Contents

1 Module `Ecc` : An elliptic curve point.

It is either infinity or a point (x,y).

```
module Ecc :
  sig
```

```
type point =
  | Infinity
  | Point of Z.t * Z.t
```

An elliptic curve point. It is either infinity or a point (x,y).

```
type elliptic_curve = {
  p : Z.t ;
  a : Z.t ;
  b : Z.t ;
  g : point ;
  n : Z.t ;
  h : Z.t ;
}
```

The type of domain parameters

```
val inverse : Z.t -> Z.t -> Z.t
```

Ecc.inverse a n inverses the number a modulo n

```
val verify_range : Z.t -> Z.t -> Z.t -> bool
```

Ecc.verify_range a l h returns true if l $\leq$ a $\leq$ h or false otherwise

```
val is_point : point -> elliptic_curve -> bool
```

Returns true if a point belongs to an elliptic curve or false otherwise

```
val double_point : point -> elliptic_curve -> point
```

Given a point P on an elliptic curve and the elliptic curve retuns the point 2P on that curve.

```
val add_point : point -> point -> elliptic_curve -> point
```

Given two points P and Q, both on the same elliptic curve, and the elliptic curve returns P+Q on that curve.

```
val multiply_point : point -> Z.t -> elliptic_curve -> point
```

Given a point P on an elliptic curve, an integer k and the elliptic curve returns the scalar multiplication kP on that curve.

```
val integer_of_octet : string -> int
val octList_of_octStr : string -> string list
val integer_of_octStr : string -> Z.t
val brainpool_P256_r1 : elliptic_curve
```

An elliptic curve used for ECC as defined by Brainpool

```
val test_curve : elliptic_curve
```

An elliptic curve with small domain parameters for testing purposes

```
val random_big_int : Z.t -> Z.t
```

Ecc.random_big_int bound returns a random integer in `1, bound-1`

```
val sign : string -> Z.t -> elliptic_curve -> Z.t * Z.t
```

Ecc.sign message sk curve where sk is secret key of the user s and curve the public elliptic curve, returns the signature (r, s) of the message.

```
val verify : string -> Z.t * Z.t -> point -> elliptic_curve -> bool
```

Ecc.verify message (r, s) pk curve where pk is the public key of the user who signed the message, returns true if the (r, s) is a valid signature or false otherwise.

```
val create_keys : elliptic_curve -> point * Z.t
```

Creates a tuple (public_key, secret_key) where public_key is a point of the curve and secret_key an integer.

```
end
```