

[Discover Packages](#) > [Standard library](#) > [crypto](#) > [ecdsa](#) 





ecdsa

package

standard library

Version: [go1.20.1](#) **Latest** | Published: Feb 14, 2023 | License: [BSD-3-Clause](#) | Imports: [19](#) |Imported by: [32,050](#)

Details

[✓ Valid go.mod file](#)  [✓ Redistributable license](#)  [✓ Tagged version](#) [✓ Stable version](#) [Learn more](#)

Repository

cs.opensource.google/go/go

Links

[🛡️ Report a Vulnerability](#) Documentation 

<> Documentation

Overview

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

Signatures generated by this package are not deterministic, but entropy is mixed with the private key and the message, achieving the same level of security in case of randomness source failure.

[▶ Example](#)

Index

```
func Sign(rand io.Reader, priv *PrivateKey, hash []byte) (r, s *big.Int, err error)
func SignASN1(rand io.Reader, priv *PrivateKey, hash []byte) ([]byte, error)
func Verify(pub *PublicKey, hash []byte, r, s *big.Int) bool
func VerifyASN1(pub *PublicKey, hash, sig []byte) bool
type PrivateKey
    func GenerateKey(c elliptic.Curve, rand io.Reader) (*PrivateKey, error)
    func (k *PrivateKey) ECDH() (*ecdh.PrivateKey, error)
    func (priv *PrivateKey) Equal(x crypto.PrivateKey) bool
    func (priv *PrivateKey) Public() crypto.PublicKey
    func (priv *PrivateKey) Sign(rand io.Reader, digest []byte, opts crypto.SignerOpts) ([]byte, error)
type PublicKey
    func (k *PublicKey) ECDH() (*ecdh.PublicKey, error)
    func (pub *PublicKey) Equal(x crypto.PublicKey) bool
```

Examples

Package

Constants

This section is empty.

Variables

This section is empty.

Functions

func Sign

```
func Sign(rand io.Reader, priv *PrivateKey, hash []byte) (r, s *big.Int, err error)
```

Sign signs a hash (which should be the result of hashing a larger message) using the private key, priv. If the hash is longer than the bit-length of the private key's curve order, the hash will be truncated to that length. It returns the signature as a pair of integers. Most applications should use SignASN1 instead of dealing directly with r, s.

func SignASN1

added in go1.15

```
func SignASN1(rand io.Reader, priv *PrivateKey, hash []byte) ([]byte, error)
```

SignASN1 signs a hash (which should be the result of hashing a larger message) using the private key, priv. If the hash is longer than the bit-length of the private key's curve order, the hash will be truncated to that length. It returns the ASN.1 encoded signature.

func Verify

```
func Verify(pub *PublicKey, hash []byte, r, s *big.Int) bool
```

Verify verifies the signature in r, s of hash using the public key, pub. Its return value records whether the signature is valid. Most applications should use VerifyASN1 instead of dealing directly with r, s.

func VerifyASN1

added in go1.15

```
func VerifyASN1(pub *PublicKey, hash, sig []byte) bool
```

VerifyASN1 verifies the ASN.1 encoded signature, sig, of hash using the public key, pub. Its return value records whether the signature is valid.

Types

type PrivateKey

```
type PrivateKey struct {  
    PublicKey  
    D *big.Int  
}
```

PrivateKey represents an ECDSA private key.

func GenerateKey

```
func GenerateKey(c elliptic.Curve, rand io.Reader) (*PrivateKey, error)
```

GenerateKey generates a public and private key pair.

func (*PrivateKey) ECDH

added in go1.20

```
func (k *PrivateKey) ECDH() (*ecdh.PrivateKey, error)
```

ECDH returns k as a [ecdh.PrivateKey](#). It returns an error if the key is invalid according to the definition of [ecdh.Curve.NewPrivateKey](#), or if the Curve is not supported by crypto/ecdh.

func (*PrivateKey) Equal

added in go1.15

```
func (priv *PrivateKey) Equal(x crypto.PrivateKey) bool
```

Equal reports whether priv and x have the same value.

See PublicKey.Equal for details on how Curve is compared.

func (*PrivateKey) Public

added in go1.4

```
func (priv *PrivateKey) Public() crypto.PublicKey
```

Public returns the public key corresponding to priv.

func (*PrivateKey) Sign

added in go1.4

```
func (priv *PrivateKey) Sign(rand io.Reader, digest []byte, opts crypto.SignerOpts)  
([]byte, error)
```

Sign signs digest with priv, reading randomness from rand. The opts argument is not currently used but, in keeping with the crypto.Signer interface, should be the hash function used to digest the message.

This method implements crypto.Signer, which is an interface to support keys where the private part is kept in, for example, a hardware module. Common uses can use the SignASN1 function in this package directly.

type PublicKey

```
type PublicKey struct {  
    elliptic.Curve  
    X, Y *big.Int  
}
```

PublicKey represents an ECDSA public key.

func (*PublicKey) ECDH

added in go1.20

```
func (k *PublicKey) ECDH() (*ecdh.PublicKey, error)
```

ECDH returns k as a [ecdh.PublicKey](#). It returns an error if the key is invalid according to the definition of [ecdh.Curve.NewPublicKey](#), or if the Curve is not supported by crypto/ecdh.

func (*PublicKey) Equal

added in go1.15

```
func (pub *PublicKey) Equal(x crypto.PublicKey) bool
```

Equal reports whether pub and x have the same value.

Two keys are only considered to have the same value if they have the same Curve value. Note that for example `elliptic.P256()` and `elliptic.P256().Params()` are different values, as the latter is a generic not constant time implementation.



Source Files

[View all](#)

[ecdsa.go](#)

[ecdsa_legacy.go](#)

[ecdsa_noasm.go](#)

[notboring.go](#)

Why Go

[Use Cases](#)

[Case Studies](#)

Get Started

[Playground](#)

[Tour](#)

[Stack Overflow](#)

[Help](#)

Packages

[Standard Library](#)

[About Go Packages](#)

About

[Download](#)

[Blog](#)

[Issue Tracker](#)

[Release Notes](#)

[Brand Guidelines](#)

[Code of Conduct](#)

Connect

[Twitter](#)

[GitHub](#)

[Slack](#)

[r/golang](#)

[Meetup](#)

[Golang Weekly](#)

[Copyright](#)

[Terms of Service](#)

[Privacy Policy](#)

[Report an Issue](#)



[Google](#)