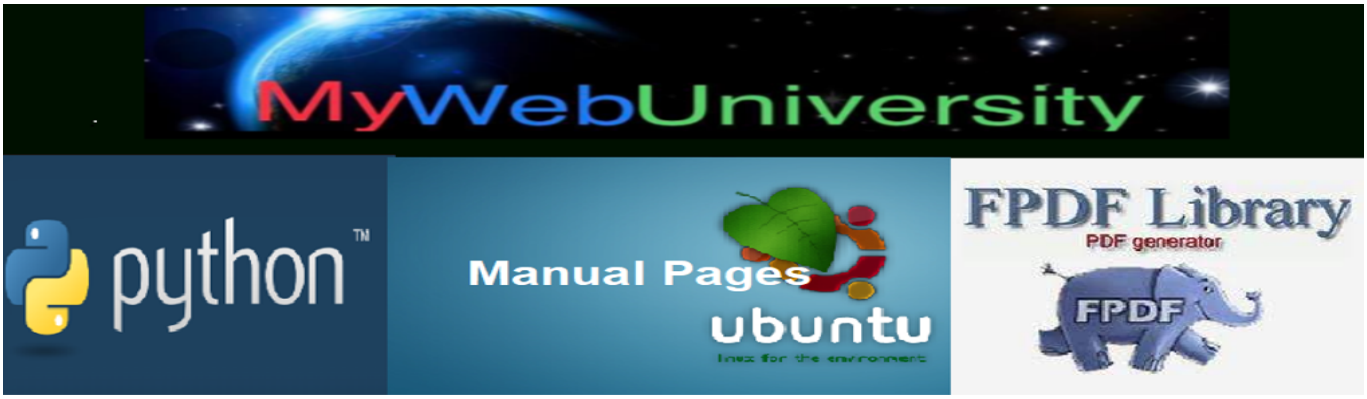

Rocky Enterprise Linux 9.2 Manual Pages on command 'Crypt::PK::X25519.3pm'

$ man Crypt::PK::X25519.3pm

Crypt::PK::X25519(3pm) User Contributed Perl Documentation Crypt::PK::X25519(3pm)

NAME

Crypt::PK::X25519 - Asymmetric cryptography based on X25519

SYNOPSIS

```
use Crypt::PK::X25519;

#Shared secret
my $priv = Crypt::PK::X25519->new('Alice_priv_x25519.der');
my $pub = Crypt::PK::X25519->new('Bob_pub_x25519.der');
my $shared_secret = $priv->shared_secret($pub);

#Load key
my $pk = Crypt::PK::X25519->new;
my $pk_hex = "EA7806F721A8570512C8F6EFB4E8D620C49A529E4DF5EAA77DEC646FB1E87E41";
$pk->import_key_raw(pack("H*", $pk_hex), "public");
my $sk = Crypt::PK::X25519->new;
my $sk_hex = "002F93D10BA5728D8DD8E9527721DABA3261C0BB1BEFDE7B4BBDAC631D454651";
$sk->import_key_raw(pack("H*", $sk_hex), "private");

#Key generation
my $pk = Crypt::PK::X25519->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::X25519->new();
#or
my $pk = Crypt::PK::X25519->new($priv_or_pub_key_filename);
#or
my $pk = Crypt::PK::X25519->new(\$buffer_containing_priv_or_pub_key);
```

Support for password protected PEM keys

```
my $pk = Crypt::PK::X25519->new($priv_pem_key_filename, $password);
#or
my $pk = Crypt::PK::X25519->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 => "x25519",
    pub   => "EA7806F721A8570512C8F6EFB4E8D620C49A529E4DF5EAA77DEC646FB1E87E41",
    priv  => "002F93D10BA5728D8DD8E9527721DABA3261C0BB1BEFDE7B4BBDAC631D454651",
});

# or a hash with items corresponding to JWK (JSON Web Key)
$pk->import_key({
     kty => "OKP",
     crv => "X25519",
     d   => "AC-T0Qulco2N2OlSdyHaujJhwLsb7957S72sYx1FRlE",
     x   => "6ngG9yGoVwUSyPbvtOjWIMSaUp5N9eqnfexkb7HofkE",
});
```

