EdDSA considerations

Cofactor multiplication and batch verification

Tanja Lange

Eindhoven University of Technology

2MMC10 – Cryptology

EdDSA – Ed25519

Let $p = 2^{255} - 19, d = -121665/121666$ and

$$E : -x^2 + y^2 = 1 + dx^2y^2.$$

Base point P has prime order ℓ, $|E(\mathbf{F}_p)| = 8\ell$.

Scheme follows Schnorr, with some improvements:

- Put $h = H(R, Q, m)$ to reduce multi-target attacks.
- Verify $8sP = 8R + 8hQ$ to deal with cofactor (can also check without 8).
- Choose r pseudorandomly to avoid issues with bad randomness.

EdDSA – Ed25519

Let $p = 2^{255} - 19, d = -121665/121666$ and

$$E : -x^2 + y^2 = 1 + dx^2y^2.$$

Base point P has prime order ℓ, $|E(\mathbf{F}_p)| = 8\ell$.

Scheme follows Schnorr, with some improvements:

- Put $h = H(R, Q, m)$ to reduce multi-target attacks.
- Verify $8sP = 8R + 8hQ$ to deal with cofactor (can also check without 8).
- Choose r pseudorandomly to avoid issues with bad randomness.
- Signing equation ensures $sP = R + hQ \Rightarrow 8sP = 8R + 8hQ$. However, $8sP = 8R + 8hQ$ does not imply $sP = R + hQ$:

EdDSA – Ed25519

Let $p = 2^{255} - 19, d = -121665/121666$ and

$$E : -x^2 + y^2 = 1 + dx^2y^2.$$

Base point P has prime order ℓ, $|E(\mathbf{F}_p)| = 8\ell$.

Scheme follows Schnorr, with some improvements:

- Put $h = H(R, Q, m)$ to reduce multi-target attacks.
- Verify $8sP = 8R + 8hQ$ to deal with cofactor (can also check without 8).
- Choose r pseudorandomly to avoid issues with bad randomness.
- Signing equation ensures $sP = R + hQ \Rightarrow 8sP = 8R + 8hQ$. However, $8sP = 8R + 8hQ$ does not imply $sP = R + hQ$: Let $R' = rP + P_8$, where P_8 is a point of order 8. Use R' in $h = H(R', Q, m)$, compute $s \equiv r + ha \bmod \ell$, and put (R', s) as signature. Then

EdDSA – Ed25519

Let $p = 2^{255} - 19, d = -121665/121666$ and

$$E : -x^2 + y^2 = 1 + dx^2y^2.$$

Base point P has prime order ℓ, $|E(\mathbf{F}_p)| = 8\ell$.

Scheme follows Schnorr, with some improvements:

- Put $h = H(R, Q, m)$ to reduce multi-target attacks.
- Verify $8sP = 8R + 8hQ$ to deal with cofactor (can also check without 8).
- Choose r pseudorandomly to avoid issues with bad randomness.
- Signing equation ensures $sP = R + hQ \Rightarrow 8sP = 8R + 8hQ$. However, $8sP = 8R + 8hQ$ does not imply $sP = R + hQ$: Let $R' = rP + P_8$, where P_8 is a point of order 8. Use R' in $h = H(R', Q, m)$, compute $s \equiv r + ha \bmod \ell$, and put (R', s) as signature. Then $8sP = 8(r + ha)P = 8R + 8hQ = 8R + (0, 1) + 8hQ = 8R + 8P_8 + 8hQ = 8R' + 8hQ$. Thus (R', s) verifies. Note, this did need a in signing.

What is going on here?

$8s \equiv 8r + 8ha \bmod \ell$ implies $s \equiv r + ha \bmod \ell$
as $\gcd(8, \ell) = 1$ for large prime ℓ.

What is going on here?

$8s \equiv 8r + 8ha \bmod \ell$ implies $s \equiv r + ha \bmod \ell$
as $\gcd(8, \ell) = 1$ for large prime ℓ.

