



**Full credit is given to the above companies including the Operating System (OS) that this PDF file was generated!**

### ***Rocky Enterprise Linux 9.2 Manual Pages on command 'Crypt::Digest::BLAKE2b\_384.3pm'***

***\$ man Crypt::Digest::BLAKE2b\_384.3pm***

*Crypt::Digest::BLAKE2b\_384(3pm) User Contributed Perl Documentation Crypt::Digest::BLAKE2b\_384(3pm)*

#### **NAME**

*Crypt::Digest::BLAKE2b\_384 - Hash function BLAKE2b [size: 384 bits]*

#### **SYNOPSIS**

*### Functional interface:*

```
use Crypt::Digest::BLAKE2b_384 qw( blake2b_384 blake2b_384_hex blake2b_384_b64 blake2b_384_b64u
    blake2b_384_file blake2b_384_file_hex blake2b_384_file_b64 blake2b_384_file_b64u );
```

*# calculate digest from string/buffer*

```
$blake2b_384_raw = blake2b_384('data string');
```

```
$blake2b_384_hex = blake2b_384_hex('data string');
```

```
$blake2b_384_b64 = blake2b_384_b64('data string');
```

```
$blake2b_384_b64u = blake2b_384_b64u('data string');
```

*# calculate digest from file*

```
$blake2b_384_raw = blake2b_384_file('filename.dat');
```

```
$blake2b_384_hex = blake2b_384_file_hex('filename.dat');
```

```
$blake2b_384_b64 = blake2b_384_file_b64('filename.dat');
```

```
$blake2b_384_b64u = blake2b_384_file_b64u('filename.dat');
```

*# calculate digest from filehandle*

```
$blake2b_384_raw = blake2b_384_file(*FILEHANDLE);
```

```
$blake2b_384_hex = blake2b_384_file_hex(*FILEHANDLE);
```

```
$blake2b_384_b64 = blake2b_384_file_b64(*FILEHANDLE);  
$blake2b_384_b64u = blake2b_384_file_b64u(*FILEHANDLE);
```

### OO interface:

```
use Crypt::Digest::BLAKE2b_384;
```

```
$d = Crypt::Digest::BLAKE2b_384->new;
```

```
$d->add('any data');
```

```
$d->addfile('filename.dat');
```

```
$d->addfile(*FILEHANDLE);
```

```
$result_raw = $d->digest; # raw bytes
```

```
$result_hex = $d->hexdigest; # hexadecimal form
```

```
$result_b64 = $d->b64digest; # Base64 form
```

```
$result_b64u = $d->b64udigest; # Base64 URL Safe form
```

## DESCRIPTION

Provides an interface to the BLAKE2b\_384 digest algorithm.

## EXPORT

Nothing is exported by default.

You can export selected functions:

```
use Crypt::Digest::BLAKE2b_384 qw(blake2b_384 blake2b_384_hex blake2b_384_b64 blake2b_384_b64u  
    blake2b_384_file blake2b_384_file_hex blake2b_384_file_b64 blake2b_384_file_b64u);
```

Or all of them at once:

```
use Crypt::Digest::BLAKE2b_384 ':all';
```

## FUNCTIONS

blake2b\_384

Logically joins all arguments into a single string, and returns its BLAKE2b\_384 digest

encoded as a binary string.

```
$blake2b_384_raw = blake2b_384('data string');
```

```
#or
```

```
$blake2b_384_raw = blake2b_384('any data', 'more data', 'even more data');
```

#### blake2b\_384\_hex

Logically joins all arguments into a single string, and returns its BLAKE2b\_384 digest encoded as a hexadecimal string.

```
$blake2b_384_hex = blake2b_384_hex('data string');
```

```
#or
```

```
$blake2b_384_hex = blake2b_384_hex('any data', 'more data', 'even more data');
```

#### blake2b\_384\_b64

Logically joins all arguments into a single string, and returns its BLAKE2b\_384 digest encoded as a Base64 string, with trailing '=' padding.

```
$blake2b_384_b64 = blake2b_384_b64('data string');
```

```
#or
```

```
$blake2b_384_b64 = blake2b_384_b64('any data', 'more data', 'even more data');
```

#### blake2b\_384\_b64u

Logically joins all arguments into a single string, and returns its BLAKE2b\_384 digest encoded as a Base64 URL Safe string (see RFC 4648 section 5).

```
$blake2b_384_b64url = blake2b_384_b64u('data string');
```

```
#or
```

```
$blake2b_384_b64url = blake2b_384_b64u('any data', 'more data', 'even more data');
```

#### blake2b\_384\_file

Reads file (defined by filename or filehandle) content, and returns its BLAKE2b\_384 digest encoded as a binary string.

```
$blake2b_384_raw = blake2b_384_file('filename.dat');
```

```
#or
```

```
$blake2b_384_raw = blake2b_384_file(*FILEHANDLE);
```

#### blake2b\_384\_file\_hex

Reads file (defined by filename or filehandle) content, and returns its BLAKE2b\_384 digest encoded as a hexadecimal string.

```
$blake2b_384_hex = blake2b_384_file_hex('filename.dat');
```

```
#or
```

```
$blake2b_384_hex = blake2b_384_file_hex(*FILEHANDLE);
```

BEWARE: You have to make sure that the filehandle is in binary mode before you pass it as argument to the addfile() method.

#### blake2b\_384\_file\_b64

Reads file (defined by filename or filehandle) content, and returns its BLAKE2b\_384 digest encoded as a Base64 string, with trailing '=' padding.

```
$blake2b_384_b64 = blake2b_384_file_b64('filename.dat');
```

```
#or
```

```
$blake2b_384_b64 = blake2b_384_file_b64(*FILEHANDLE);
```

#### blake2b\_384\_file\_b64u

Reads file (defined by filename or filehandle) content, and returns its BLAKE2b\_384 digest encoded as a Base64 URL Safe string (see RFC 4648 section 5).

```
$blake2b_384_b64url = blake2b_384_file_b64u('filename.dat');
```

```
#or
```

```
$blake2b_384_b64url = blake2b_384_file_b64u(*FILEHANDLE);
```

The OO interface provides the same set of functions as Crypt::Digest.

new

```
$d = Crypt::Digest::BLAKE2b_384->new();
```

clone

```
$d->clone();
```

reset

```
$d->reset();
```

add

```
$d->add('any data');
```

#or

```
$d->add('any data', 'more data', 'even more data');
```

addfile

```
$d->addfile('filename.dat');
```

#or

```
$d->addfile(*FILEHANDLE);
```

add\_bits

```
$d->add_bits($bit_string); # e.g. $d->add_bits("111100001010");
```

#or

```
$d->add_bits($data, $nbits); # e.g. $d->add_bits("\xF0xA0", 16);
```

hashsize

```
$d->hashsize;
```

#or

```
Crypt::Digest::BLAKE2b_384->hashsize();
```

#or

```
Crypt::Digest::BLAKE2b_384::hashsize();
```

digest

```
$result_raw = $d->digest();
```

hexdigest

```
$result_hex = $d->hexdigest();
```

b64digest

```
$result_b64 = $d->b64digest();
```

b64udigest

```
$result_b64url = $d->b64udigest();
```

SEE ALSO

? CryptX, Crypt::Digest

? <<https://blake2.net/>>

? <<https://tools.ietf.org/html/rfc7693>>

perl v5.34.0

2022-02-06

Crypt::Digest::BLAKE2b\_384(3pm)