Supported key formats:

```
# all formats can be loaded from a file
my $pk = Crypt::PK::X25519->new($filename);

# or from a buffer containing the key
my $pk = Crypt::PK::X25519->new(\$buffer_with_key);
```

? X25519 private keys in PEM format

```
-----BEGIN X25519 PRIVATE KEY-----
MC4CAQAwBQYDK2VuBCIEIAAvk9ELpXKNjdjpUnch2royYcC7G+/ee0u9rGMdRUZR
-----END X25519 PRIVATE KEY-----
```

? X25519 private keys in password protected PEM format

```
-----BEGIN X25519 PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-CBC,DEEFD3D6B714E75A

dfFWP5bKn49aZ993NVAhQQPdFWgsTb4j8CWhRjGBVTPl6ITstAL17deBIRBwZb7h
pAyIka81Kfs=
-----END X25519 PRIVATE KEY-----
```

? X25519 public keys in PEM format

```
-----BEGIN PUBLIC KEY-----
MCowBQYDK2VuAyEA6ngG9yGoVwUSyPbvtOjWIMSaUp5N9eqnfexkb7HofkE=
-----END PUBLIC KEY-----
```

? PKCS#8 private keys

```
-----BEGIN PRIVATE KEY-----
MC4CAQAwBQYDK2VuBCIEIAAvk9ELpXKNjdjpUnch2royYcC7G+/ee0u9rGMdRUZR
-----END PRIVATE KEY-----
```

? PKCS#8 encrypted private keys

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

MIGHMEsGCSqGSIb3DQEFDTA+MCkGCSqGSIb3DQEFDDAcBAiS0NOFZmjJswICCAAw

DAYIKoZIhvcNAgkFADARBgUrDgMCBwQIGd40Hdso8Y4EONSRCTrqvftl9hl3zbH9

2QmHF1KJ4HDMdLDRxD7EynonCw2SV7BO+XNRHzw2yONqiTybfte7nk9t

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

? X25519 private keys in JSON Web Key (JWK) format

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

{

"kty":"OKP",

"crv":"X25519",

"x":"6ngG9yGoVwUSyPbvtOjWIMSaUp5N9eqnfexkb7HofkE",

"d":"AC-T0Qulco2N2OlSdyHaujJhwLsb7957S72sYx1FRlE",

}

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

? X25519 public keys in JSON Web Key (JWK) format

{

"kty":"OKP",

"crv":"X25519",

"x":"6ngG9yGoVwUSyPbvtOjWIMSaUp5N9eqnfexkb7HofkE",

}

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

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');
```

shared_secret

```
# Alice having her priv key $pk and Bob's public key $pkb
my $pk  = Crypt::PK::X25519->new($priv_key_filename);
my $pkb = Crypt::PK::X25519->new($pub_key_filename);
my $shared_secret = $pk->shared_secret($pkb);

# Bob having his priv key $pk and Alice's public key $pka
my $pk = Crypt::PK::X25519->new($priv_key_filename);
my $pka = Crypt::PK::X25519->new($pub_key_filename);
my $shared_secret = $pk->shared_secret($pka);  # same value as computed by Alice
```

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 => "x25519",
  # raw public key as a hexadecimal string
  pub   => "EA7806F721A8570512C8F6EFB4E8D620C49A529E4DF5EAA77DEC646FB1E87E41",
  # raw private key as a hexadecimal string. undef if key is public only
  priv  => "002F93D10BA5728D8DD8E9527721DABA3261C0BB1BEFDE7B4BBDAC631D454651",
}
```

SEE ALSO

? <https://en.wikipedia.org/wiki/Curve25519>

? <https://tools.ietf.org/html/rfc7748>