But $8s \equiv 8r + 8ha \bmod 8\ell$ does not imply $s \equiv r + ha \bmod \ell$ and that's what we're checking in $8sP = 8R + 8hQ$ unless we test that R and Q have order ℓ (rather than $2^i\ell$ for $i \in \{1, 2, 3\}$).

What is going on here?

$8s \equiv 8r + 8ha \bmod \ell$ implies $s \equiv r + ha \bmod \ell$
as $\gcd(8, \ell) = 1$ for large prime ℓ.

But $8s \equiv 8r + 8ha \bmod 8\ell$ does not imply $s \equiv r + ha \bmod \ell$ and that's what we're checking in $8sP = 8R + 8hQ$ unless we test that R and Q have order ℓ (rather than $2^i\ell$ for $i \in \{1, 2, 3\}$).

Note that we could have tweaked Q to $Q' = Q + P_8$ and produced a signature under Q' using $s \equiv r + ha \bmod \ell$.

So, tweaked Alice with Q' could use Alice to generate signatures without herself knowing a? This is not quite CMA as the key differs but would be undesirable.

What is going on here?

$8s \equiv 8r + 8ha \bmod \ell$ implies $s \equiv r + ha \bmod \ell$
as $\gcd(8, \ell) = 1$ for large prime ℓ.

But $8s \equiv 8r + 8ha \bmod 8\ell$ does not imply $s \equiv r + ha \bmod \ell$ and that's what we're checking in $8sP = 8R + 8hQ$ unless we test that R and Q have order ℓ (rather than $2^i\ell$ for $i \in \{1, 2, 3\}$).

Note that we could have tweaked Q to $Q' = Q + P_8$ and produced a signature under Q' using $s \equiv r + ha \bmod \ell$.

So, tweaked Alice with Q' could use Alice to generate signatures without herself knowing a? This is not quite CMA as the key differs but would be undesirable.

No, Q' is included in $h = H(R, Q', m)$ and Alice would use Q.

Neither of these is an attack as h fixes R and Q.
But why include the 8 in verifying? First some interlude.

Prelude to batch verification

Assume Bob needs to verify many signatures and signatures are typically correct. No, do not slack off!

Prelude to batch verification

Assume Bob needs to verify many signatures and signatures are typically correct. No, do not slack off!

Write (m_i, Q_i, R_i, s_i) for the ith signature (R_i, s_i), which is on message m_i under public key Q_i.

Need to compute $h_i = H(R_i, Q_i, m_i)$ for each of them.
But can combine checking $s_1P = R_1 + h_1Q_1, s_2P = R_2 + h_2Q_2$:

$$(s_1 + s_2)P \stackrel{?}{=} R_1 + R_2 + h_1Q_1 + h_2Q_2$$

computed as $(s_1 + s_2)P - h_1Q_1 - h_2Q_2 \stackrel{?}{=} R_1 + R_2$ with one triple scalar multiplication and an addition.

Prelude to batch verification

Assume Bob needs to verify many signatures and signatures are typically correct. No, do not slack off!

Write (m_i, Q_i, R_i, s_i) for the ith signature (R_i, s_i), which is on message m_i under public key Q_i.

Need to compute $h_i = H(R_i, Q_i, m_i)$ for each of them.
But can combine checking $s_1 P = R_1 + h_1 Q_1, s_2 P = R_2 + h_2 Q_2$:

$$(s_1 + s_2)P \stackrel{?}{=} R_1 + R_2 + h_1 Q_1 + h_2 Q_2$$

computed as $(s_1 + s_2)P - h_1 Q_1 - h_2 Q_2 \stackrel{?}{=} R_1 + R_2$ with one triple scalar multiplication and an addition.

Problem: Eve can sign as Alice!

Prelude to batch verification

Assume Bob needs to verify many signatures and signatures are typically correct. No, do not slack off!

