

CTF BlackAlps - Mixed Schnorr/ECDSA Nonce Reuse

Nathan Rayburn

November 2025

Introduction

The challenge ships a mixed-signature oracle built from `signatures.py`. It lets you request Schnorr or ECDSA signatures on arbitrary messages and then verify a provided signature on the fixed message “Flag please” to obtain the flag. The core flaw: both Schnorr and ECDSA signers deterministically derive the nonce from (d, P, m) in exactly the same way, so signing the same message under both schemes reuses the nonce k (and thus the same $r = x(kG)$). With one Schnorr signature (r, s_2) and one ECDSA signature (r, s_1) on the same message, we can solve for the private key d , forge a valid signature on “Flag please,” and retrieve the flag.

Challenge Setup

An oracle exposes two signers on secp256k1 (from `signatures.py`):

- Schnorr signature $\sigma_{\text{sch}} = (r, s_2)$ on a chosen message.
- ECDSA signature $\sigma_{\text{ecdsa}} = (r, s_1)$ on the same message.
- A flag option that verifies a signature on “Flag please” with either scheme.

Both signers reuse the same deterministic nonce k for a given (d, P, m) , so r matches across Schnorr and ECDSA when the message is the same.

Oracle Interaction

```
(sage) nathanrayburn@Nathans-MacBook-Air-2:~/chal_mix_signing % nc mix-signing.ctf.  
blackalps.ch 2560  
Welcome in our mixed signature system.  
  
My public key is  
a37b97df0c5a0f70b978fd3f1732ce0241a57141be19219437eacc5a6d26dbed  
cd1570705e2b053b247f9f2e9e0aea64a1aa5ebd50bd2164ad47875dc3da186d  
  
What do you want to do:  
1: Get a Schnorr signature  
2: Get a ECDSA signature  
3: Get the flag  
4: Quit  
1  
Give me a message to sign:  
test  
7d84d017ed925d7858454b0ba10870ad02cb65855c7e8e3618a71f84d00f5c59
```

```
dba379ac3cce807f94cb106c47a0e89240b84cf9ea50c8fa5dfb0880aca8a143
```

```
What do you want to do:  
1: Get a Schnorr signature  
2: Get a ECDSA signature  
3: Get the flag  
4: Quit  
2  
Give me a message to sign:  
test  
7d84d017ed925d7858454b0ba10870ad02cb65855c7e8e3618a71f84d00f5c59  
112458fb74f6936aa87282dcb359ea6502a1033d660dd8ee8036799ee1d18303
```

Key Code Snippets (signatures.py)

Shared nonce derivation:

```
1 k = int_from_bytes(tagged_hash("nonce",  
2     bytes_from_int(d) + P.to_bytes() + msg)) % n  
3 R = k*G
```

Listing 1: Shared nonce derivation

Schnorr signing core:

```
1 e = int_from_bytes(tagged_hash("challenge",  
2     R.to_bytes()[0:32] + P.to_bytes() + msg)) % n  
3 sig = R.to_bytes()[0:32] + bytes_from_int((k + e * d) % n)
```

Listing 2: Schnorr signing

ECDSA signing core:

```
1 z = int_from_bytes(tagged_hash("challenge", msg)) % n  
2 sig = bytes_from_int(r) + bytes_from_int(  
3     gmpy2.invert(k, n) * (z + r * d) % n)
```

Listing 3: ECDSA signing

These show the same k feeds both s computations when the message is identical.

Notation

Curve: secp256k1 with generator G , order n . Private key d , public key $P = dG$. Nonce k , $R = kG$, $r = x(R)$. Hashes:

$$z = H_{\text{tagged}}(\text{"challenge"}, m) \bmod n, \quad e = H_{\text{tagged}}(\text{"challenge"}, r \parallel P \parallel m) \bmod n.$$

Signatures

Schnorr.

$$s_2 = k + ed \pmod{n}, \quad \sigma_{\text{sch}} = (r, s_2).$$

ECDSA.

$$s_1 = k^{-1}(z + rd) \pmod{n}, \quad \sigma_{\text{ecdsa}} = (r, s_1).$$

Key-Recovery Algebra

With the same k :

$$\begin{cases} s_2 \equiv k + ed \pmod{n} \\ s_1 \equiv k^{-1}(z + rd) \pmod{n} \end{cases}$$

1. $k \equiv s_2 - ed \pmod{n}$.
2. Substitute into ECDSA, clear the inverse:

$$s_1(s_2 - ed) \equiv z + rd \pmod{n}.$$

3. Rearrange:

$$s_1s_2 - s_1ed \equiv z + rd \Rightarrow s_1s_2 - z \equiv d(r + s_1e).$$

4. Solve for d :

$$d \equiv (s_1s_2 - z)(r + s_1e)^{-1} \pmod{n}.$$

Recovery Script (`recover_key.py`)

Core computation:

```
1 numer = (s1 * s2 - z) % n
2 den   = (r + s1 * e) % n
3 inv_den = int(gmpy2.invert(den, n))
4 d = (numer * inv_den) % n
```

Listing 4: Recover private key

Inputs: public key hex (for e), Schnorr sig $r||s_2$, ECDSA sig $r||s_1$, and message m . The script checks r matches in both signatures, computes z and e , applies the formula, and sanity-checks $dG = P$.

Practical Use

Ask the oracle for Schnorr and ECDSA signatures on the same message (e.g., "test"), recover d with the formula, then forge a valid signature on "Flag please" and submit to get the flag.

Oracle Interaction Flag

```
What do you want to do:
1: Get a Schnorr signature
2: Get a ECDSA signature
3: Get the flag
4: Quit
3
Give me a signature of the message "Flag please":
97e8ba3a77e11cdfbff630adf8f62c8c324272fda8e565ccc6d6292223683639
c0aa17e396329106a239cc638e65741fe565cf3eb73331d8d53bf0ffb653339d
Well done here is the flag: BA25{Tolerance_gives_the_k3y}
```