crypto

package standard library

Version: go1.20.1 Latest | Published: Feb 14, 2023 | License: BSD-3-Clause | Imports: 3 | Imported by: 26,693

Details ✓ Valid go.mod file ? ✓ Redistributable license ? ✓ Tagged version ?

✓ Stable version ?

Learn more

Repository cs.opensource.google/go/go

Links Report a Vulnerability

Documentation

Overview

Package crypto collects common cryptographic constants.

Index

Constants

This section is empty.

Variables

This section is empty.

Functions

func RegisterHash

```
func RegisterHash(h Hash, f func() hash.Hash)
```

RegisterHash registers a function that returns a new instance of the given hash function. This is intended to be called from the init function in packages that implement hash functions.

Types

type Decrypter

added in go1.5

```
type Decrypter interface {
	// Public returns the public key corresponding to the opaque,
	// private key.
	Public() PublicKey

	// Decrypt decrypts msg. The opts argument should be appropriate for
	// the primitive used. See the documentation in each implementation for
	// details.
	Decrypt(rand io.Reader, msg []byte, opts DecrypterOpts) (plaintext []byte, err erro
}
```

Decrypter is an interface for an opaque private key that can be used for asymmetric decryption operations. An example would be an RSA key kept in a hardware module.

type DecrypterOpts

added in go1.5

```
type DecrypterOpts any
```

type Hash

```
type Hash uint
```

Hash identifies a cryptographic hash function that is implemented in another package.

```
const (
	MD4         Hash = 1 + iota // import golang.org/x/crypto/md4
	MD5                         // import crypto/md5
	SHA1                        // import crypto/sha1
	SHA224                      // import crypto/sha256
	SHA256                      // import crypto/sha256
	SHA384                      // import crypto/sha512
	SHA512                      // import crypto/sha512
	MD5SHA1                     // no implementation; MD5+SHA1 used for TLS RSA
	RIPEMD160                   // import golang.org/x/crypto/ripemd160
	SHA3_224                    // import golang.org/x/crypto/sha3
	SHA3_256                    // import golang.org/x/crypto/sha3
	SHA3_384                    // import golang.org/x/crypto/sha3
	SHA3_512                    // import golang.org/x/crypto/sha3
	SHA512_224                  // import crypto/sha512
	SHA512_256                  // import crypto/sha512
	BLAKE2s_256                 // import golang.org/x/crypto/blake2s
	BLAKE2b_256                 // import golang.org/x/crypto/blake2b
	BLAKE2b_384                 // import golang.org/x/crypto/blake2b
	BLAKE2b_512                 // import golang.org/x/crypto/blake2b

)
```

func (Hash) Available

```
func (h Hash) Available() bool
```

Available reports whether the given hash function is linked into the binary.

func (Hash) HashFunc

added in go1.4

```
func (h Hash) HashFunc() Hash
```

HashFunc simply returns the value of h so that Hash implements SignerOpts.

func (Hash) New

```
func (h Hash) New() hash.Hash
```

New returns a new hash.Hash calculating the given hash function. New panics if the hash function is not linked into the binary.

func (Hash) Size

```
func (h Hash) Size() int
```

Size returns the length, in bytes, of a digest resulting from the given hash function. It doesn't require that the hash function in question be linked into the program.

func (Hash) String

added in go1.15

```
func (h Hash) String() string
```

type PrivateKey

```
type PrivateKey any
```

PrivateKey represents a private key using an unspecified algorithm.

Although this type is an empty interface for backwards compatibility reasons, all private key types in the standard library implement the following interface

```
interface{
    Public() crypto.PublicKey
    Equal(x crypto.PrivateKey) bool
}
```

as well as purpose-specific interfaces such as Signer and Decrypter, which can be used for increased type safety within applications.

type PublicKey

added in go1.2

```
type PublicKey any
```

PublicKey represents a public key using an unspecified algorithm.

Although this type is an empty interface for backwards compatibility reasons, all public key types in the standard library implement the following interface

```
interface{
    Equal(x crypto.PublicKey) bool
}
```

which can be used for increased type safety within applications.

type Signer

added in go1.4

```
type Signer interface {
    // Public returns the public key corresponding to the opaque,
    // private key.
    Public() PublicKey

    // Sign signs digest with the private key, possibly using entropy from
    // rand. For an RSA key, the resulting signature should be either a
    // PKCS #1 v1.5 or PSS signature (as indicated by opts). For an (EC)DSA
    // key, it should be a DER-serialised, ASN.1 signature structure.
    //
    // Hash implements the SignerOpts interface and, in most cases, one can
    // simply pass in the hash function used as opts. Sign may also attempt
    // to type assert opts to other types in order to obtain algorithm
    // specific values. See the documentation in each package for details.
    //
    // Note that when a signature of a hash of a larger message is needed,
    // the caller is responsible for hashing the larger message and passing
    // the hash (as digest) and the hash function (as opts) to Sign.
    Sign(rand io.Reader, digest []byte, opts SignerOpts) (signature []byte, err error)
}
```

Signer is an interface for an opaque private key that can be used for signing operations. For example, an RSA key kept in a hardware module.

type SignerOpts

added in go1.4

```
type SignerOpts interface {
	// HashFunc returns an identifier for the hash function used to produce
	// the message passed to Signer.Sign, or else zero to indicate that no
	// hashing was done.
	HashFunc() Hash
}
```

SignerOpts contains options for signing with a Signer.

Source Files

View all

crypto.go

Directories

Expand all

aes

Package aes implements AES encryption (formerly Rijndael), as defined in U.S. Federal Information Processing Standards Publication 197.

cipher

Package cipher implements standard block cipher modes that can be wrapped around low-level block cipher implementations.

des

Package des implements the Data Encryption Standard (DES) and the Triple Data Encryption Algorithm (TDEA) as defined in U.S. Federal Information Processing Standards Publication 46-3.

dsa

Package dsa implements the Digital Signature Algorithm, as defined in FIPS 186-3.

ecdh

Package ecdh implements Elliptic Curve Diffie-Hellman over NIST curves and Curve25519.

ecdsa

Package ecdsa implements the Elliptic Curve Digital Signature Algorithm, as defined in FIPS 186-4 and SEC 1, Version 2.0.

ed25519

Package ed25519 implements the Ed25519 signature algorithm.

elliptic

Package elliptic implements the standard NIST P-224, P-256, P-384, and P-521 elliptic curves over prime fields.

hmac

Package hmac implements the Keyed-Hash Message Authentication Code (HMAC) as defined in U.S. Federal Information Processing Standards Publication 198.

md5

Package md5 implements the MD5 hash algorithm as defined in RFC 1321.

rand

Package rand implements a cryptographically secure random number generator.

rc4

Package rc4 implements RC4 encryption, as defined in Bruce Schneier's Applied Cryptography.

rsa

Package rsa implements RSA encryption as specified in PKCS #1 and RFC 8017.

sha1

Package sha1 implements the SHA-1 hash algorithm as defined in RFC 3174.

sha256

Package sha256 implements the SHA224 and SHA256 hash algorithms as defined in FIPS 180-4.

sha512

Package sha512 implements the SHA-384, SHA-512, SHA-512/224, and SHA-512/256 hash algorithms as defined in FIPS 180-4.

subtle

Package subtle implements functions that are often useful in cryptographic code but require careful thought to use correctly.

tls

Package tls partially implements TLS 1.2, as specified in RFC 5246, and TLS 1.3, as specified in RFC 8446.

x509

Package x509 implements a subset of the X.509 standard.

internal

Twitter

GitHub

Slack

r/golang

Meetup

Golang Weekly