Write (m_i, Q_i, R_i, s_i) for the ith signature (R_i, s_i), which is on message m_i under public key Q_i.

Need to compute $h_i = H(R_i, Q_i, m_i)$ for each of them.
But can combine checking $s_1P = R_1 + h_1Q_1, s_2P = R_2 + h_2Q_2$:

$$(s_1 + s_2)P \stackrel{?}{=} R_1 + R_2 + h_1Q_1 + h_2Q_2$$

computed as $(s_1 + s_2)P - h_1Q_1 - h_2Q_2 \stackrel{?}{=} R_1 + R_2$ with one triple scalar multiplication and an addition.

Problem: Eve can sign as Alice!
$(m_1, Q_A, rQ_A, s_1), (m_2, Q_E = eP, -(r + h_1)Q_A, eh_2 - s_1 \bmod \ell)$
for random r, s_1 verify when batched:

Prelude to batch verification

Assume Bob needs to verify many signatures and signatures are typically correct. No, do not slack off!

Write (m_i, Q_i, R_i, s_i) for the ith signature (R_i, s_i), which is on message m_i under public key Q_i.

Need to compute $h_i = H(R_i, Q_i, m_i)$ for each of them.
But can combine checking $s_1P = R_1 + h_1Q_1, s_2P = R_2 + h_2Q_2$:

$$(s_1 + s_2)P \stackrel{?}{=} R_1 + R_2 + h_1Q_1 + h_2Q_2$$

computed as $(s_1 + s_2)P - h_1Q_1 - h_2Q_2 \stackrel{?}{=} R_1 + R_2$ with one triple scalar multiplication and an addition.

Problem: Eve can sign as Alice!
$(m_1, Q_A, rQ_A, s_1), (m_2, Q_E = eP, -(r + h_1)Q_A, eh_2 - s_1 \bmod \ell)$
for random r, s_1 verify when batched:
$(s_1 + s_2)P = (s_1 + eh_2 - s_1)P = eh_2P =$
$rQ_A - rQ_A - h_1Q_A + h_1Q_A + eh_2P = R_1 + R_2 + h_1Q_A + h_2Q_E$

Batch verification

Write (m_i, Q_i, R_i, s_i) for the ith signature (R_i, s_i), which is on message m_i under public key Q_i.

Need to compute $h_i = H(R_i, Q_i, m_i)$ for each of them.
But can combine checking $s_1P = R_1 + h_1Q_1, s_2P = R_2 + h_2Q_2$:

$$(z_1s_1 + z_2s_2)P \stackrel{?}{=} z_1R_1 + z_2R_2 + z_1h_1Q_1 + z_2h_2Q_2$$

using randomly chosen z_1, z_2.

Batch verification

Write (m_i, Q_i, R_i, s_i) for the ith signature (R_i, s_i), which is on message m_i under public key Q_i.

Need to compute $h_i = H(R_i, Q_i, m_i)$ for each of them.
But can combine checking $s_1 P = R_1 + h_1 Q_1, s_2 P = R_2 + h_2 Q_2$:

$$(z_1 s_1 + z_2 s_2)P \stackrel{?}{=} z_1 R_1 + z_2 R_2 + z_1 h_1 Q_1 + z_2 h_2 Q_2$$

using randomly chosen z_1, z_2.

Important: verifier chooses z_1, z_2, no flexibility for Eve.

What does this prove?

Batch verification

Write (m_i, Q_i, R_i, s_i) for the ith signature (R_i, s_i), which is on message m_i under public key Q_i.

Need to compute $h_i = H(R_i, Q_i, m_i)$ for each of them.
But can combine checking $s_1 P = R_1 + h_1 Q_1, s_2 P = R_2 + h_2 Q_2$:

$$(z_1 s_1 + z_2 s_2)P \stackrel{?}{=} z_1 R_1 + z_2 R_2 + z_1 h_1 Q_1 + z_2 h_2 Q_2$$

using randomly chosen z_1, z_2.

