elliptic

package | standard library

Version: go1.20.1 Latest | Published: Feb 14, 2023 | License: BSD-3-Clause | Imports: 5 | Imported by: 16,882

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

Rendered for linux/amd64

Overview

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

The P224(), P256(), P384() and P521() values are necessary to use the crypto/ecdsa package. Most other uses should migrate to the more efficient and safer crypto/ecdh package.

Index

Constants

This section is empty.

Variables

This section is empty.

Functions

func GenerateKey

```
func GenerateKey(curve Curve, rand io.Reader) (priv []byte, x, y *big.Int, err error)
```

GenerateKey returns a public/private key pair. The private key is generated using the given reader, which must return random data.

Note: for ECDH, use the GenerateKey methods of the crypto/ecdh package; for ECDSA, use the GenerateKey function of the crypto/ecdsa package.

func Marshal

```
func Marshal(curve Curve, x, y *big.Int) []byte
```

Marshal converts a point on the curve into the uncompressed form specified in SEC 1, Version 2.0, Section 2.3.3. If the point is not on the curve (or is the conventional point at infinity), the behavior is undefined.

Note: for ECDH, use the crypto/ecdh package. This function returns an encoding equivalent to that of PublicKey.Bytes in crypto/ecdh.

func MarshalCompressed

added in go1.15

```
func MarshalCompressed(curve Curve, x, y *big.Int) []byte
```

MarshalCompressed converts a point on the curve into the compressed form specified in SEC 1, Version 2.0, Section 2.3.3. If the point is not on the curve (or is the conventional point at infinity), the behavior is undefined.

func Unmarshal

```
func Unmarshal(curve Curve, data []byte) (x, y *big.Int)
```

Unmarshal converts a point, serialized by Marshal, into an x, y pair. It is an error if the point is not in uncompressed form, is not on the curve, or is the point at infinity. On error, x = nil.

Note: for ECDH, use the crypto/ecdh package. This function accepts an encoding equivalent to that of the NewPublicKey methods in crypto/ecdh.

func UnmarshalCompressed

added in go1.15

```
func UnmarshalCompressed(curve Curve, data []byte) (x, y *big.Int)
```

UnmarshalCompressed converts a point, serialized by MarshalCompressed, into an x, y pair. It is an error if the point is not in compressed form, is not on the curve, or is the point at infinity. On error, x = nil.

Types

type Curve

```
type Curve interface {
	// Params returns the parameters for the curve.
	Params() *CurveParams

	// IsOnCurve reports whether the given (x,y) lies on the curve.
	//
	// Note: this is a low-level unsafe API. For ECDH, use the crypto/ecdh
	// package. The NewPublicKey methods of NIST curves in crypto/ecdh accept
	// the same encoding as the Unmarshal function, and perform on-curve checks.
	IsOnCurve(x, y *big.Int) bool

	// Add returns the sum of (x1,y1) and (x2,y2).
	//
	// Note: this is a low-level unsafe API.
	Add(x1, y1, x2, y2 *big.Int) (x, y *big.Int)

	// Double returns 2*(x,y).
	//
	// Note: this is a low-level unsafe API.
	Double(x1, y1 *big.Int) (x, y *big.Int)

	// ScalarMult returns k*(x,y) where k is an integer in big-endian form.
	//
	// Note: this is a low-level unsafe API. For ECDH, use the crypto/ecdh
	// package. Most uses of ScalarMult can be replaced by a call to the ECDH
	// methods of NIST curves in crypto/ecdh.
	ScalarMult(x1, y1 *big.Int, k []byte) (x, y *big.Int)

	// ScalarBaseMult returns k*G, where G is the base point of the group
	// and k is an integer in big-endian form.
	//
	// Note: this is a low-level unsafe API. For ECDH, use the crypto/ecdh
	// package. Most uses of ScalarBaseMult can be replaced by a call to the
	// PrivateKey.PublicKey method in crypto/ecdh.
	ScalarBaseMult(k []byte) (x, y *big.Int)
}
```

A Curve represents a short-form Weierstrass curve with a=-3.

The behavior of Add, Double, and ScalarMult when the input is not a point on the curve is undefined.

