPTED PRIVATE KEY-----

? Ed25519 public keys in PEM format

-----BEGIN PUBLIC KEY-----

MCowBQYDK2VwAyEAoF0a6lgwrJplzfs4RmDUl+NpfEa0Gc8s7IXei9JFRZ0=

-----END PUBLIC KEY-----

? Ed25519 public key from X509 certificate

-----BEGIN CERTIFICATE-----

MIIBODCB66ADAgECAhRWDU9FZBBUZ7KTdX8f7Bco8jsoaTAFBgMrZXAwETEPMA0G

A1UEAwwGQ3J5cHRYMCAXDTIwMDExOTEzMDIwMloYDzIyOTMxMTAyMTMwMjAyWjAR

MQ8wDQYDVQQDDAZDcnlwdFgwKjAFBgMrZXADIQCgXRrqWDCsmmXN+zhGYNSX42l8

RrQZzyzshd6L0kVFnaNTMFEwHQYDVR0OBBYEFHCGFtVibAxxWYyRt5wazMpqSZDV

MB8GA1UdIwQYMBaAFHCGFtVibAxxWYyRt5wazMpqSZDVMA8GA1UdEwEB/wQFMAMB

Af8wBQYDK2VwA0EAqG/+98smzqF/wmFX3zHXSaA67as202HnBJod1Tiurw1f+lr3

BX6OMtsDpgRq9O77IF1Qyx/MdJEwwErczOIbAA==

-----END CERTIFICATE-----

? SSH public Ed25519 keys

ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIL0XsiFcRDp6Hpsoak8OdiiBMJhM2UKszNTxoGS7dJ++

? SSH public Ed25519 keys (RFC-4716 format)

---- BEGIN SSH2 PUBLIC KEY ----

Comment: "256-bit ED25519, converted from OpenSSH"

AAAAC3NzaC1lZDI1NTE5AAAAIL0XsiFcRDp6Hpsoak8OdiiBMJhM2UKszNTxoGS7dJ++

---- END SSH2 PUBLIC KEY ----

? Ed25519 private keys in JSON Web Key (JWK) format

See <https://tools.ietf.org/html/rfc8037>

{
"kty":"OKP",
"crv":"Ed25519",
"x":"oF0a6lgwrJplzfs4RmDUl-NpfEa0Gc8s7IXei9JFRZ0",
"d":"RcEJum_STotn0j77a5LZnNRX4hNxcsDXSf4rWgwULa0",
}

BEWARE: For JWK support you need to have JSON module installed.

? Ed25519 public keys in JSON Web Key (JWK) format

{
"kty":"OKP",
"crv":"Ed25519",
"x":"oF0a6lgwrJplzfs4RmDUl-NpfEa0Gc8s7IXei9JFRZ0",
}

BEWARE: For JWK support you need to have JSON module installed.

import_key_raw

Import raw public/private key - can load raw key data exported by "export_key_raw".

```
$pk->import_key_raw($key, 'public');
$pk->import_key_raw($key, 'private');
```

export_key_der

```
my $private_der = $pk->export_key_der('private');
```

```
    #or
    my $public_der = $pk->export_key_der('public');
```

export_key_pem

```
    my $private_pem = $pk->export_key_pem('private');
    #or
    my $public_pem = $pk->export_key_pem('public');
```

Support for password protected PEM keys

```
    my $private_pem = $pk->export_key_pem('private', $password);
    #or
    my $private_pem = $pk->export_key_pem('private', $password, $cipher);

    # supported ciphers: 'DES-CBC'
    #                    'DES-EDE3-CBC'
    #                    'SEED-CBC'
    #                    'CAMELLIA-128-CBC'
    #                    'CAMELLIA-192-CBC'
    #                    'CAMELLIA-256-CBC'
    #                    'AES-128-CBC'
    #                    'AES-192-CBC'
    #                    'AES-256-CBC' (DEFAULT)
```

export_key_jwk

Exports public/private keys as a JSON Web Key (JWK).

```
    my $private_json_text = $pk->export_key_jwk('private');
    #or
    my $public_json_text = $pk->export_key_jwk('public');
```

Also exports public/private keys as a perl HASH with JWK structure.

```
    my $jwk_hash = $pk->export_key_jwk('private', 1);
    #or
    my $jwk_hash = $pk->export_key_jwk('public', 1);
```

BEWARE: For JWK support you need to have JSON module installed.

export_key_raw

Export raw public/private key

```
    my $private_bytes = $pk->export_key_raw('private');
    #or
    my $public_bytes = $pk->export_key_raw('public');
```

sign_message

```
    my $signature = $priv->sign_message($message);
```

verify_message

```
    my $valid = $pub->verify_message($signature, $message)
```

is_private

```
    my $rv = $pk->is_private;
    # 1 .. private key loaded
    # 0 .. public key loaded
    # undef .. no key loaded
```

key2hash

```
    my $hash = $pk->key2hash;

    # returns hash like this (or undef if no key loaded):
    {
      curve => "ed25519",
      # raw public key as a hexadecimal string
      pub   => "A05D1AEA5830AC9A65CDFB384660D497E3697C46B419CF2CEC85DE8BD245459D",
```

```
    # raw private key as a hexadecimal string. undef if key is public only
    priv  => "45C109BA6FD24E8B67D23EFB6B92D99CD457E2137172C0D749FE2B5A0C142DAD",
}
```

SEE ALSO

- <https://en.wikipedia.org/wiki/EdDSA#Ed25519>
- <https://en.wikipedia.org/wiki/Curve25519>
- <https://tools.ietf.org/html/rfc8032>