Important: verifier chooses z_1, z_2, no flexibility for Eve.

What does this prove?
Let $T_i = R_i + h_i Q_i - s_i P$, then verification implies
$z_1 T_1 + z_2 T_2 = (0, 1)$ for verifier-chosen z_i.

Batch verification

Write (m_i, Q_i, R_i, s_i) for the ith signature (R_i, s_i), which is on message m_i under public key Q_i.

Need to compute $h_i = H(R_i, Q_i, m_i)$ for each of them.
But can combine checking $s_1 P = R_1 + h_1 Q_1, s_2 P = R_2 + h_2 Q_2$:

$$(z_1 s_1 + z_2 s_2)P \stackrel{?}{=} z_1 R_1 + z_2 R_2 + z_1 h_1 Q_1 + z_2 h_2 Q_2$$

using randomly chosen z_1, z_2.

Important: verifier chooses z_1, z_2, no flexibility for Eve.

What does this prove?
Let $T_i = R_i + h_i Q_i - s_i P$, then verification implies
$z_1 T_1 + z_2 T_2 = (0, 1)$ for verifier-chosen z_i.
R_i and Q_i may not have order ℓ, so the T_i could have order 8 or less $\Rightarrow \geq 1/8$ chance of holding for random z_i and such $T_i \neq (0, 1)$.

Batch verification

Write (m_i, Q_i, R_i, s_i) for the ith signature (R_i, s_i), which is on message m_i under public key Q_i.

Need to compute $h_i = H(R_i, Q_i, m_i)$ for each of them.
But can combine checking $s_1 P = R_1 + h_1 Q_1, s_2 P = R_2 + h_2 Q_2$:

$$(z_1 s_1 + z_2 s_2)P \stackrel{?}{=} z_1 R_1 + z_2 R_2 + z_1 h_1 Q_1 + z_2 h_2 Q_2$$

using randomly chosen z_1, z_2.

Important: verifier chooses z_1, z_2, no flexibility for Eve.

What does this prove?
Let $T_i = R_i + h_i Q_i - s_i P$, then verification implies
$z_1 T_1 + z_2 T_2 = (0, 1)$ for verifier-chosen z_i.
R_i and Q_i may not have order ℓ, so the T_i could have order 8 or less $\Rightarrow \geq 1/8$ chance of holding for random z_i and such $T_i \neq (0, 1)$.

Let $T_i' = 8R_i + 8h_i Q_i - 8s_i P$, then verification implies
$z_1 T_1' + z_2 T_2' = (0, 1)$ for T_i' in a group of order ℓ

Batch verification

Write (m_i, Q_i, R_i, s_i) for the ith signature (R_i, s_i), which is on message m_i under public key Q_i.

Need to compute $h_i = H(R_i, Q_i, m_i)$ for each of them.
But can combine checking $s_1 P = R_1 + h_1 Q_1, s_2 P = R_2 + h_2 Q_2$:

$$(z_1 s_1 + z_2 s_2)P \stackrel{?}{=} z_1 R_1 + z_2 R_2 + z_1 h_1 Q_1 + z_2 h_2 Q_2$$

using randomly chosen z_1, z_2.

Important: verifier chooses z_1, z_2, no flexibility for Eve.

What does this prove?
Let $T_i = R_i + h_i Q_i - s_i P$, then verification implies
$z_1 T_1 + z_2 T_2 = (0, 1)$ for verifier-chosen z_i.
R_i and Q_i may not have order ℓ, so the T_i could have order 8 or less $\Rightarrow \geq 1/8$ chance of holding for random z_i and such $T_i \neq (0, 1)$.

Let $T_i' = 8R_i + 8h_i Q_i - 8s_i P$, then verification implies
$z_1 T_1' + z_2 T_2' = (0, 1)$ for T_i' in a group of order $\ell \Rightarrow 1/\ell$ chance of holding for random z_i and $T_i' \neq (0, 1)$,

Batch verification

Write (m_i, Q_i, R_i, s_i) for the ith signature (R_i, s_i), which is on message m_i under public key Q_i.