Note that the conventional point at infinity (0, 0) is not considered on the curve, although it can be returned by Add, Double, ScalarMult, or ScalarBaseMult (but not the Unmarshal or UnmarshalCompressed functions).

func P224

```
func P224() Curve
```

P224 returns a Curve which implements NIST P-224 (FIPS 186-3, section D.2.2), also known as secp224r1. The CurveParams.Name of this Curve is "P-224".

Multiple invocations of this function will return the same value, so it can be used for equality checks and switch statements.

The cryptographic operations are implemented using constant-time algorithms.

func P256

```
func P256() Curve
```

P256 returns a Curve which implements NIST P-256 (FIPS 186-3, section D.2.3), also known as secp256r1 or prime256v1. The CurveParams.Name of this Curve is "P-256".

Multiple invocations of this function will return the same value, so it can be used for equality checks and switch statements.

The cryptographic operations are implemented using constant-time algorithms.

func P384

```
func P384() Curve
```

P384 returns a Curve which implements NIST P-384 (FIPS 186-3, section D.2.4), also known as secp384r1. The CurveParams.Name of this Curve is "P-384".

Multiple invocations of this function will return the same value, so it can be used for equality checks and switch statements.

The cryptographic operations are implemented using constant-time algorithms.

func P521

```
func P521() Curve
```

P521 returns a Curve which implements NIST P-521 (FIPS 186-3, section D.2.5), also known as secp521r1. The CurveParams.Name of this Curve is "P-521".

Multiple invocations of this function will return the same value, so it can be used for equality checks and switch statements.

The cryptographic operations are implemented using constant-time algorithms.

type CurveParams

```
type CurveParams struct {
	P       *big.Int // the order of the underlying field
	N       *big.Int // the order of the base point
	B       *big.Int // the constant of the curve equation
	Gx, Gy  *big.Int // (x,y) of the base point
	BitSize int      // the size of the underlying field
	Name    string   // the canonical name of the curve
}
```

CurveParams contains the parameters of an elliptic curve and also provides a generic, non-constant time implementation of Curve.

Note: Custom curves (those not returned by P224(), P256(), P384(), and P521()) are not guaranteed to provide any security property.

func (*CurveParams) Add

```
func (curve *CurveParams) Add(x1, y1, x2, y2 *big.Int) (*big.Int, *big.Int)
```

Add implements Curve.Add.

Note: the CurveParams methods are not guaranteed to provide any security property. For ECDH, use the crypto/ecdh package. For ECDSA, use the crypto/ecdsa package with a Curve value returned directly from P224(), P256(), P384(), or P521().

func (*CurveParams) Double

```
func (curve *CurveParams) Double(x1, y1 *big.Int) (*big.Int, *big.Int)
```

Double implements Curve.Double.

Note: the CurveParams methods are not guaranteed to provide any security property. For ECDH, use the crypto/ecdh package. For ECDSA, use the crypto/ecdsa package with a Curve value returned directly from P224(), P256(), P384(), or P521().

func (*CurveParams) IsOnCurve

```
func (curve *CurveParams) IsOnCurve(x, y *big.Int) bool
```

IsOnCurve implements Curve.IsOnCurve.

Note: the CurveParams methods are not guaranteed to provide any security property. For ECDH, use the crypto/ecdh package. For ECDSA, use the crypto/ecdsa package with a Curve value returned directly from P224(), P256(), P384(), or P521().

func (*CurveParams) Params

```
func (curve *CurveParams) Params() *CurveParams
```

func (*CurveParams) ScalarBaseMult

```
func (curve *CurveParams) ScalarBaseMult(k []byte) (*big.Int, *big.Int)
```

ScalarBaseMult implements Curve.ScalarBaseMult.

Note: the CurveParams methods are not guaranteed to provide any security property. For ECDH, use the crypto/ecdh package. For ECDSA, use the crypto/ecdsa package with a Curve value returned directly from P224(), P256(), P384(), or P521().

func (*CurveParams) ScalarMult

```
func (curve *CurveParams) ScalarMult(Bx, By *big.Int, k []byte) (*big.Int, *big.Int)
```

ScalarMult implements Curve.ScalarMult.

Note: the CurveParams methods are not guaranteed to provide any security property. For ECDH, use the crypto/ecdh package. For ECDSA, use the crypto/ecdsa package with a Curve value returned directly from P224(), P256(), P384(), or P521().

Source Files

View all

elliptic.go
nistec.go
nistec_p256.go
params.go

Twitter

GitHub

Slack

r/golang

Meetup

Golang Weekly