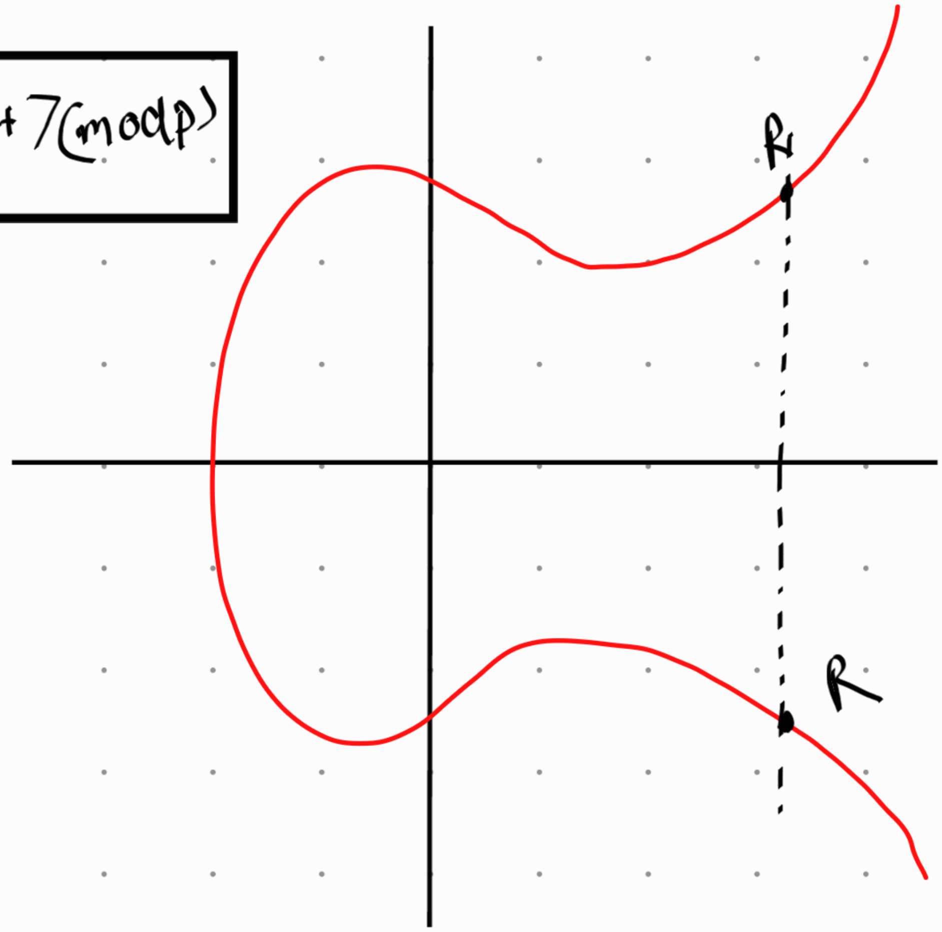
Description: Signature Malliability exploit

$$\boxed{y^2 = x^3 + 7 \pmod{p}}$$



ECDSA

Elliptic Curve Digital Signature Algorithm

e
Private Key

E
Public Key

For purposes of signing a Tx and minimizing the risk of private key leak we generate temporary private key.

R
Temp Priv Key

K
Temp Public Key

Key

(x, y)

r

$$\boxed{s = ((z + re) \times k^{-1}) \pmod n}$$

r = x Co-ordinate of Public Key on elliptic curve

s = derived with r

v = teles which of the two corrosponding public keys on the curve to use.

Root Cause

Since each K on a curve also has a corrosponding value, it is what causes signature maleability.

It is easy to compute the corrosponding s value that will work for the opposite Public Key. and provide a second valid signature that also recovers to the same signer.

Modular nature of SECP256K1 Allows:

$$s > \frac{n}{2} \text{ valid for } R$$

$$s' < \frac{n}{2} \text{ valid for } R'$$

where R is Public Key

$$n - s = s'$$

Example

```
pragma solidity 0.8.17;

contract signatureMealleable is Ownable {

    address token;
    mapping(bytes32 => bool) executed;

    function signedTransfer(
        address to,
        uint256 amount,
        uint8 v,
        bytes32 r,
        bytes32 s
    ) external {
        bytes32 msgHash = keccak256(abi.encode(msg.sender, to, amount));
        address signer = ecrecover(msgHash, v, r, s);
        require(signer == owner);

        bytes32 sigHash = keccak256(abi.encode(msgHash, v, r, s));
        require(!executed[sigHash]);
        executed[sigHash] = true;

        IERC20(token).safeTransfer(to, amount);
    }

}
```

An attacker can observe Tx v,r,s and then provide the second tampered v,r,s that recovers to the same owner and double spend from the contract.

Mitigation:

Restrict the s value to a single half of the range of n.

i.e. either $s > \frac{n}{2}$

or $s' < \frac{n}{2}$