Need to compute $h_i = H(R_i, Q_i, m_i)$ for each of them.
But can combine checking $s_1 P = R_1 + h_1 Q_1, s_2 P = R_2 + h_2 Q_2$:

$$(z_1 s_1 + z_2 s_2)P \stackrel{?}{=} z_1 R_1 + z_2 R_2 + z_1 h_1 Q_1 + z_2 h_2 Q_2$$

using randomly chosen z_1, z_2.

Important: verifier chooses z_1, z_2, no flexibility for Eve.

What does this prove?
Let $T_i = R_i + h_i Q_i - s_i P$, then verification implies
$z_1 T_1 + z_2 T_2 = (0, 1)$ for verifier-chosen z_i.
R_i and Q_i may not have order ℓ, so the T_i could have order 8 or less $\Rightarrow \geq 1/8$ chance of holding for random z_i and such $T_i \neq (0, 1)$.

Let $T'_i = 8R_i + 8h_i Q_i - 8s_i P$, then verification implies
$z_1 T'_1 + z_2 T'_2 = (0, 1)$ for T'_i in a group of order $\ell \Rightarrow 1/\ell$ chance of holding for random z_i and $T'_i \neq (0, 1)$, i.e., $T'_i = (0, 1)$.

Batch verification for Ed25519

Assume Bob needs to verify many signatures (m_i, Q_i, R_i, s_i).

1. Compute $h_i = H(R_i, Q_i, m_i)$ for each of them.
2. Pick random integers $z_i < 2^{128}$.
   Good enough for failure probability and cheaper to handle.
3. Compute batch verification as

$$-\left(\sum z_i s_i \bmod \ell\right) P + \sum (z_i h_i \bmod \ell) Q_i + \sum z_i R_i \stackrel{?}{=} (0,1)$$

   with one multi-scalar multiplication (Bos–Coster algorithm).
4. if failure, check signatures individually/in smaller batches.

For k signatures these are $k+1$ scalars of size ℓ and k of size 2^{128}.

Batch verification for Ed25519

Assume Bob needs to verify many signatures (m_i, Q_i, R_i, s_i).

1. Compute $h_i = H(R_i, Q_i, m_i)$ for each of them.
2. Pick random integers $z_i < 2^{128}$.
   Good enough for failure probability and cheaper to handle.
3. Compute batch verification as

$$-\left(\sum z_i s_i \bmod \ell\right) P + \sum (z_i h_i \bmod \ell) Q_i + \sum z_i R_i \stackrel{?}{=} (0,1)$$

   with one multi-scalar multiplication (Bos–Coster algorithm).
4. if failure, check signatures individually/in smaller batches.

For k signatures these are $k+1$ scalars of size ℓ and k of size 2^{128}.

Forgery $8s_iP \neq 8R_i + 8h_iQ_i$ passes with probability 2^{-128}.

Batch verification for Ed25519

Assume Bob needs to verify many signatures (m_i, Q_i, R_i, s_i).

1. Compute $h_i = H(R_i, Q_i, m_i)$ for each of them.
2. Pick random integers $z_i < 2^{128}$.
   Good enough for failure probability and cheaper to handle.
3. Compute batch verification as

$$-\left(\sum z_i s_i \bmod \ell\right) P + \sum (z_i h_i \bmod \ell) Q_i + \sum z_i R_i \stackrel{?}{=} (0,1)$$

   with one multi-scalar multiplication (Bos–Coster algorithm).
4. if failure, check signatures individually/in smaller batches.

For k signatures these are $k+1$ scalars of size ℓ and k of size 2^{128}.

Forgery $8s_iP \neq 8R_i + 8h_iQ_i$ passes with probability 2^{-128}.

To match security guarantees, test $8s_iP \stackrel{?}{=} 8R_i + 8h_iQ_i$ also for single Ed25519 signature verification