ECDSA and EdDSA algorithms

Youssef Lamriji[1], Khalid El Makkaoui[2], and Abderrahim Beni-Hssane[1]

[1]LAROSERI laboratory, FS, Univ Chouaib Doukkali, B.P. 20, 24000, El Jadida, Morocco
youssef1amr2018@gmail.com, abenihssane@yahoo.fr

[2]LaMAO laboratory, MSC team, FPD, Mohammed First University, Nador, Morocco
kh.elmakkaoui@gmail.com

I. ECDSA algorithm

The ECDSA is an EC analog of the DSA that was first proposed in 1992 by Scott Vanstone. It is a better-known algorithm in the digital signature, especially in BC apps like Bitcoin, Ethereum, Ripple, and others.

ECDSA uses a curve (E_{ECDSA}) defined over a finite field F_p with $p > 3$ is a prime number, which consists of the points satisfying the equation:

$$y^2 \equiv x^3 + ax + b \ (mod\ p) \tag{1}$$

where $a, b \in F_p$ such that $4a^3 + 27b^2 \not\equiv 0 \ (mod\ p)$.

ECDSA takes place in three phases namely key generation, signature generation and verification.

Now, we give three constituent algorithms of ECDSA: key generation (Algorithm 1), signature generation (Algorithm 2) and verification (Algorithm 3).

Algorithm 1 ECDSA key generation

Input: a prime p and coefficients (a, b) of E_{ECDSA}.
Generate G, which generates a cyclic group of prime order n.
Choose a random integer $x \in$ [1,n-1].
Compute $\boxed{B = xG}$.
Output: K_{pub} = (p, a, b, n, G, B) and $K_{pr} = (x)$.

Algorithm 2 ECDSA signature generation

Input: K_{pub}, K_{pr}, and H = hash of message .
$r\ ,\ s \leftarrow 0\ ,\ 0$
while $r = 0$ or $s = 0$ **do**
  Choose a random integer $k \in$ [1,n-1].
  Compute $\boxed{R = kG}$
  if $r \neq 0$ **then**
    $s = \boxed{k^{-1}}\ (H + rK_{pr})\ (mod\ n)$
    if $s \neq 0$ **then**
      return (r, s)
    else
      $r \leftarrow 0$
    end if
  end if
end while
Output: (r, s)

Algorithm 3 ECDSA verification

Input: K_{pub}, (r, s), and H = hash of message.
if $r, s \notin$ [1,n-1] **then**
  return Signature Invalid
end if
Compute $w = \boxed{s^{-1}}\ (mod\ n)$
Compute $u_1 = Hw\ (mod\ n)$ and $u_2 = rw\ (mod\ n)$
Compute $X = \boxed{u_1 G + u_2 B}$
if X = point at infinity **then**
  return Signature Invalid
end if
if $X_x = r$ **then**
  return Signature Valid
else
  return Signature Invalid
end if
Output: *valid* or *invalid*.

II. EdDSA algorithm

EdDSA is a modern and secure algorithm based on performance-optimized ECs. It was proposed by Bernstein et al. to perform fast public-key digital signatures as ECDSA. EdDSA uses the two forms of Edwards: edwards25519 (255-bit curve) and edwards448 (448-bit curve). It has been used in many products and libraries, such as OpenSSH and some cryptocurrencies.

The E_{EdDSA} is an EC defined over a finite field F_p with $p = 2^{255} - 19$ (for edwards25519) and $p = 2^{448} - 2^{224} - 1$ (for edwards448), which consists of the points satisfying the equation:

$$ax^2 + y^2 \equiv 1 + dx^2y^2 \ (mod\ p) \tag{2}$$

where $a, d \in F_p$ such that $a \neq 0$, $d \neq 0$ and $a \neq d$.

In what follows, we describe three constituent algorithms of EdDSA, namely key generation (Algorithm 4), signature generation (Algorithm 5) and verification (Algorithm 6).

Algorithm 4 EdDSA key generation

Input: a prime p and coefficients (a, d) of E_{EdDSA}.

Generate G, which generates a cyclic group of prime order n.
Choose a random integer $x \in$ [1,n-1].
Compute $\boxed{B = xG}$.

Output: K_{pub} = (p, a, d, n, G, B) and $K_{pr} = (x)$.

Algorithm 5 EdDSA signing

Input: K_{pub}, K_{pr}, and $H =$ hash of message.

Compute $r = H\ (mod\ n)$
Compute $\boxed{R = rG}$
Compute $h = Hash(R,\ B,\ H)$
Compute $s = (s + hK_{pr})\ (mod\ n)$
return (R, s)

Output: (R, s)

Algorithm 6 EdDSA verification

Input: K_{pub}, (R, s), and $H =$ hash of message.

if $s \notin$ [1,n-1] **then**
    return Signature Invalid
end if
Compute $h = Hash(R,\ K_{pub},\ H)$
Compute $\boxed{P_1 = sG}$
Compute $\boxed{P_2 = R + hB}$
if $P_1 = P_2$ **then**
    return Signature Valid
else
    return Signature Invalid
end if

Output: *valid* or *invalid*.
